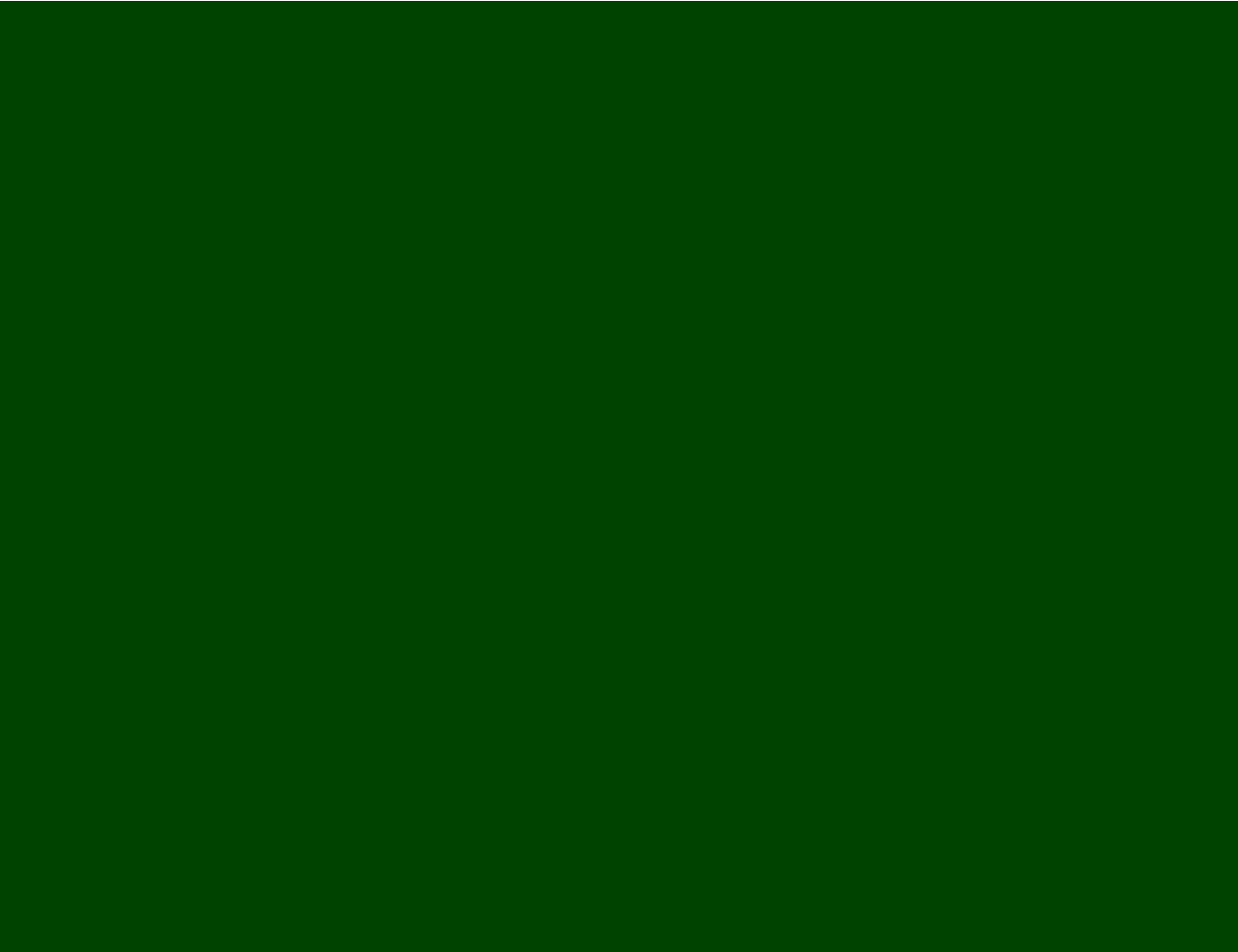




# **CCNP ROUTE Complete Guide**

**1st Edition**

Yap Chin Hoong



Dear valued customer,

Your investment of the CCNP ROUTE Complete Guide 1st Edition Companion CD will really worth it because it contains much valuable information that can enhance your CCNP studies. ☺  
Kindly download the Companion CD by following the instructions at \*link removed\*.

The **Dynamips** folder contains a FREE software that provides a tool to simulate real Cisco routers (and switches) for your CCNP practices. It is so powerful that can simulate any real Cisco IOS commands because it actually loads and runs real Cisco IOS software. ☺

Setup the Dynamips/Dynagen using a tutorial file included in the folder. However, you may face some issues with Telnet in Windows Vista and Windows 7. Try to Google around to solve that, it isn't that difficult. ☺

The **MISC Tools and Guides** folder contains some extra info regarding Dynamips. Actually you don't really need to look into it. It contains the tools and guides when you wanted to use other IOS files other than those provided in the **IOS** folder in the CD. The **VBUnzip** is actually a tool used to extract Cisco IOS files. So when Dynamips load an extracted IOS image file, it doesn't need to extract it because it is already extracted. This will speed up the boot up time of the IOS. If you managed to see how real Cisco routers boot, you will see "extracting images.....". Basically we want to skip that step in the simulation. ☺

The **Lab Setups** folder contains all the labs setup using Dynamips according to the CCNP ROUTE Guide. Whenever you saw a network diagram with some routers and IP addresses, and feel like wanted to see how it works yourself. You may first look at the page number in the CCNP ROUTE Guide, then heads towards the **Lab Setups** folder, most likely that there is a lab for it. Copy it out to your desktop, extract it, launch the Dynamips engine, and run the Network.net file for the lab, the lab should be loaded in 10 seconds. Console into every routers, copy and paste the basic configuration into the routers (the config files are included in the folder for a particular lab setup itself). TATA! You are ready to practice the commands according to the CCNP ROUTE Guide. Just follow the commands and you will be able to see how things work. All commands in the CCNP ROUTE Guide have been fully tested and working fine. ☺

Basically we can setup Cisco labs and practice Cisco IOS commands in 2 minutes time. ☺ Before this, we would need to look for real routers, power cords, UTP network cables, power them on, took 5 minutes, clear the configuration, etc. From the time we are motivated to practice until the lab is up and ready for practice (maybe take able 30 minutes), we may already feel tired and say: "OK, let me watch a movie and come back to this later...". ☺ Hope you get the idea of using this wonderful tool.

Finally, the **Proof of Concepts** folder contains many packet captures and command output captured for the various topics throughout the CCNP ROUTE Guide. Download and install **Wireshark** <http://www.wireshark.org/> to view the packet capture files. Packet captures shows the bits and bytes of network packets. Basically I spend many days and nights capturing them to prove how networking works, and documented them down in the CCNP ROUTE Guide. Basically most of the concepts have been proven using Cisco IOS commands and real network packets. Hope you get the idea. ☺

The files in the **Proof of Concept** folder are basically used to enhance you learning experience. Those info are saved separately there because it will overwhelm the most of the readers and make the CCNP ROUTE Guide too lengthy if everything is included in the CCNP ROUTE Guide itself. ☺

OK, I have briefed the overall usages of the Companion CD. Have fun and keep in touch! ☺

Regards,  
YapCH

# CCNP ROUTE Complete Guide 1st Edition

Copyright © 2010 Yap Chin Hoong

[www.yapchinhoong.com](http://www.yapchinhoong.com)

Chapter	Title	Page
Chapter 1	Designing IP Networks	1
Chapter 2	Advanced IP Addressing	5
Chapter 3	IPv6	11
Chapter 4	On-Demand Routing, RIPv2, and Routing Principles	37
Chapter 5	EIGRP	49
Chapter 6	EIGRP Lab	65
Chapter 7	OSPF in a Single Area	83
Chapter 8	OSPF in a Single Area Lab	99
Chapter 9	Interconnecting Multiple OSPF Areas	115
Chapter 10	Advanced OSPF – OSPF Stub Areas and OSPF Virtual Links	135
Chapter 11	Route Redistribution and Manipulating Routing Updates	151
Chapter 12	Policy-Based Routing and IP SLA (Service-Level Agreement)	175
Chapter 13	Basic BGP	189
Chapter 14	Basic BGP Lab	219
Chapter 15	BGP Route Summarization, Route Filtering, and Route Reflection	231
Chapter 16	Advanced BGP – Path Manipulation and Multihoming	251
<b>Bonus Chapters</b>		
Chapter 17	Integrated IS-IS	263
Chapter 18	Integrated IS-IS Lab	295
Chapter 19	IP Multicast Routing	309
Chapter 20	IP Multicast Routing Lab	325
Appendix 1	Cisco IOS Architecture	335
Appendix 2	Cisco IOS Packet Switching Architectures	341
Appendix 3	Cisco IOS Image Naming Convention, Packaging, and Deployment	353
Appendix 4	ICMP and ICMPv6 Type and Code Numbers	357
Appendix 5	Netmask Table	360
Appendix 6	CCNP ROUTE Extra Knowledge	361



## About the Author

Yap Chin Hoong is a senior network engineer with a computer network consulting firm at Malaysia. He found great satisfaction when conveyed complex networking concepts to his peers. Yap holds a bachelor's degree in Information Technology from Universiti Tenaga Nasional.

When not sitting in front of computers, Yap enjoying playing various types of musical instruments. Visit his YouTube channel during your study breaks. :-)

**Facebook:**

<http://www.facebook.com/yapchinhoong>

**Website:**

<http://www.itcertguides.com/>

**YouTube:**

<http://www.youtube.com/user/yapchinhoong>

## Chapter 1

# Designing IP Networks

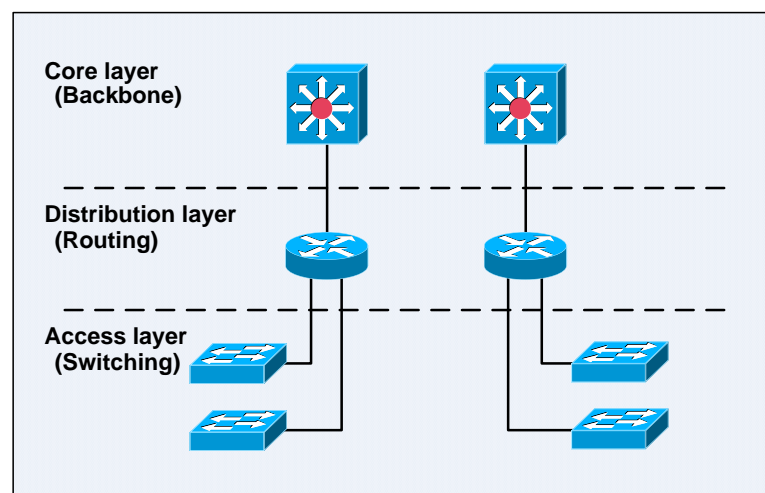
- Proper network design with efficient use of addressing structure is able to reduce the size of routing tables and conserve network resources.
- This chapter explains **why** there is a need for hierarchical structure and design. The next chapter describes **how** to design networks with hierarchical addressing scheme to support VLSM and route summarization.
- Generally, a corporate organizational structure does affect its network design. The structure of a scalable and hierarchical network design often reflects a corporation's information flow.
- There are 2 types of hierarchical network design:

<b>Functional Structured Design</b>	Divisions of an organization with different scope of operations (eg: finance, marketing, IT, etc) have their own networks and are connected according to their <b>functional purposes</b> within the organization. The network architecture often follows the organizational chart.
<b>Geographic Structured Design</b>	Most retail corporations are organized by geographical location of retail stores. The divisions of the corporation have their own networks which are organized and connected according to their <b>locations</b> (eg: countries, states, or provinces). [local retail stores → regional offices → HQ]

- The geographic network structure is more cost-effective as fewer network links are required.

## Cisco Hierarchical Design Model

- Defined by Cisco to simplify the design, implementation, and maintenance of responsive, scalable, reliable, and cost-effective networks.
- The 3 layers are **logical and not physical** – there may be many devices in a single layer, or a single device may perform functions of 2 layers.



**Figure 1-1:** The Cisco Hierarchical Model



- Below are the 3 layers in the Cisco Hierarchical Model:

<b>Core layer</b>	Also referred to as the <b>backbone layer</b> . It is responsible for transferring large amounts of traffic reliably and quickly – switches traffic as fast as possible. A failure in the core can affect many users; hence fault tolerance is the main concern in this layer. The core layer should be designed for <b>high reliability, high availability, high speed, and low convergence</b> . Do not support workgroup access, implement access lists, VLAN routing, and packet filtering which can introduce latency to this layer.
<b>Distribution layer</b>	Also referred to as the <b>workgroup layer</b> . Its primary functions are routing, Inter-VLAN routing, defining or segmenting broadcast and multicast domains, network security and filtering with firewalls and access lists, WAN access, and determining (or filtering) how packets access across the core layer.
<b>Access layer</b>	Also referred to as the <b>desktop layer</b> . Here is where end systems gain access to the network. The access layer (switches) handles traffic for local services (within a network) whereas the distribution layer (routers) handles traffic for remote services. It mainly creates separate collision domains. It also defines the access control policies for accessing the access and distribution layers.

- In a hierarchical network, traffic on a lower layer is only allowed to be forwarded to the upper layer after it meets some clearly defined criteria. Filtering rules and operations restrict unnecessary traffic from traversing the entire network, which results in a more responsive (lower network congestion), scalable (easy to grow), and reliable (higher availability) network.
- A clear understanding of the traffic flow patterns of an organization helps to ensure the placement of network devices and end systems within the organization.

- Below are some considerations for hierarchical layer network designs:

<b>Full-Meshed Core Layer</b>	In this core layer design, all routers between headquarters and other divisions have <b>direct connections</b> to all other routers, which allow the network to react quickly upon a link failure. This design is more practical for small organizations with limited number of offices as its implementation cost is very high for large organizations.
<b>Hub-and-Spoke Core Layer</b>	This core layer design addresses the limitations faced in full-mesh design by introducing regional data centers. Data travels to a centralized headquarters where the corporate databases and network services reside.
<b>Access and Distribution Layers</b>	End users or customers at remote sites gain access to network services through the access layer; while the distribution layer provides connectivity between the remote and local sites. Services such as DHCP and DNS should be placed in the distribution layer if there is no benefit of having duplicated services at the remote sites. A hub-and-spoke topology is recommended to connect remote sites to at least 2 distribution layer devices for redundancy and easier maintenance.

- The formula for calculating the number of links in a full mesh network that has  $n$  nodes is  $\frac{n(n-1)}{2}$
- A well-designed large-scale internetwork with an effective scalable IP addressing plan has many benefits, eg: **scalable, flexible, predictable**, and able to **reduce the size of routing tables** through route summarization.

- Below are some benefits and characteristics of a good network design:

<b>Scalability</b>	Allows for significant increases in the number of sites, and facilitates the process of adding routers to an existing network. When 2 companies merge, and both use 172.16.0.0 private addresses, there will be likely some overlapping addressing spaces. A scalable network that integrates private addressing with a good IP addressing plan minimizes the impact of merging networks (additions or reorganizations). It allows the companies to connect at the core layer, and implements NAT as a temporary solution to <b>translate overlapping address space</b> to an unused address space. The overlapping network number can then be changed later on the network devices, DHCP servers, and endpoint hosts in the new network.
<b>Predictability</b>	The <b>behavior</b> and <b>performance</b> of a scalable network is predictable. Packets are load-balanced when equal-cost paths exist between any 2 routers in the network. When a circuit or router fails, an alternative equal-cost path that exists in the routing table can be used without any recalculation. This reduces convergence times and produces a <b>predictable traffic pattern</b> .
<b>Flexibility</b>	Minimizes the impact of <b>unexpected growth, restructuring</b> or <b>downsizing</b> of an organization network.

- An optimized IP addressing plan uses a **hierarchical addressing scheme**. Below describes some benefits of using hierarchical addressing:

<b>Reduced number of routing table entries</b>	<b>Route summarization</b> should be used for keeping routing tables as small as possible by having a single IP address that represents a group of IP addresses. Other benefits are <b>more efficient routing, reduced CPU cycles</b> for finding the best path, <b>reduced memory requirements, conserves bandwidth</b> (fewer routing updates), <b>faster convergence</b> upon topology changes, <b>easier troubleshooting</b> , and <b>increased network stability and availability</b> .
<b>Efficient allocation of addresses</b>	Hierarchical addressing <b>makes use of all possible addresses</b> by grouping them contiguously; compared to unplanned address assignment, which might end up <b>wasting</b> groups of addresses.

- **Flat networks** are networks in which devices are connected to a single large collision and broadcast domains. **Flat addressing** does not use a logical hierarchical addressing scheme. Route summarization and the benefits of hierarchical addressing scheme are not applicable for networks designed and implemented with flat addressing scheme.
- **Hierarchical addressing** often uses **Variable-Length Subnet Masks (VLSMs)** and **Classless Interdomain Routing (CIDR)** to implement an effective IP addressing plan which is crucial for the scalability and the implementation of route summarization for a network.
- The difference between route summarization and CIDR is as below:
  - i) Route summarization is generally done up to the classful network number boundary. (Fixed masks – /8, /16, /24).
  - ii) CIDR is commonly used to combine and summarize several classful networks and goes beyond the classful network number boundary. (Flexible masks).
- **Collapsed core** or **collapsed backbone** referred to as a network with no distribution layer where all network segments are connected to each other through an internetworking device.

- A single point of failure is any device, interface on a device, or link that can isolate users from the services they depend on if it fails. Networks that follow a strict hierarchical model tend to have many single points of failure due to the emphasis on summarization points and clean points of entry between the layers.

Redundancy provides alternate paths around these failure points, providing some measure of safety against loss of service. However, redundancy, if not designed and implemented properly, can cause more trouble than it is worth, as each redundant link and connection point in the network weakens the hierarchy and reduces stability.

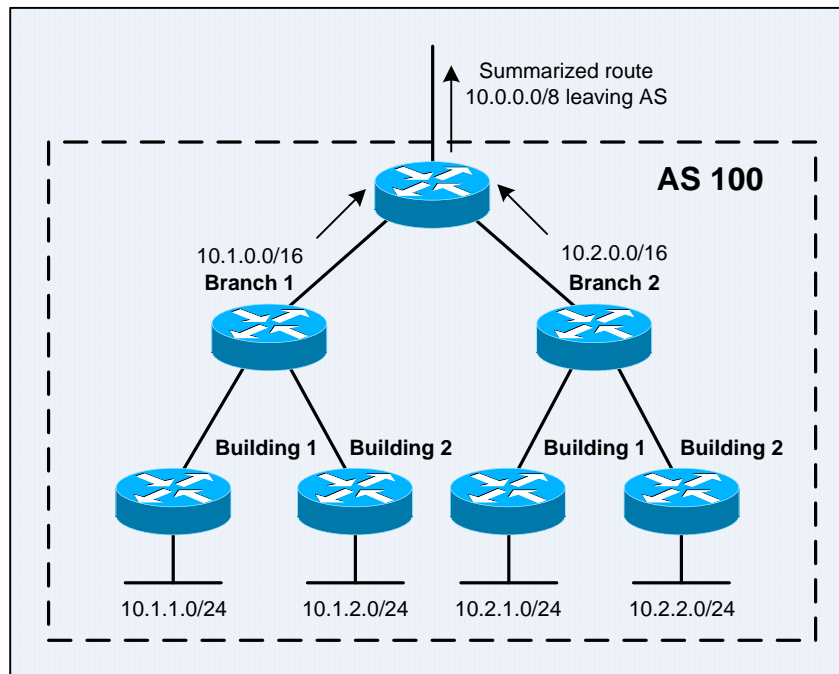
## Chapter 2

# Advanced IP Addressing

- Scalable and stable networks are the result of good network design with a planned IP addressing scheme and effective implementation planning. The use of hierarchical addressing and the capability to manipulate traffic flow results in a network that is designed to grow.
- Network problems often start to occur as the size of routing table increases, in which more CPU resources are required for topology convergence, and the delays caused by routing table lookup in large routing tables. These problems can be resolved with route summarization and CIDR.
- Advanced IP addressing techniques such as NAT and VLSM are being used to implement route summarization and CIDR in controlling the size of routing tables.
- The difference between route summarization and CIDR is as below:
  - i) Route summarization is generally done up to the classful network number boundary. (Fixed masks – /8, /16, /24).
  - ii) CIDR is commonly used to combine and summarize several classful networks and goes beyond the classful network number boundary. (Flexible masks).
- NAT allows the use of a private addressing space within an organization while using globally unique addresses for Internet access. Different address pools may be used for different groups of users, which can ease the management of the network.
- VLSM is an advanced feature that allows the best use of the available address spaces.
- The current solution for address depletion or exhaustion is private addressing and NAT. The long-term solution is IPv6.

## IP Addressing Design

- A network that is designed with a hierarchical addressing scheme supports VLSM, CIDR, and route summarization.
- Below are some problems faced by unsummarized large networks:
  - i) Excessive **unnecessary bandwidth usage** for **high volume of routing updates**, which also introduces **unnecessary workloads** (perform more routing table lookups) for routers.
  - ii) **Extra CPU and memory resources usage** for updating all routing tables upon a route change. Ex: SPF calculations which performed by OSPF are expensive, as each router needs to recalculate all paths to all networks.
- RIP, IGRP, RIPv2, and EIGRP perform autosummarization at their classful boundaries; whereas OSPF and IS-IS require manual configuration to implement route summarization.
- Kindly refer to Chapter 15: Variable-Length Subnet Masks and Route Summarization, CCNA Complete Guide 2nd Edition for the review on VLSM and route summarization.
- Kindly refer to Chapter 17: Scaling the Internet with CIDR and NAT, CCNA Complete Guide 2nd Edition for the review on CIDR and NAT.



**Figure 2-1: Hierarchical and Scalable Addressing allows Summarization**

- There are some other methods other than CIDR, and VLSM that can be used as the solutions for address exhaustion, eg: IP unnumbered. **IP unnumbered** is useful on point-to-point serial links. It can conserve one subnet per point-to-point link by allowing them to have no IP address assigned. Each end of the serial line borrows an IP address from another interface on the router whenever an address is required (a source address is always required when generating a packet).

## The Internet Authoritative Bodies

- They belong to the group within the Internet community that is responsible for assigning unique classful networks. Everything started with the government-funded IANA, which is being commercially administered by **Networks Solutions of Herndon, Virginia** recently. On 25/11/1998, the **Internet Corporation for Assigned Names and Numbers (ICANN)**, a nonprofit corporation managed by the US government, was officially recognized to perform administrative functions for the Internet, eg: coordinating the assignment of protocol parameters, managing the domain name and root server systems, and allocating IP addresses.
- The growth of the Internet has led to regional organizations for the allocation of IP addresses. **Regional Internet Registries (RIRs):**
  - American Registry for Internet Numbers (ARIN)**, (<http://www.arin.net>) serves North America, and parts of Caribbean.
  - Réseaux IP Européens (RIPE)**, (<http://www.ripe.net>) serves Europe, Middle East, and Central Asia.
  - Latin American and Caribbean Internet Addresses Registry (LACNIC)**, (<http://www.lacnic.net>) serves Central and South America, and Caribbean.
  - African Region Internet Registry (AfrINIC)**, (<http://www.afrinic.net>) serves Africa.
  - Asia Pacific Network Information Center (APNIC)**, (<http://www.apnic.net>) serves Asia, and Pacific Ocean regions.

### Domain registration:

- The Internet's Network Information Center (InterNIC)**, (<http://www.internic.net/>)

## Network Address Translation

- Below are the main features and usages of NAT as supported by Cisco IOS:
  - i) **Static NAT**, a manually configured one-to-one address translation.
  - ii) **Dynamic NAT**, a pool of addresses that is defined and used for address translation.
  - iii) **Port Address Translation (PAT)**, a group of local addresses (normally within an organization) is translated into a single globally unique public address. IP addresses along with port numbers ensure the uniqueness of different connections.
  - iv) **Overlapping Addresses Translation**, commonly being used when companies merge.
  - v) **Destination Address Rotary Translation**. Also known as **TCP load distribution**, as it can be used only for TCP traffic.
  
- **TCP load distribution** is a dynamic form of NAT that can be configured for **outside-to-inside** traffic (only for connections that is opened from the outside to the inside). A destination address that matched against an access list is translated or replaced with an address from a **rotary pool** in round-robin basis.

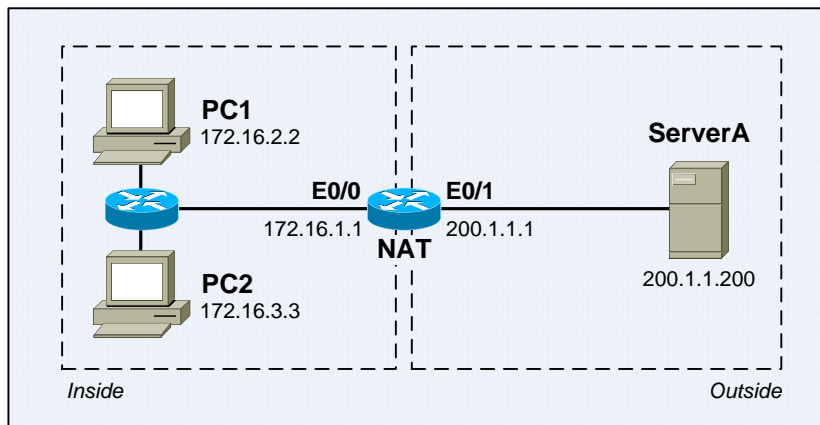


Figure 2-2: Network Setup for NAT

### Standard Access Lists Translation Configuration

- Configures NAT to meet the following requirements:
  - i) For packets with a source address of 172.16.2.x, translate them using the NAT pool of addresses defined in sales\_pool.
  - ii) For packets with a source address of 172.16.3.x, translate them using the NAT pool of addresses defined in marketing\_pool.
  
- Standard Access List Translation configuration on NAT:

```
NAT#conf t
Enter configuration commands, one per line. End with CNTL/Z.
NAT(config)#ip nat pool sales_pool 200.1.2.1 200.1.2.254 prefix-length 24
NAT(config)#ip nat pool marketing_pool 200.1.3.1 200.1.3.254 prefix-length 24
NAT(config)#ip nat inside source list 1 pool sales_pool
NAT(config)#ip nat inside source list 2 pool marketing_pool
NAT(config)#
NAT(config)#access-list 1 permit 172.16.2.0 0.0.0.255
NAT(config)#access-list 2 permit 172.16.3.0 0.0.0.255
NAT(config)#^Z
NAT#
```

## Extended Access Lists Translation Configuration

- Configures NAT to meet the following requirements:
  - i) Only translate packets from the 172.16.1.0/24 subnet.
  - ii) For packets with a destination address to either 10.0.0.0/24 or 10.0.1.0/24 subnet, translate them from the NAT pool of addresses defined in trusted\_pool.
  - iii) For packets with a destination address that does not match either 10.0.0.0/24 or 10.0.1.0/24 subnet, translate them from the NAT pool of addresses defined in untrusted\_pool.
- Extended Access List Translation configuration on NAT:

```
NAT#conf t
Enter configuration commands, one per line. End with CNTL/Z.
NAT(config)#ip nat pool trusted_pool 200.1.4.1 200.1.4.254 prefix-length 24
NAT(config)#ip nat pool untrusted_pool 200.1.5.1 200.1.5.254 prefix-length 24
NAT(config)#ip nat inside source list 101 pool trusted_pool
NAT(config)#ip nat inside source list 102 pool untrusted_pool
NAT(config)#
NAT(config)#access-list 101 permit ip 172.16.1.0 0.0.0.255 10.0.0.0 0.0.0.255
NAT(config)#access-list 101 permit ip 172.16.1.0 0.0.0.255 10.0.1.0 0.0.0.255
NAT(config)#access-list 102 permit ip 172.16.1.0 0.0.0.255 any
NAT(config)#^Z
NAT#
```

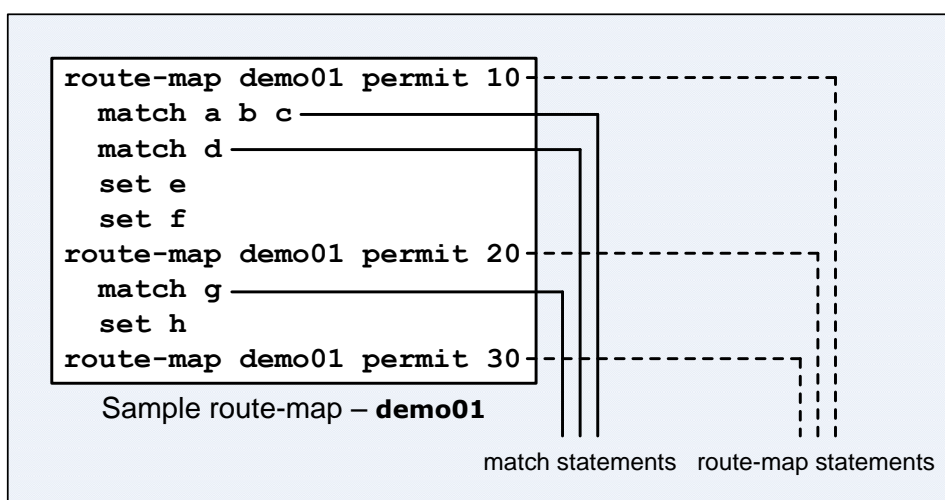
## Route Maps

- Route map is a Cisco IOS feature that serves a variety of purposes. This section compares the results of NAT configuration with a route map and NAT configuration with an access list.
- In NAT configuration with an access list, the NAT table has only **simple translation entries**, which shows only the translation between the inside local and inside global addresses. It does not include any TCP or UDP port numbers information as well as the packet's destination address. It would be difficult to troubleshoot connectivity problems with only these information.

```
NAT#sh ip nat translations
Pro Inside global      Inside local      Outside local      Outside global
--- 200.1.2.1           172.16.2.2       ---                ---
--- 200.1.3.1           172.16.3.3       ---                ---
NAT#
```

- Simple translation entries might also prevent proper translation among multiple address pools. Ex: The 1st session that matched the 1st address pool creates a NAT entry. The 2nd session initiated by the same source host to a different host won't be translated again with the 2nd address pool, as the source address would match the NAT entry created during the 1st session. Route maps can be used to distinguish between different sessions.
- PAT and route map are the available methods that can produce **extended translation entries**.

- Route maps are complex ACLs that use **match** commands to test some conditions upon interesting packets or routes. Once the conditions are matched, the actions specified by **set** commands will be taken to modify the attributes of the packet or routes.
- A route map is a collection of route map statements that have the same route map name. Within a route map, each route map statement is numbered and can be edited individually. Like an access list, there is an implicit **deny any** at the end of a route map. The consequences of this deny depend upon the usage of the route map.
- A single **match** statement may contain multiple conditions; just a **single** condition needs to be true for the **match** statement to be considered matched. (Logical OR)  
A single route map statement may contain multiple **match** statements; **all match** statements in the route map statement must be true for the route map statement to be considered matched.  
**Multiple match conditions → A match statement / clause.** (Logical AND)  
**Multiple match statements / clauses → A route map statement.**  
**Multiple route map statements → A route map.**



**Figure 2-3: Route Map Interpretation**

- The sample route map named **demo01** in Figure 2-3 is interpreted as:  

```

if ((a or b or c) and d)
  set e and f
else if (g)
  set h
else
  set nothing

```
- The **route-map** {*map-tag*} [**permit** | **deny**] [*seq-num*] global configuration command can be used to define the conditions for NAT. The *map-tag* is the name of the route map. The **permit** and **deny** are optional parameters that specify the action to be taken when a route map match conditions are met. The optional sequence number indicates the position for a new route map statement in an already existed route map (used for inserting or deleting specific route map statements in a route map).
- **Note:** The default action for the **route-map** command is **permit**, with sequence number of 10. The actions defined with the **set** {*condition*} route map configuration command will be effective only when the action of the route map is **permit**.  
**Note:** Do not leave out the *seq-num* when editing and adding statements in a route map list, or else only the 1st statement with the sequence number of 10 will always be referred to. Route map sequence numbers **do not automatically increment** as with ACL configuration!



- Alternative NAT with Route Map configuration on NAT:

```

NAT#clear ip nat translation *
NAT#conf t
Enter configuration commands, one per line. End with CNTL/Z.
NAT(config)#ip nat pool sales_pool 200.1.2.1 200.1.2.254 prefix-length 24
NAT(config)#ip nat pool marketing_pool 200.1.3.1 200.1.3.254 prefix-length 24
NAT(config)#ip nat inside source route-map rm_sales pool sales_pool
NAT(config)#ip nat inside source route-map rm_marketing pool marketing_pool
NAT(config)#
NAT(config)#route-map rm_sales
NAT(config-route-map)#match ip address 1
NAT(config-route-map)#exit
NAT(config)#route-map rm_marketing
NAT(config-route-map)#match ip address 2
NAT(config-route-map)#exit
NAT(config)#
NAT(config)#access-list 1 permit 172.16.2.0 0.0.0.255
NAT(config)#access-list 2 permit 172.16.3.0 0.0.0.255
NAT(config)#^Z
NAT#
NAT#sh route-map
route-map rm_marketing, permit, sequence 10
  Match clauses:
    ip address (access-lists): 2
  Set clauses:
    Policy routing matches: 0 packets, 0 bytes
route-map rm_sales, permit, sequence 10
  Match clauses:
    ip address (access-lists): 1
  Set clauses:
    Policy routing matches: 0 packets, 0 bytes
NAT#

```

- The **clear ip nat translation \*** privileged command can be used to forcefully remove all active NAT translation mappings. Issue this command with caution as it will terminate and interrupt all existing active NAT connections.
- Below shows the output of the **show ip nat translations EXEC** command when PC1 accesses an Internet server – ServerA via Telnet and HTTP. Extended NAT translation entries are produced as a result of route maps and access lists configuration.

```

NAT#sh ip nat translations
Pro Inside global      Inside local      Outside local     Outside global
tcp 200.1.2.1:1050      172.16.2.2:1050  200.1.1.200:23   200.1.1.200:23
tcp 200.1.2.1:1051      172.16.2.2:1051  200.1.1.200:80   200.1.1.200:80
NAT#

```

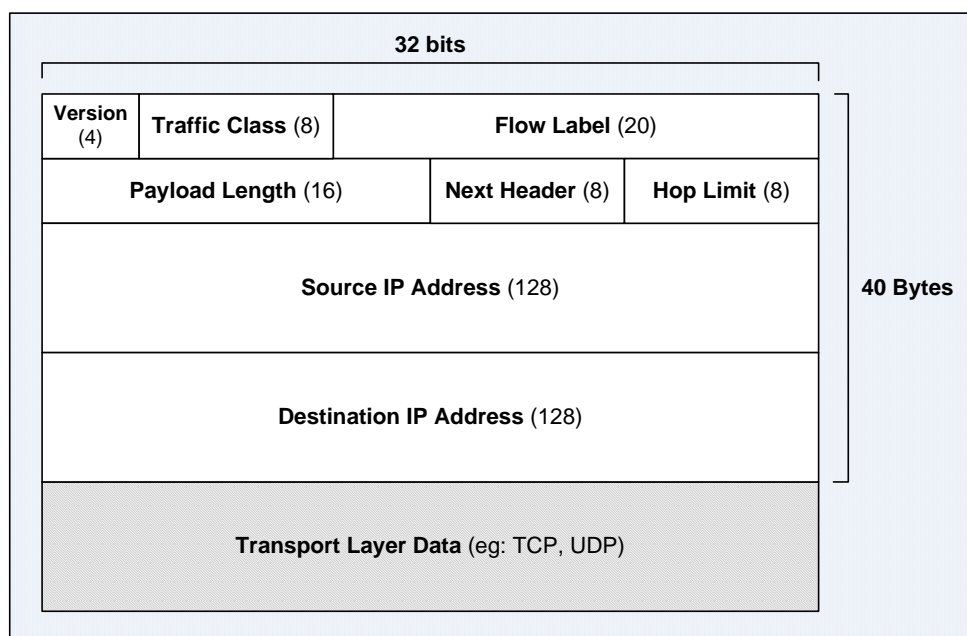
## Chapter 3

# IPv6

- IPv6 is the solution for many limitations in IPv4. However, IPv6 is not yet vastly deployed due to the overwhelming tasks of readdressing and upgrading of existing networks and applications.
- Below are some benefits of implementing IPv6:
  - i) Larger address space provides better support for more granular hierarchical addressing, greater number of addressable nodes, and simpler autoconfiguration of addresses.
  - ii) The simpler and fixed-size header enables better routing efficiency and performance.
  - iii) Various transition mechanisms, eg: **dual stack**, **tunneling**, and **translation** allow existing IPv4 networks to coexist with IPv6 features.
  - iv) Provides **native support** for new mobility and security standards – Mobile IP and IPsec.
  - v) Security and QoS can be implemented more efficiently with end-to-end connectivity instead of intermediate address translations (IPv6 eliminates the need for deploying NAT).
- Mobility provides roaming service for mobile devices (eg: Global Positioning Systems, IP phones) without losing connectivity and interrupting the current connection. Mobile IP is available for both IPv4 (as an add-in) and IPv6 (built-in).
- IPsec ensures better security (integrity, authentication, and confidentiality) for IPv6 networks. It is available for IPv4 and is **mandatory** for IPv6 – it is enabled and available on all IPv6 nodes. IPsec support and implementation is a mandatory part of IPv6 but is not an integral part of IPv4. However, due to the slow uptake of IPv6, IPsec is commonly used to secure IPv4 traffic.
- A **node** is a device that implements IPv6, be it a host or a router.  
A **host** is a node that is not a router.  
A **link** is equivalent to a network or a broadcast domain.  
A **prefix** is equivalent to a subnet.

## IPv6 Header Format

- The IPv6 header has been simplified to have fewer fields for easier, faster and efficient packet processing, enhanced performance, and routing efficiency.
- With the design and implementation of the fewer fields and 64-bit aligned fields, IPv6 is able to take advantage of the upcoming 64-bit processors for faster and efficient processing.
- IPv6 basic header has a fixed length of 40 bytes.
- Since most current link-layer technologies are relatively reliable and perform error detection, the **IP header checksum** is considered redundant and hence **has been removed**. Without the IP header checksum, both the connection and connectionless transport layer protocols are required to perform error detection and recovery. The removal of the IP checksum field further reduces the network layer processing time, as routers can concentrate solely on forwarding packets.
- If checksumming is required, it can be done via an AH header which provides cryptographically strong authentication and eventually a checksum for the whole packet.



**Figure 3-1: IPv6 Datagram Format**

- The IPv6 header comprises of the following 8 fields:

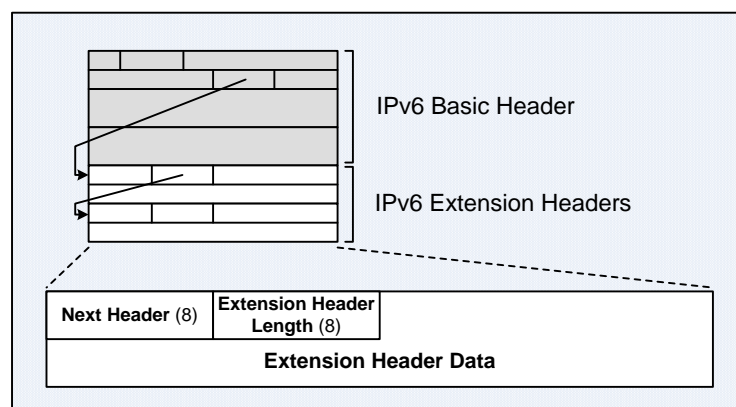
Field	Description
<b>Version</b>	Indicates the IP version. Always contains 0110 (6 in decimal – IPv6).
<b>Traffic Class</b>	Similar and functions the same as the Type of Service field in IPv4. Used to tag the packet with a traffic class that can be used in <b>Differentiated Class of Service (DiffServ)</b> . IPv6 allows this field to be rewritten at each router hop.
<b>Flow Label</b>	A new field introduced in IPv6 used to tag or label packets in a particular traffic flow – packets that are not just originated from the same source to the same destination, but belong to the same application at the source or destination. This allows faster identification and differentiation of packets at the network layer – routers no longer required to process the application data to identify the flow, as the information is available in the packet header. An advantage of differentiating traffic flows is that when load balancing traffic across multiple paths, the packets that belong to the same flow are always forwarded across the same path to prevent possible packet reordering at the destination. It can also be used for multilayer switching techniques and achieve faster packet-switching performance, eg: QoS for IPsec-encrypted packets.
<b>Payload Length</b>	Similar to the Total Length field in IPv4. Used to indicate the total length of application data (IP Payload). <b>Note:</b> Finding the payload length in an IPv4 packet requires the subtraction of the Header Length field from the Total Length field. <b>Note:</b> The IPv4 Total Length field is 16 bit; the IPv6 Payload Length field is 20 bits. Theoretically IPv6 packets are capable of carrying larger payload (1,048,575 bytes in IPv6 vs 65,535 bytes in IPv4).
<b>Next Header</b>	Similar to the Protocol field in IPv4. Used to specify the type of header following the basic header – a transport layer (TCP, UDP) header, or an IPv6 extension header. IPv6 uses extension headers to manage optional header information. Refer to the next section for more info.

<b>Hop Limit</b>	Similar to the TTL field in IPv4. Used to specify the maximum number of hops that a packet can pass through before it is considered invalid. Each router decrements the value by 1 without recalculating the checksum (there is no checksum field in the IPv6 header). Recalculation costs processing time on IPv4 routers.
<b>Source Address</b>	Indicates the source address of an IPv6 packet.
<b>Destination Address</b>	Indicates the destination address of an IPv6 packet.

## IPv6 Extension Headers

- Instead of having the Options field as in IPv4 header, IPv6 attaches extension headers to the end of a basic or extension header, with the 8-bit Next Header field specifying the next extension header if any. The use of extension headers allows **faster processing** and **protocol evolution**.
- Extension headers are 64-bit in length and the number of extension headers in an IPv6 packet is **variable**. Extension headers are daisy-chained **one after another** with the Next Header field of the previous basic or extension header specifies the current extension header. The last extension header (or the basic header if extension header is not used) has a Next Header field specifies a transport layer protocol, eg: TCP, UDP.
- The use of extension headers allows **end-to-end security**, as no firewalls and NAT are involved.
- Mobility provides roaming service for mobile devices (eg: IP phones) without interrupting the current connection. The IPv6 routing header allows an end system to change its source IP address with a stable **home address**, and hence allows the roaming address to maintain mobility.
- Cisco IOS Mobility IP is a tunneling-based solution that uses **Cisco GRE** or **IP-in-IP** tunnel. Tunneling allows a router on a device's home subnet to transparently forward IP packets to the roaming devices. IPv4 offers Mobile IP via **triangle routing**, where data is tunneled back to the home network before being forwarded to the final destination. However, this approach is less efficient than Mobile IPv6.

GRE is referred to as **Generic Routing Encapsulation**, a Cisco-proprietary tunneling protocol. It forms (unencrypted) virtual point-to-point links which are able to encapsulate a variety of protocols inside IP packets.



**Figure 3-2: IPv6 Extension Header**

- IPv6 has 6 types of extension headers. When multiple extension headers are used in the same packet, the order of the extension header as specified in RFC 1883 – IPv6 Specification is as below:

**Note:** The source node must follow this order; while the destination node may receive in any order.

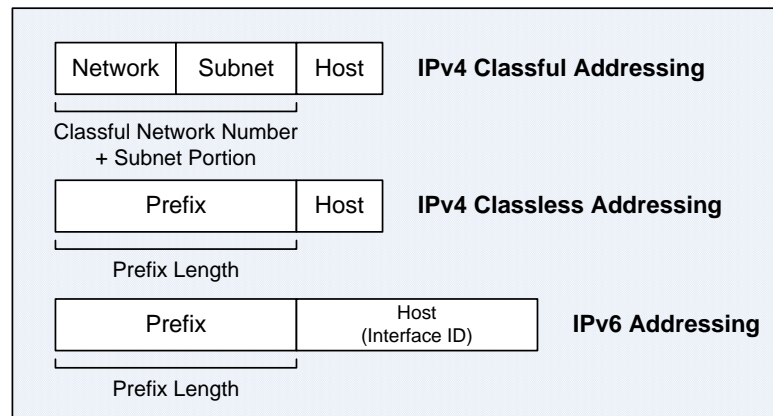
<b>Hop-by-Hop Options header (0)</b>	Used for the Router Alert (RSVP and MLDv1) and the IPv6 Jumbogram. It is being processed at all nodes along the path. <b>Note:</b> MLD → <b>Multicast Listener Discovery</b> . IPv6 routers use MLD to discovery nodes that want to receive multicast packets destined to a specified multicast address. <b>Note:</b> Jumbograms (RFC 2675 – IPv6 Jumbograms) are packets that contain payload larger than 65,535 bytes – the maximum packet size supported by the 16-bit Payload Length field as in basic IPv6 header.
<b>Destination Options header (60)</b>	It is processed at the <b>destination node</b> when it follows an ESP header; or at <b>intermediate node</b> (eg: routers) as specified in the Routing header when it follows a hop-by-hop options header.
<b>Routing header (43)</b>	Specify the routing path in source routing and Mobile IPv6. A source node uses the Routing header to list the addresses of routers that the packet must pass through. Intermediate routers will use the addresses as destination addresses of the packet when forwarding the packet from one router to another. The final destination host will process the next header following the routing header. When there are multiple ISPs, the Routing header allows a router to specify which ISP to use.
<b>Fragment header (44)</b>	It is used in fragmented packets when the application does not perform PMTUD and hence the source node must fragment a packet that is larger than the MTU of the path to the destination. It contains the Fragment Offset, Identification, and More Fragment fields that were removed from the basic header. It is used in each fragmented packet.
<b>Authentication header (AH) (51) and Encapsulating Security Payload (ESP) header (50)</b>	Used in IPsec to provide authentication, integrity, and confidentiality of IPv6 packets. These headers are identical for both IPv4 and IPv6.
<b>Upper-Layer header</b>	Identify the transport layer header, eg: TCP (6) and UDP (17).

**Note:** With IPv6, only the originating nodes can fragment packets; IPv6 routers no longer perform fragmentation. Originating node must either perform Path MTU Discovery (PMTUD) to find the lowest MTU along the path to the destination or never produce packets larger than 1280 bytes. All links that support IPv6 must be able to support at least 1280-byte packet size so originators can use the minimum-packet-size option rather than performing PMTUD if intended.  
**Note:** AH and ESP extension headers are identical for both IPv4 and IPv6 IPsec. IPsec is a network layer security mechanism.

- The value of the Next Header field in the **last** basic or extended header is 59, which specifies that there is no extension header following it.

## IPv6 Address Format

- IPv6 provides approximately  $3.4 \times 10^{38}$  ( $2^{128}$ ) IPv6 addresses.
- IPv6 addresses are represented in **hexadecimal** format as compared to dotted-decimal in IPv4. **Note:** 32-bit IPv4 addresses are represented in 4 8-bit segments; each segment is written in decimal between 0 and 255 and separated with periods (dotted-decimal). 128-bit IPv6 addresses are represented in 8 16-bit segments; each segment is written in hexadecimal between 0x0000 and 0xFFFF and separated with colons.
- IPv6 addresses and prefixes often contain successive **hexadecimal** fields of 0s. There are 2 **zero compression** rules available for **shortening** the size of written IPv6 addresses and prefixes:
  - i) The leading 0s (and not trailing 0s) in any 16-bit segment can be omitted. If a segment has fewer than 4 hexadecimal digits, it is assumed that the missing digits are leading 0s. If the 16-bit segment contains all 0s, a 0 must be left there.
  - ii) Successive 0s can be represented with a **double colon (::)**; but this is allowed **only once**.  
 Ex: 2::/4 is an invalid abbreviation for 2000::/4, as it could represent 0x0002 or 0x2000; FE8::/10 is an invalid abbreviation for FE80::/10.  
 Ex: 2000:1111:0000:0000:0012:0000:0000:0001 can be written as 2000:1111:0:0:12::1 or 2000:1111::12:0:0:1.
- An IPv6 host can have multiple IPv6 addresses, and an IPv6 network can have multiple prefixes. As like IPv4 prefixes, an IPv6 prefix represents the **network** part of an address, as well as a range or block of consecutive IPv6 addresses.
- IPv4 addresses can be interpreted using either **classful addressing** or **classless address** rule. Classful addressing means that the interpretation of an IP address and subnet includes the idea of a classful network number, which is a separate network part of the IP address.



**Figure 3-3:** IPv4 Classful and Classless Addressing, and IPv6 Addressing

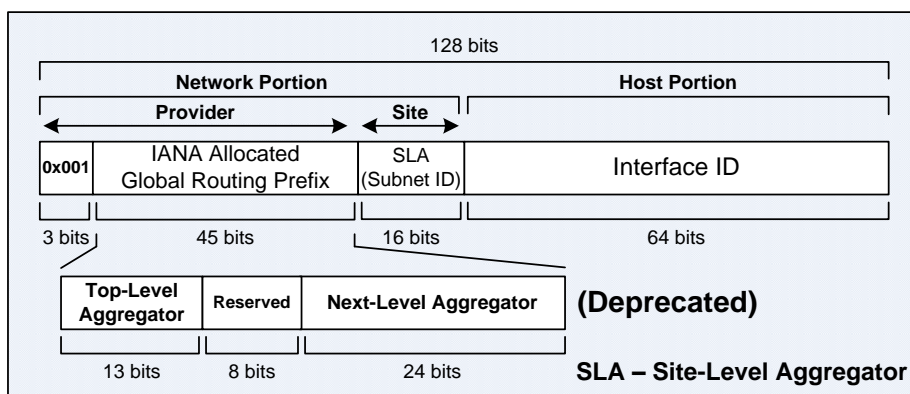
- With classful rule, 190.128.101.0/24 would be interpreted as 16 network bits (Class B address), 8 subnet bits, and 8 host bits. When the same network address is interpreted with classless rule, it means prefix 192.128.101.0 with prefix length of 24. Both rules have same subnet or prefix, same meaning, same router operation, and same configuration. It is just 2 different ways of interpreting the meaning of numbers.
- IPv6 uses a classless view of addressing, with no concept of classful addressing. Hence, it is no longer required to consider the classful boundaries of addresses, the default network bits or prefix lengths for different classes of addresses, etc for the operation of IPv6.

- Below lists the IPv6 address types:

<b>Unicast</b>	<b>One-to-one</b> mapping. A single source sends data to a single destination. A packet sent to a unicast address is delivered to the interface identified by the address. There are 3 main classes or types of IPv6 unicast addresses – <b>Global Unicast</b> , <b>Unique-Local Unicast</b> , and <b>Link-Local Unicast</b> .
<b>Multicast</b>	<b>One-to-many</b> mapping. A packet sent to a multicast address is delivered to all interfaces (usually belong to different nodes) identified by a <b>multicast group</b> . The members of a multicast group may include only a single device, or all devices in a network. Unlike IPv4, there is no broadcast address in IPv6. The all-nodes multicast address (FF02::1) serves as the same purpose as a broadcast address.
<b>Anycast</b>	<b>One-to-nearest</b> and <b>one to one-of-many</b> mappings. A packet sent to an anycast address is delivered to the <b>closest</b> , <b>nearest</b> , and <b>lowest-cost</b> interface (as determined by the routing protocol metric) identified by the address. An anycast address represents a <b>service</b> rather than of a device; and the same anycast address can reside on one or more devices providing the same service. Devices with the same characteristics are assigned with the same anycast address. Routers deliver client requests and localize / scope the traffic to the nearest device. Anycast address cannot be used as the source address of an IPv6 packet. Anycast addresses are defined by their service function rather than format, and hence it can be any IPv6 unicast address of any scope. <b>Note:</b> The scopes of IPv6 unicast address are global, site-local, and link-local.

### Aggregatable Global Unicast Addresses

- As like IPv4, IPv6 address and route aggregations reduce the size of routing tables and allow more **efficient**, **scalable**, and **manageable** Internet routing. It should be used whenever possible.



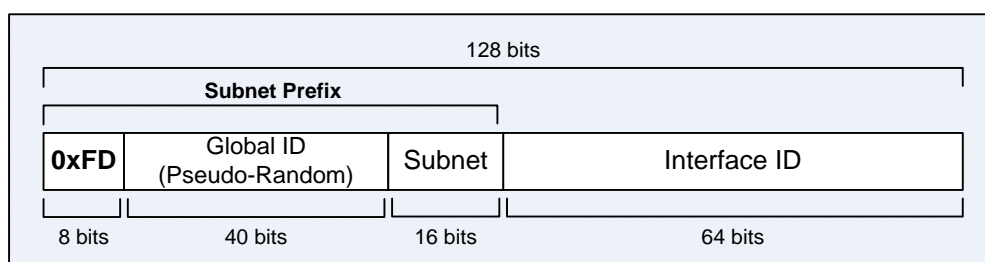
**Figure 3-4:** Aggregatable Global Unicast Address Format

- Figure 3-4 shows the format and bit allocation of an **Aggregatable Global Unicast Address**. This structure allows route summarization that reduces the number of routing entries in the global routing table. RFC 3587 – IPv6 Global Unicast Address Format specifies a new format which obsoletes and simplifies the old format which includes the **Top-Level Aggregator (TLA)** and **Next-Level Aggregator (NLA)**.
- **Global Routing Prefix** in an IPv6 address is globally unique and can be routed throughout the Internet; it serves the same purpose as **public IPv4 address**. The 1st 48 bits of the address is allocated by the IANA [1] for external routing within the Internet, with the **fixed prefix** of 001 in binary (2000::/3 – 2000::/4 or 3000::/4 in hexadecimal) to indicate a global IPv6 address. [1] **IANA** – Internet Assigned Numbers Authority ([www.iana.org](http://www.iana.org)).

- **Site Level Aggregator (SLA) or Subnet Identifier** is the address that is used by organizations to create local addressing hierarchy for routing and identifying the subnets within an AS. It can be used without the 48-bit prefix assigned by the IANA. If the global routing prefix is not used, the addressing scheme is similar to IPv4 private addressing, and the AS must not be connected to the Internet. This field allows the creation up to 65,536 ( $2^{16}$ ) subnets.
- Pay attention to the **subnetting** concept of IPv6. The SLA or Subnet ID is considered as a part of the network portion of an IPv6 address rather than the host portion as with IPv4! When performing subnetting in IPv4, the host portion of an IPv4 address shrinks and borrowed to create the subnet portion of an IPv4 address. The advantage of defining the IPv6 Subnet ID as a part of the network portion is that the size of the Interface ID can be **consistent** for all IPv6 addresses, which simplifies the parsing of IPv6 addresses. This also creates a **clear separation** in which the network portion provides the location of a device down to the specific data link segment while the host portion provides the identity of a device on a particular data link segment.
- The **Interface ID** is used to identify interfaces on a link (network) and it must be unique on a particular link. Interface IDs are used in IPv6 unicast addresses and often autoconfigured with the MAC address of an interface in the **Extended Unique Identifier-64 (EUI-64)** format.
- Below are some important rules when constructing an Interface ID in the EUI-64 format.
  - i) For IEEE 802 interface types (eg: Ethernet, FDDI), insert 0xFFFE between the upper 3 bytes OUI (24 bits) and the lower 3 bytes NIC serial number (24 bits) of a MAC address, and set the **Universal/Local (U/L)** bit (the 7th bit of the 1st octet) to binary 0 or 1. A value of 0 indicates a locally administered identifier, and a value of 1 indicates a globally unique IPv6 Interface ID. **Note:** By the way, the 7th bit of OUI is always 0. Ex: **MAC address** → 1111.1122.2222, **EUI-64** → 1311.11FF.FE22.2222.
  - ii) For other interface types (eg: serial, ATM, Frame Relay, loopback, and tunnel interfaces that are not being used with IPv6 overlay tunnels), the 1st MAC address of the router is used to construct the Interface ID with the same method above.
  - iii) For tunnel interface types that are used with IPv6 overlay tunnels, the Interface ID is construct with the **source IPv4 address for the tunnel** with all 0s in the first 32 bits. Ex: With 172.16.0.1 as the source IPv4 address for the tunnel, the link-local address for the tunnel interface is FE80::AC10:1.

## Local Unicast Addresses

The **IPv6 Unique-Local Unicast Address** serves the same purpose as **private IPv4 address** – 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16. It uses a prefix of FD00::/8 (1111111101). An IPv6 unique-local unicast address is globally unique but is intended for local communications – they are not expected to be routable throughout the Internet but rather routable within a site. The IPv6 Unique-Local Unicast address range uses 1/256 ( $2^8$ ) of the total IPv6 address space. **Note:** Kindly refer to Page 364 for the explanation of address space usage calculation.



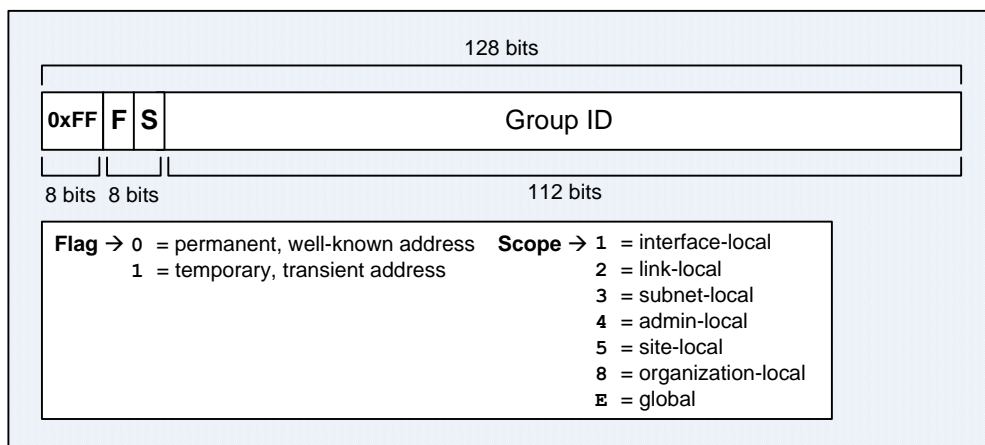
**Figure 3-5: IPv6 Unique-Local Unicast Address Format**



- The 40-bit Global ID is chosen in pseudorandom manner and hope that the addresses will be unique throughout the universe. Take note that pseudorandom numbers appear random but they are deterministic! The 16-bit Subnet field and 64-bit Interface ID work just like with global unicast addresses – identifying different subnets and hosts.
- **Note:** The IPv6 Site-Local Unicast Address which defined in original IPv6 RFCs has been **deprecated** and **replaced** with IPv6 Unique-Local Unicast Address as defined in RFC 4193 – Unique Local IPv6 Unicast Addresses!  
**Reference:** RFC 3879 – Deprecating Site-Local Addresses
- The **IPv6 Link-Local Unicast Address** is an IPv6 address that are automatically configured on an IPv6 interface with a prefix of FE80::/10 (1111111010) and the Interface ID in the EUI-64 format. Its scope is confined to a **single link** and hence is not routable off the link. Link-local addresses are often being used in the **neighbor discovery** and **stateless autoconfiguration** processes that communicate only on a single local link; this allow devices that reside on the same local link to create IPv6 addresses which allow them to communicate among each other without the need of a router, a global routing prefix, or a site-local address. The IPv6 Link-Local Unicast address range uses 1/1024 ( $2^{10}$ ) of the total IPv6 address space. **Note:** All IPv6 addresses begin with FE80, FE90, FEA0, and FEB0 are IPv6 link-local addresses. Kindly refer to the **IPv6 Autoconfiguration** section below for more information.
- The **IPv4-Compatible IPv6 Address** is used for IPv4-IPv6 coexistence and transition by tunneling IPv6 packets in IPv4 networks. It is a type of IPv6 unicast address that embeds an IPv4 address in the last 32 bits with 0s in the first 96 bits of an IPv6 address. The format of the address is 0:0:0:0:0:0:A.B.C.D/96 or ::A.B.C.D/96, with A.B.C.D as the IPv4 address in **hexadecimal**. Why /96? Because 32 out of 128 bits IPv6 addressing space are used to represent IPv4 nodes. Therefore a /96 prefix has enough address space to represent the entire IPv4 Internet. IPv4-compatible IPv6 addresses are assigned to **dual-stack nodes** that support both IPv4 and IPv6 protocol stacks, and are being used when implementing automatic tunnels. A dual-stack node configured with an IPv4-compatible address use the complete address as its IPv6 address, and use the embedded IPv4 address as its IPv4 address.  
Ex: 172.16.0.1 in IPv4 = 0:0:0:0:0:0:172.16.0.1/96 = ::172.16.0.1/96 = ::AC10:1/96 in IPv6.
- 6to4 tunneling using embedded IPv4 addresses called **unicast 6to4 addresses** (2002::/16) in which the IPv4 address is encoded in hexadecimal instead of dotted-decimal.  
Ex: 172.16.0.1 in hexadecimal is AC10:0001. A 6to4 prefix with 172.16.0.1 embedded would be 2002:AC10:1::/48.  
**Note:** The format of unicast 6to4 address is 2002:AABB:CCDD::/48, where AABB:CCDD is the **colon-hexadecimal** representation of A.B.C.D, an IPv4 address in dotted-decimal format.
- The **IPv6 All-zeroes Address** (::/0) is used as the default address when configuration default routes. Its prefix length is 0.
- The **IPv6 Unspecified Address** (::/128) is another all-zeroes IPv6 address used in the neighbor discovery process; when a node does not have an assigned unicast address and request an address via DHCP upon system startup; or when sending a duplicate address detection packet. The unspecified address is differentiated from a default address by its prefix length.
- The **IPv6 Loopback Address** (::1/128) is used to identify the local interface of the IP stack. It cannot be assigned to a physical interface. It can be used for basic IP stack troubleshooting.
- Both the IPv6 unspecified and loopback addresses cannot be assigned to physical interfaces.

## IPv6 Multicast Address

- Broadcast storms caused many problems in IPv4 networks, eg: high network response time. IPv6 does not use broadcasts; it relies solely on multicasts. IPv6 multicasts are being used in a different manner compared to IPv4 multicasts. IPv6 supports million groups of multicast addresses, and specific multicast group addresses are used for various functions.
- Multicasting is more efficient than broadcasting, which can interrupt and consume unnecessary processing time and resources on end system not intended for the data. Multicasts can be recognized and dropped at Layer 2; whereas broadcasts must be processed through the TCP/IP stack up to the network, transport, or application layer before an end system can determine whether the broadcast is intended for it.
- Multicasting is frequently being used in the IPv6 operation especially for some plug-and-play features, eg: router discovery and autoconfiguration.
- An IPv6 multicast address has a prefix of FF00::/8 (11111111). The 2nd byte identifies the **lifetime** (4 bits) and **scope** (4 bits) of a multicast group. The IPv6 Multicast address range uses 1/256 ( $2^8$ ) of the total IPv6 address space.
- A **permanent** and **temporary** multicast address have a lifetime value of 0 and 1 respectively.



**Figure 3-6: IPv6 Multicast Address Format**

- Below lists some reserved and well-known IPv6 multicast address in the reserved multicast address range (FF00:: to FF0F::):

Multicast Address	Multicast Group
FF01::1	All IPv6 nodes within the node-local scope
FF01::2	All IPv6 routers within the node-local scope
FF02::1	All IPv6 nodes within the link-local scope
FF02::2	All IPv6 routers within the link-local scope
FF02::5	All OSPFv3 routers within the link-local scope
FF02::6	All OSPFv3 designated routers within the link-local scope
FF02::9	All RIPng routers within the link-local scope
FF02::A	All EIGRP routers within the link-local scope
FF02::D	All PIM routers within the link-local scope
FF02::1:2	All DHCPv6 agents (servers and relays) within the link-local scope
FF05::2	All IPv6 routers within the site-local scope
FF02::1:FF00:0/104	IPv6 solicited-node multicast address within the link-local scope

- Since a multicast group always refers to a set of nodes, there is no sense for having a subnet field in the multicast address. Hence the last 112 bits are designated as the Group ID for identifying multicast groups. The current usage sets the first 80 bits to 0 and just uses the last 32 bits.
- An IPv6 node (host or router) is required to join the following multicast groups:
  - i) All-nodes multicast group FF02::1 (link-local scope).
  - ii) Solicited-Node multicast group (prefix FF02:0:0:0:0:1:FF00:0000/104).  
**Note:** 6 x 16 bits = 96 bits. 96 bits + 8 bits = 104 bits.
- Additionally, an IPv6 router must also join the all-routers multicast group FF02:0:0:0:0:0:0:2 (link-local scope).
- **IPv6 Solicited-Node Multicast Address** is used for generating Neighbor Solicitation messages (equivalent to IPv4 ARP Requests) for the **neighbor discovery** (the **address resolution**) process. The IPv4 ARP Requests are sent to the data link level broadcast, which introduce unnecessary processing for all nodes within the same broadcast domain. An IPv6 node must join the solicited-node multicast group for every IPv6 unicast and anycast address assigned to it. It has a prefix of FF02::1:FF00:0/104 with the last 24 bits being resolved from the last 24 bits of the corresponding IPv6 unicast or anycast address. Ex: The solicited-node multicast address for the IPv6 address FE80::1311:11FF:FE11:1111 is FF02::1:FF11:1111.  
Kindly refer to the **IPv6 Neighbor Discovery** section below for more information.
- An IPv6 host requires the following IPv6 addresses for proper operation:
  - i) Loopback address
  - ii) Link-local unicast address for every interface
  - iii) Assigned unicast address(es)
  - iv) All-node multicast address
  - v) Solicited-node multicast address for every unicast and anycast address assigned to it
  - vi) Multicast addresses of all other groups
  - vii) Unique-local unicast address (if applicable)
- An IPv6 router requires the following IPv6 addresses for proper operation:
  - i) All the required node addresses
  - ii) All-router multicast address
  - iii) Subnet-router anycast addresses for the configured forwarding interfaces
  - iv) Other assigned anycast addresses
  - v) Specific multicast addresses for routing protocols

### Identifying IPv6 Address Types

- The first few bits of an IPv6 address specify its address type. Below lists the IPv6 address types along with their allocated leading bit combinations.

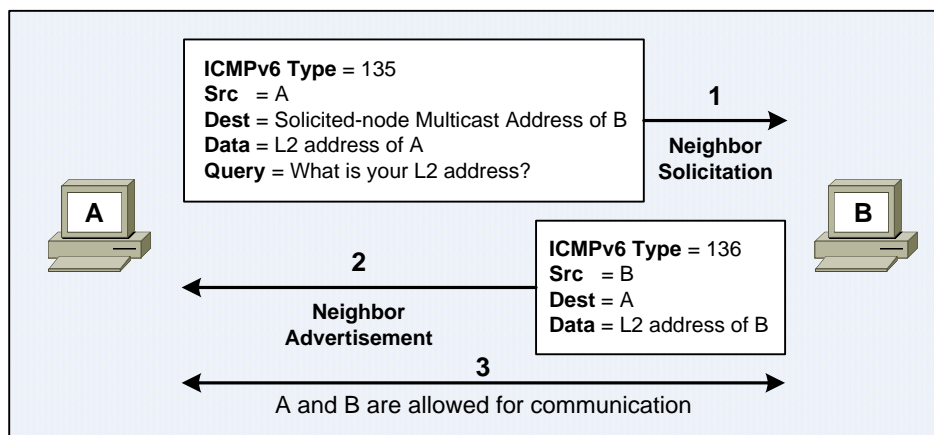
Address Type	High-order Bits (binary)	High-order Bits (hex)
Unspecified	00...0	:: or ::/128
Loopback	00...1	:::1 or :::1/128
Multicast	1111 1111	FF
Link-Local Unicast	1111 1110 10	FE8
Site-Local Unicast	1111 1110 11	FEC
Global Unicast	001	2 or 3

## IPv6 Neighbor Discovery Protocol

- The main characteristic of IPv6 besides its increased address space is its plug-and-play features. The **Neighbor Discovery Protocol (NDP)** provides the following functions and plug-and-play features for IPv6 hosts and routers when they are connected to an IPv6 link:

<b>Router Discovery</b>	A node it can discover the local routers without using DHCP.
<b>Prefix Discovery</b>	A node can discover the prefix(es) assigned to the link.
<b>Parameter Discovery</b>	A node can discover parameters (eg: link MTU, hop limits) for the link.
<b>Address Autoconfiguration</b>	A node can determine its full address without using DHCP.
<b>Next-Hop Determination</b>	A node can determine the link-layer next hop for a destination, either as a local destination or a router to the destination.
<b>Neighbor Unreachable Detection</b>	A node can determine when a neighbor (host or router) on the link is no longer reachable.
<b>Duplicate Address Detection</b>	A node can determine if an address it would like to use is already being used by another node (host or router) on the link.
<b>Redirect</b>	A router can notify a host for a better next-hop other than itself to a destination on another link. The redirect function is part of ICMPv4 functionality but is redefined as part of NDP in IPv6.

- The scope of NDP messages is **link-local**; hence the IPv6 packets encapsulating them are always IPv6 link-local unicast address or multicast address with a link-local scope. The Hop Limit of the IPv6 packets encapsulating NDP messages is **255**. If a packet is received with a Hop Limit less than 255, it means that the packet has passed through at least 1 router. The packet is dropped for preventing NDP from being attacked or spoofed from a source not connected to the local link.
- IGMP is used in IPv4 to allow a host to inform its local router that it was joining a multicast group and would like to receive traffic for the particular multicast group. This function has been replaced by the ICMPv6 **Multicast Listener Discovery** process.
- ICMPv6 messages and IPv6 Solicited-Node Multicast addresses are used to perform the above mentioned tasks. Hence an IPv6 node (host or router) must join the solicited-node multicast group for every unicast and anycast address assigned to it.



**Figure 3-7: IPv6 Neighbor Discovery Process**

- The neighbor discovery process utilizes **neighbor solicitation** and **neighbor advertisement** messages. **Neighbor solicitation** messages are being sent to the local link when a node would like to determine the data link layer address of another node on the same local link. A neighbor solicitation message is sent from the source node destined to the solicited-node multicast group address with the last 24 bits of the IPv6 unicast address of the destination node. The destination node will then respond with its data link layer address using a **neighbor advertisement** message. This operation is similar to **ARP resolution** in IPv4, but without the use of broadcast messages. **Note:** The source node must identify the IPv6 unicast address of the destination node prior to sending a neighbor solicitation message using a naming service mechanism (eg: DNSv6).
- The IPv6 neighbor solicitation and IPv6 neighbor advertisement messages have a value of 135 and 136 respectively in the Type field of the ICMPv6 header.
- When a node changes its data link layer address, it can send an unsolicited neighbor advertisement message to advertise the new address.
- IPv6 **router discovery** allows IPv6 nodes to discover the routers on the local link. It is similar to **ICMP Router Discovery Protocol (IRDP)** in IPv4.
- The router discovery process utilizes **router solicitation** and **router advertisement** messages. **Router solicitation** messages allow a node without an assigned unicast address to autoconfigure itself without waiting for the next scheduled router advertisement message from an IPv6 router. Router solicitation messages are only sent upon boot time and 3 times afterward to avoid flooding of router solicitation messages in the absence of a router on the network.
- An IPv6 router solicitation message has a value of 133 in the Type field of the ICMPv6 header. Normally the IPv6 unspecified address (0::0) is used as the source address, and the all-routers link-local multicast address (FF02::2) is used as the destination address.
- **Router advertisement** messages are periodically sent out from all interfaces of an IPv6 router (destined to the unsolicited all-nodes link-local multicast address – FF02::1). They are also being sent out as responses to router solicitation messages from IPv6 nodes on the local link (destined to the IPv6 unicast address of the node that sent out the router solicitation message).
- An IPv6 router advertisement message has a value of 134 in the Type field of the ICMPv6 header and contains the following information:
  - i) Whether nodes can use address autoconfiguration.
  - ii) Flags to indicate the type of autoconfiguration – stateless or stateful.
  - iii) One or more IPv6 prefixes that local link nodes could use for autoconfiguration.
  - iv) Lifetime information for each prefix.
  - v) Whether the router should be used as a default router. If yes, includes the amount of time.
  - vi) Additional information, eg: link prefix(es), hop limit, and link MTU a node should use.
- **Renumbering** of IPv4 networks and nodes will at least take months if not years. However, renumbering of IPv6 nodes is possible with the help of router advertisements. Router advertisement messages can contain both the old and new prefixes, with a lifetime value for the old prefix to tell the nodes to begin to use the new prefix, while still maintaining their current connections with the old prefix. During this period, nodes have 2 unicast addresses. When the old prefix is retired, the router advertisements will only advertise the new prefix.
- Renumbering networks also requires the renumbering of all routers and changes of DNS entries. A router renumbering protocol has been proposed and is currently under review.

## DHCPv6

- IPv6 supports 2 methods for dynamic configuration of IPv6 addresses, prefix lengths, and default routers, namely **stateful DHCPv6** and **stateless configuration**.
- DHCPv6 works similar to DHCPv4 – DHCPv6 client first sends a multicast DHCPv6 discovery message to search for a DHCPv6 server; a DHCPv6 server replies with unicast DHCPv6 offer message along with an IPv6 address, prefix length, default router, and DNS server IP addresses; the DHCPv6 client sends a multicast DHCPv6 request message to obtain a lease of an IP address; finally the DHCPv6 server replies with a unicast DHCPv6 acknowledge message to the DHCPv6 client and the DHCPv6 client may start to use the leased IPv6 address.
- There are 2 operational modes of DHCPv6 servers – **stateful** and **stateless**. Stateful operation track state information (eg: the IP address leased to a client and the valid period for the lease); while stateless operation do not track state information. It acts as the same role as DHCPv4. However, note that stateful DHCPv6 does not supply the default router information but instead rely upon the Neighbor Discovery Protocol between the client and the local routers.
- **Stateless DHCPv6** often work in conjunction with stateless autoconfiguration to provide information such as domain name, as well as the IP addresses of DNS and NTP servers, which stateless autoconfiguration is unable to provide.  
**Note:** Cisco IOS only provides support for stateless DHCPv6, which means it does not offer any address assignment and management of the DHCP pool.
- The multicast address FF02::1:2 (**All DHCPv6 Relay Agents within the link-local scope**) is reserved for hosts to send packets to a DHCPv6 server. Routers would forward these packets to the appropriate DHCPv6 servers.  
**Note:** The multicast addresses FF05::1:3 and FF05::1:4 are **All DHCPv6 Servers within the site-local scope** and **All DHCPv6 Relay Agents within the site-local scope** respectively.

## IPv6 Autoconfiguration

- IPv6 supports autoconfiguration of globally unique addresses. With **stateless autoconfiguration**, a router sends periodical router advertisement messages to all nodes on the local link for them to autoconfigure their IPv6 addresses. An IPv6 host uses the 48-bit global routing prefix and the 16-bit SLA advertised by the router as the first 64 bits for its address, and its 48-bit MAC address in **Extended Universal Identifier 64-bit** (EUI-64) format as the last 64 bits for its address.
- Autoconfiguration provides the **plug-and-play** feature which allows devices to connect to the Internet without any IP address configuration or DHCP server. Plug-and-play is the key feature to provide Internet connectivity for devices such as cordless phones and even bread toasters.
- The **stateless autoconfiguration** process is initiated when a host sends a router solicitation message upon system startup to request for an **immediate** transmission of a router advertisement message, which contains the autoconfiguration information from an IPv6 router on the local link, without waiting for the next scheduled router advertisement message from an IPv6 router.
- An IPv6 router must be configured with router advertisement retransmission timer and other parameters (eg: router lifetime, reachable time) for the operation of stateless autoconfiguration.

- Comparison of Stateful and Stateless DHCPv6, as well as Stateless Autoconfiguration:

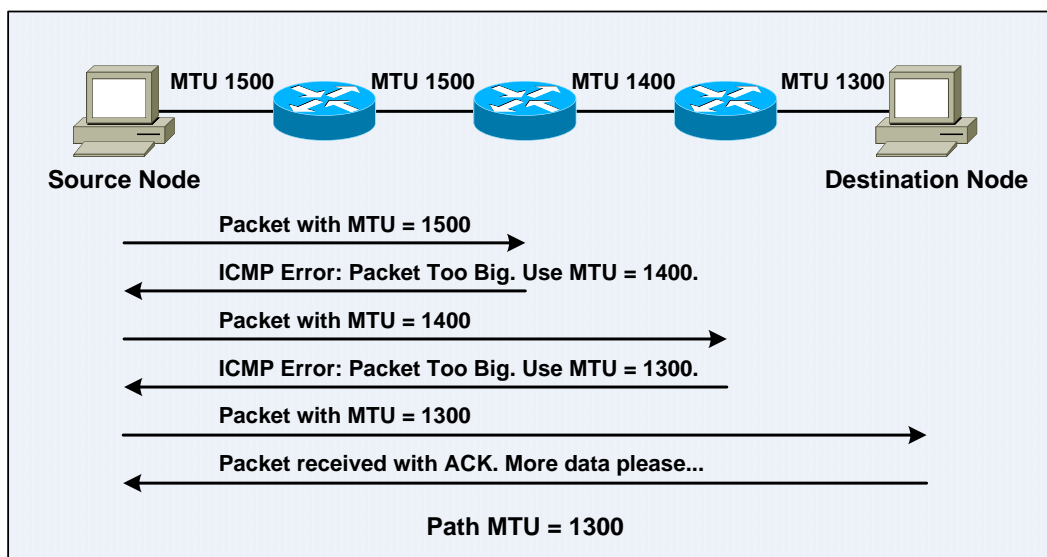
Feature	Stateful DHCPv6	Stateless DHCPv6	Stateless Autoconfiguration
Dynamically assign IPv6 address to client	Yes	No	Yes
Supplies useful information, eg: domain name, DNS and NTP servers IP addresses, etc.	Yes	Yes	No

**Note:** Stateless autoconfiguration often works in conjunction with Stateless DHCPv6.

- IPv6 provides a mechanism to detect duplicate addresses and prevent address collision, although the use of MAC address to derive the Interface ID and eventually the IPv6 address almost guarantees a unique address. Neighbor solicitation messages are used to detect duplicate address on the link. Duplicate address detection occurs during the autoconfiguration process.

## Path MTU Discovery

- IPv6 routers along a path no longer perform fragmentation as in IPv4. Fragmentation is performed at the source IPv6 node when necessary.
- The main purpose of PMTUD discovery process is determining the most optimum (maximum) MTU for a path to eliminate the need of fragmentation. PMTUD allows IPv4 and IPv6 nodes to dynamically discover and adjust to differences in the MTU of the links along a path accordingly.
- An ICMPv6 Type 2 – Packet Too Big error message will be sent by a router when it cannot forward a packet that is larger than the MTU of the outgoing link to the destination.
- When an IPv6 node attempts to send a packet at the size specified by the upper layers and receives an ICMP Packet Too Big error message (which would contain a recommended MTU), it tells the upper layer to discard the packet and use the new MTU. Each device needs to track the MTU size for each session. The tracking of the MTU size can be built by creating a **cache** based on destination address, flow label, or source address (if source routing is being performed).
- A host ages cached MTU values and performs PMTUD every 10 minutes to see if the MTU has increased along the path, as there can be more appropriate MTU when routing paths change.



**Figure 3-8:** Path MTU Discovery

## Stream Control Transmission Protocol

- IPv6 uses **Stream Control Transmission Protocol (SCTP)** as the transport layer protocol. SCTP provides **reliable transport service**, as well as **sequencing** and **acknowledging** functions as provided by TCP.
- SCTP was built to overcome the limitations of TCP which requires a strict order of transmission that can cause **head-of-line blocking** and eventually delay due to the reassembled of out-of-order segments and retransmission of loss segments. TCP sends a stream of bytes, whereas SCTP sends several **independent streams** of messages that are sequenced and delivered independently. SCTP uses a selective acknowledgement (SACK) mechanism to recover error SCTP segments.
- Another main benefit of SCTP is the support for **multi-homing**, which can provide transparent failover upon network failures. Multi-homed nodes have multiple NICs and can be reached via several IP addresses as well as a variety of paths. During SCTP setup, a multi-homed client informs the server about all its addresses in the INIT chunk. The client needs to know only a single address for the server (the server provides all its addresses to the client in the INIT-ACK). SCTP monitors all paths between the hosts with a heartbeat function and identifies one path as the primary. Secondary paths are used for retransmission or when the primary path fails.
- SCTP also provides greater security than TCP by using a cookie function for each session. Below describes the steps when Host1 establishes an SCTP session with Host2:
  - i) Host1 sends an initialization request to Host2. Host1 waits for a message from Host2.
  - ii) Host2 receives the request. Host2 generates an encrypted key and a message authentication code (indicates the creator of the message – Host2), and includes these information into a cookie message. Host2 sends the cookie message to Host1.
  - iii) Host1 receives the cookie message. Host1 replies to Host2 with a cookie echo message. Host1 waits for a message from Host2.
  - iv) Host2 receives the cookie echo message. Host2 examines the message to ensure that the message authentication code indicates Host2 was the creator of the cookie. Host2 sends a cookie acknowledgment to Host1. Host2 initiates the SCTP session. Host2 is now ready to accept and send data.
  - v) Host1 receives the cookie acknowledgment. Host1 is now ready to accept and send data.
- **Note:** Even though SCTP looks promising, it is rarely used in real-life environments due to:
  - It was designed by the wrong IETF working group – SIGTRAN, which was focused on transport of PSTN signaling over IP networks.
  - It was not properly promoted. The SIGTRAN working group solved their problems and moved on.
  - Limited support for it in the networking equipments, eg: access lists, stateful inspection, etc.
  - It is not shipped with modern operating systems by default, which is a major stopper for widespread deployment. Some installation and integration works are needed to support it.
  - The biggest stopper to SCTP adoption is the lack of Session layer in TCP/IP and the legacy Berkeley Sockets API. The SCTP protocol must be specified in the socket() function call in order to use SCTP with the Berkeley Sockets API, which means that every application that would like to benefit from SCTP support must be changed, recompiled, and tested. It is impossible to simply add SCTP support into the operating systems to provide better performance for existing applications.



## IPv4 and IPv6 Interoperability / Coexistence / Integration and Transition

- Until IPv4 completely transitioned to IPv6, IPv6 hosts must be able to communicate with IPv4 hosts and through IPv4 networks. IPv6 transition mechanisms allow IPv6 hosts to reach IPv4 services and isolated IPv6 hosts and networks to reach the IPv6 Internet over IPv4 networks.
- IPv4 to IPv6 transition is a slow process, as it requires planning and implementation of new addressing, protocol stacks, and applications. Generally, the deployment of IPv6 should start from the network edges, and move towards the network core.
- There are many transition mechanisms available to smooth the IPv4 to IPv6 transition. The most common IPv6 transition techniques are **dual stacking** and **tunneling**. The most common type of tunneling is IPv6 to IPv4 (6to4) tunneling, which encapsulates IPv6 packets into IPv4 packets. Another transition technique known as **protocol proxying and translation** uses an extension of IP NAT – **NAT Protocol Translator (NAT-PT)** to translate between IPv4 and IPv6 addresses.
- The dual-stack transition mechanism is a network interface that is configured with an IPv4 address and an IPv6 address. A node implementing a dual stack is called a **dual-stack node**.
- A dual-stack router runs both IPv4 and IPv6 stacks, and can communicate with both IPv4 and IPv6 devices. A dual-stack interface can forwards both IPv4 and IPv6 traffic. The **ipv6 unicast-routing** global configuration command enables the forwarding of IPv6 packets between interfaces (similar to the **ip routing** command which enables the forwarding of IPv4 packets). The **ipv6 address ipv6-addr/prefix-length** interface subcommand assigns an IPv6 address and enables IPv6 processing for an interface.
- IPv6 tunneling is the mechanism where encapsulating IPv6 packets within IPv4 packets to allow an isolated network or host to reach the IPv6 Internet.
- Tunnels are often used to transport an incompatible protocol across an existing network. Tunneling IPv6 traffic over an IPv4 network requires edge routers at each end of the tunnel for encapsulating and decapsulating the packets. Figure 3-9 shows the interconnection of IPv6 networks without migrating the entire network to IPv6.

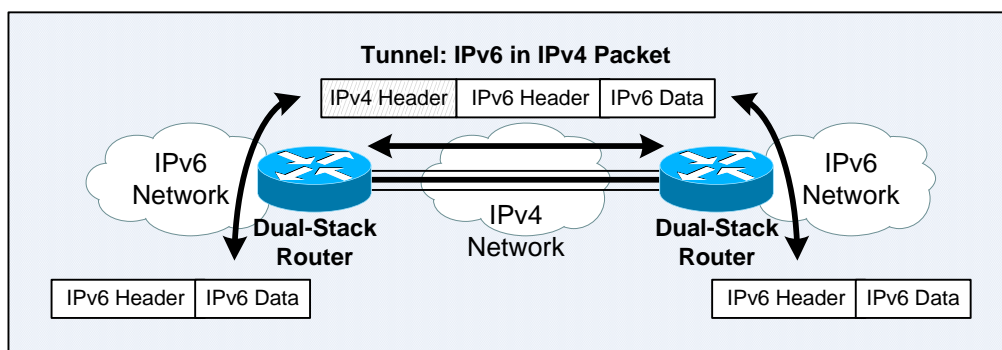
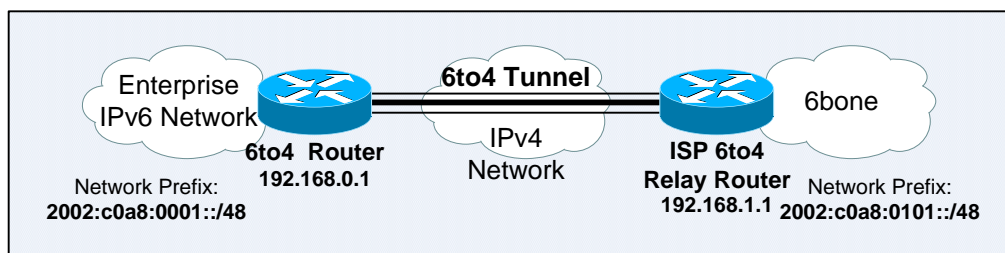


Figure 3-9: Overlay Tunneling

- **Note:** A tunneled network is often difficult to troubleshoot and thus should be considered as a transition technique (temporary) that should be used only where it is appropriate. Using native IPv6 throughout the network is still the final goal.
- There are 2 types of tunnels – **static** (manually configured) and **automatic** (6to4 tunneling).

- In a manually configured tunnel, the source and destination IPv4 addresses for the tunnel as well as IPv6 addresses are statically configured on the dual-stack routers at each end of the tunnel. The configuration does not change upon network and routing needs change. Routing must be configured properly to forward packets between the IPv6 networks. Typically between routers.
- In 6to4 tunneling, the connection of IPv6 networks through an IPv4 network is **dynamically established**. The IPv4 address of the tunnel endpoints can be dynamically discovered based on the destination IPv6 addresses. Typically between routers.
- A 6to4 tunnel treats the IPv4 network as a virtual link. Each 6to4 edge router has an /48 prefix IPv6 address, which is the concatenation of 2002::/16 and the IPv4 address of the edge router (32-bit in hexadecimal format). 2002::/16 is the assigned address range for 6to4 tunneling. The edge routers automatically build the tunnel using their IPv4 addresses.  
Ex: The IPv6 network prefix for an edge router with an IPv4 address of 192.168.0.1 is 2002:c0a8:0001::/48 (c0a80001 is the hexadecimal representation of 192.168.0.1).
- When an edge router receives an IPv6 packet with a destination address in the range of 2002::/16, it determines from its routing table that the packet must go through a tunnel. The router extracts the IPv4 address of the 6to4 router at the other end of the tunnel from the 3rd to 6th octets in the destination IPv6 address. The router would encapsulate the IPv6 packets in IPv4 packets destined to the extracted IPv4 address and forward them out to the IPv4 network. The destination edge router decapsulates the IPv6 packets from the received IPv4 packets and forwards the IPv6 packets to the final destination.
- Figure 3-10 shows a scenario of 6to4 tunneling – an enterprise with an IPv4 network connects to the 6bone in order to assess the connectivity impact and expand its knowledge of IPv6 before merges with another company that runs IPv6 on its network. A **6to4 relay router** is required to be able to reach a native IPv6 Internet. It offers traffic forwarding to the IPv6 Internet.



**Figure 3-10: 6to4 Tunneling to the 6bone**

- The other 2 available tunneling methods are **Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)**, typically between routers; and **Teredo tunneling**, typically between hosts.
- When an IPv6 host would like to communicate with an IPv4 web server, the proxying and translation mechanism is best suit. The easiest solution is setup a web proxy that can translate the IPv6 host address to an IPv4 address for communication with the IPv4 web server.
- **NAT Protocol Translator (NAT-PT)** is a device that can translate between IPv4 and IPv6 addresses for the communication between IPv4 and IPv6 hosts.  
**Note:** NAT-PT as defined in RFC 2766 is obsolete and deprecated to historic status on July 2007 due to numerous issues. **Transport Relay Translation (TRT)** as defined in RFC 3142 is the most common form of NAT-PT. The NAT-PT (and TRT) translation mechanism typically used in conjunction with a **DNS Application-Level Gateway (DNS-ALG)** which performs translation between AAAA and A records.

## IPv6 Routing Protocols

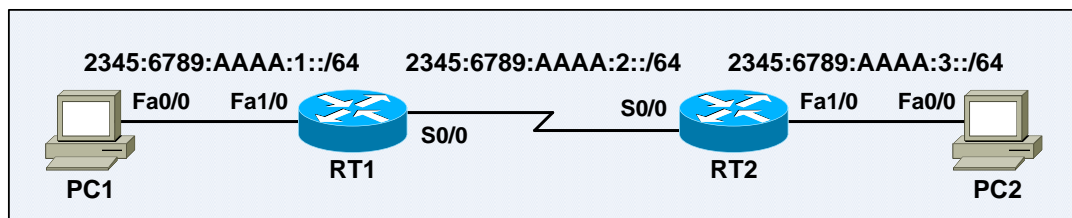
- All current interior and exterior gateway routing protocols have been updated to support IPv6. The IPv6 routing protocols still retain most of the internal features from their IPv4 predecessors. Below lists the IPv6 routing protocols along with some basic information:

Interior Gateway Protocols (IGPs)	
<b>RIPng</b> (RIP Next Generation)	Similar to RIPv2. Still has a limit of 15 hops and uses split horizon and poison reverse to prevent routing loops. Uses the all-RIP-routers multicast group FF02::9 for sending updates to all RIPng routers. Updates are sent on UDP port 521. Advertises routes every 30 seconds.
<b>EIGRPv6</b>	Similar to EIGRP. It includes a new <b>Protocol-Dependent Module (PDM)</b> for IPv6. EIGRPv6 hello packets and updates are sent using the all-EIGRP-routers multicast group FF02::A.
<b>OSPFv3</b>	Similar to OSPFv2. OSPFv3 runs directly over IPv6. Advertises routes using multicast groups FF02::5 (all-OSPFv3-routers) and FF02::6 (all-OSPFv3-designated-routers); uses IPv6 link-local unicast addresses as the source addresses for Hello and LSU packets. OSPFv3 does not provide authentication as IPv6 authentication is handled through IPsec.
<b>Integrated IS-ISv6</b>	Similar to Integrated IS-ISv4, with some extensions added, including a new Protocol Identifier and 2 new TLV ( <b>Type, Length, Value</b> ) tuples for IPv6 reachability and IPv6 interface address.
Exterior Gateway Protocol (EGP)	
<b>BGP4+ / MBGP / MP-BGP4</b>	The <b>multiprotocol extensions</b> for BGP4 allow other protocols other than IPv4 to be routed, including IPv6. BGP4+ also defines other IPv6-specific extensions, eg: a new identifier for the IPv6 address family.

**Note:** The multicast addresses for IPv6 IGPs are similar to the multicast addresses for their IPv4 predecessors, eg: RIPng FF02::9 – RIPv2 224.0.0.9; EIGRPv6 FF02::A – EIGRP 224.0.0.10; All-OSPFv3-Routers FF02::5 – All-OSPF-Routers 224.0.0.5; All-OSPFv3-Designated-Routers FF02::6 – All-OSPF-Designated-Routers 224.0.0.6.

- IPv6 routing protocols are configured and enabled directly on router interfaces from the interface configuration mode and no longer use the **network** router subcommand.

## IPv6 Configuration



**Figure 3-11:** Sample IPv6 Network

- The 3 ways for assigning an IPv6 address to a node are manual configuration (static), stateless autoconfiguration (dynamic), and stateful DHCPv6 (dynamic).
- The **ipv6 unicast-routing** global configuration command globally enables the forwarding of IPv6 packets (**IPv6 routing**) for interfaces configured with an IPv6 address. This command also enables Neighbor Discovery Protocol for LAN interface types, eg: Ethernet. RT1 and RT2 must be configured with this command in order to forward IPv6 packets between physical interfaces.

- The **ipv6 enable** interface subcommand configures an IPv6 link-local address and enables IPv6 processing for an interface and on the interface. The link-local address can be used only for communication with nodes reside on the same link or network.  
**Note:** This command does not enable the Neighbor Discovery Protocol.
- The **ipv6 address** *{ipv6-addr/prefix-length [eui-64] | {ipv6-addr link-local}* interface subcommand configures a global IPv6 address on an interface and enables IPv6 processing on the interface. The **eui-64** keyword configures an IPv6 address with the last 64 bits of the IPv6 address in EUI-64 format. The **link-local** keyword configures a specific link-local IPv6 address on an interface instead of using the automatically generated Interface ID in EUI-64 format.  
**Note:** This command does not enable the Neighbor Discovery Protocol.
- Basic IPv6 addressing configuration on RT1:

```

RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#ipv6 unicast-routing
RT1(config)#int fa1/0
RT1(config-if)#ipv6 address 2345:6789:AAAA:1::/64 eui-64
RT1(config-if)#no shut
RT1(config-if)#exit
RT1(config)#int s0/0
RT1(config-if)#ipv6 address 2345:6789:AAAA:2::1/64
RT1(config-if)#no shut
RT1(config-if)#^Z
RT1#
RT1#sh int fa1/0 | in bia
  Hardware is AmdFE, address is 0004.4e11.1111 (bia 0004.4e11.1111)
RT1#
RT1#sh ipv6 int | in is up|link-local
Serial0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::204:4EFF:FE11:1111
FastEthernet1/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::204:4EFF:FE11:1111
RT1#
RT1#sh ipv6 int fa1/0
FastEthernet1/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::204:4EFF:FE11:1111
  Global unicast address(es):
    2345:6789:AAAA:1:204:4EFF:FE11:1111, subnet is 2345:6789:AAAA:1::/64
  Joined group address(es):
    FF02::1      ! All IPv6 nodes within the link-local scope
    FF02::2      ! All IPv6 routers within the link-local scope
    FF02::1:FF11:1111 ! Solicited-node multicast address (link-local scope)
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds
  ND advertised reachable time is 0 milliseconds
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements are sent every 200 seconds
  ND router advertisements live for 1800 seconds
  Hosts use stateless autoconfig for addresses.
RT1#

```

```

RT1#sh ipv6 int s0/0
Serial0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::204:4EFF:FE11:1111
  Global unicast address(es):
    2345:6789:AAAA:2::1, subnet is 2345:6789:AAAA:2::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF00:1
    FF02::1:FF11:1111
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds
  Hosts use stateless autoconfig for addresses.
RT1#
RT1#ping 2345:6789:AAAA:2::1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2345:6789:AAAA:2::1, timeout is 2 seconds:
!!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
RT1#
RT1#ping 2345:6789:AAAA:1:204:4EFF:FE11:1111  ! OMG!!!

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2345:6789:AAAA:1:204:4EFF:FE11:1111,
timeout is 2 seconds:
!!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
RT1#

```

- Basic IPv6 addressing configuration on RT2:

```

RT2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT2(config)#ipv6 unicast-routing
RT2(config)#int s0/0
RT2(config-if)#ipv6 address 2345:6789:AAAA:2::2/64
RT2(config-if)#no shut
RT2(config-if)#exit
RT2(config)#int fa1/0
RT2(config-if)#ipv6 address 2345:6789:AAAA:3::1/64
RT2(config-if)#no shut
RT2(config-if)#^Z
RT2#
RT2#sh ipv6 int brief s0/0
Serial0/0          [up/up]
    FE80::204:4EFF:FE22:2211
    2345:6789:AAAA:2::2
RT2#
RT2#sh ipv6 int brief fa1/0
FastEthernet1/0   [up/up]
    FE80::204:4EFF:FE22:2211
    2345:6789:AAAA:3::1
RT2#

```

- Basic IPv6 addressing configuration on PC1 and PC2:

```

PC1 (config) #int fa0/0
PC1 (config-if) #ipv6 address autoconfig
PC1 (config-if) #no shut
PC1 (config-if) #^Z
PC1#
PC1#sh ipv6 int brief
FastEthernet0/0          [up/up]
    FE80::204:4EFF:FE33:3301
    2345:6789:AAAA:1:204:4EFF:FE33:3301
PC1#
PC1#sh ipv6 neighbors
IPv6 Address                Age Link-layer Addr State
Interface
FE80::204:4EFF:FE11:1111    3 0004.4e11.1111  STALE Fa0/0
2345:6789:AAAA:1:204:4EFF:FE11:1111  3 0004.4e11.1111  STALE Fa0/0

PC1#
PC1#sh ipv6 routers
Router FE80::204:4EFF:FE11:1111 on FastEthernet0/0, last update 1 min
  Hops 64, Lifetime 1800 sec, AddrFlag=0, OtherFlag=0, MTU=1500
  Reachable time 0 msec, Retransmit time 0 msec
  Prefix 2345:6789:AAAA:1::/64 onlink autoconfig
  Valid lifetime 2592000, preferred lifetime 604800
PC1#
=====
PC2 (config) #int fa0/0
PC2 (config-if) #ipv6 address autoconfig
PC2 (config-if) #ipv6 address FE80::2222 link-local
PC2 (config-if) #no shut
PC2 (config-if) #^Z
PC2#
PC2#sh ipv6 int brief
FastEthernet0/0          [up/up]
    FE80::2222
    2345:6789:AAAA:3::2222
PC2#
PC2#sh ipv6 neighbors
IPv6 Address                Age Link-layer Addr State
Interface
FE80::204:4EFF:FE22:2211    0 0004.4e22.2211  STALE Fa0/0
2345:6789:AAAA:3::1         4 0004.4e22.2211  STALE Fa0/0

PC2#
PC2#sh ipv6 routers
Router FE80::204:4EFF:FE22:2211 on FastEthernet0/0, last update 1 min
  Hops 64, Lifetime 1800 sec, AddrFlag=0, OtherFlag=0, MTU=1500
  Reachable time 0 msec, Retransmit time 0 msec
  Prefix 2345:6789:AAAA:3::/64 onlink autoconfig
  Valid lifetime 2592000, preferred lifetime 604800
PC2#

```

- Network engineers often use easier-to-remember values like ::1 instead of the automatically generated Interface ID in EUI-64 format when assigning link-local and global unicast addresses.

## IPv6 Static Routing and Default Routing Configuration

- Static Routing configuration on RT1 and Default Routing configuration on RT2:

```
RT1(config)#ipv6 route 2345:6789:AAAA:3::/64 2345:6789:AAAA:2::2
RT1(config)#^Z
RT1#
RT1#sh ipv6 route
IPv6 Routing Table - 7 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
C   2345:6789:AAAA:1::/64 [0/0]
    via ::, FastEthernet1/0
L   2345:6789:AAAA:1:204:4EFF:FE11:1111/128 [0/0]
    via ::, FastEthernet1/0
C   2345:6789:AAAA:2::/64 [0/0]
    via ::, Serial0/0
L   2345:6789:AAAA:2::1/128 [0/0]
    via ::, Serial0/0
S   2345:6789:AAAA:3::/64 [1/0]
    via 2345:6789:AAAA:2::2
L   FE80::/10 [0/0]
    via ::, Null0
L   FF00::/8 [0/0]
    via ::, Null0
RT1#
=====
RT2(config)#ipv6 route ::/0 s0/0
RT2(config)#^Z
RT2#
RT2#sh ipv6 route
IPv6 Routing Table - 7 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
S   ::/0 [1/0]
    via ::, Serial0/0
C   2345:6789:AAAA:2::/64 [0/0]
    via ::, Serial0/0
L   2345:6789:AAAA:2::2/128 [0/0]
    via ::, Serial0/0
C   2345:6789:AAAA:3::/64 [0/0]
    via ::, FastEthernet1/0
L   2345:6789:AAAA:3::1/128 [0/0]
    via ::, FastEthernet1/0
L   FE80::/10 [0/0]
    via ::, Null0
L   FF00::/8 [0/0]
    via ::, Null0
RT2#
```

- The link-local address of a neighbor can be specified as the next-hop address for a static route. If using a link-local address as the next-hop address, then both the outgoing interface and link-local address and must be specified in the static route configuration.

- Verify that PC1 is able to reach PC2:

```
PC1#ping 2345:6789:AAAA:3::2222
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 2345:6789:AAAA:3::2222, timeout is 2 seconds:
```

```
!!!!
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max = 92/107/128 ms
```

```
PC1#
```

## RIPng Configuration

- RIPng configuration on RT1 and RT2:

```
RT1(config)#no ipv6 route 2345:6789:AAAA:3::/64
```

```
RT1(config)#int s0/0
```

```
RT1(config-if)#ipv6 rip ?
```

```
WORD User selected string identifying this RIP process
```

```
RT1(config-if)#ipv6 rip 1 enable
```

```
RT1(config-if)#int fa1/0
```

```
RT1(config-if)#ipv6 rip 1 enable
```

```
RT1(config-if)#exit
```

```
RT1(config)#ipv6 router rip ?
```

```
WORD User selected string identifying this process
```

```
RT1(config)#ipv6 router rip 1
```

```
RT1(config-rtr)#^Z
```

```
RT1#
```

```
RT1#sh ipv6 protocols
```

```
IPv6 Routing Protocol is "connected"
```

```
IPv6 Routing Protocol is "static"
```

```
IPv6 Routing Protocol is "rip 1"
```

```
Interfaces:
```

```
FastEthernet1/0
```

```
Serial0/0
```

```
Redistribution:
```

```
None
```

```
RT1#
```

```
=====
```

```
RT2(config)#no ipv6 route ::/0
```

```
RT2(config)#int s0/0
```

```
RT2(config-if)#ipv6 rip 1 enable
```

```
RT2(config-if)#int fa1/0
```

```
RT2(config-if)#ipv6 rip 1 enable
```

```
RT2(config-if)#^Z
```

```
RT2#
```

```
RT2#sh run | in ipv6 router rip
```

```
ipv6 router rip 1
```

```
RT2#
```

**Note:** The **ipv6 rip {rip-proc-name} enable** interface subcommand will start a RIPng process with the defined tag. The **ipv6 router rip {rip-proc-name}** global configuration which enters the router configuration mode is optional and is not required to enable a RIPng process. The tag name is local significant and does not have to match between RIPng routers.



- Verify the RIPng operation on RT1:

```

RT1#sh ipv6 route
IPv6 Routing Table - 7 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
C   2345:6789:AAAA:1::/64 [0/0]
    via ::, FastEthernet1/0
L   2345:6789:AAAA:1:204:4EFF:FE11:1111/128 [0/0]
    via ::, FastEthernet1/0
C   2345:6789:AAAA:2::/64 [0/0]
    via ::, Serial0/0
L   2345:6789:AAAA:2::1/128 [0/0]
    via ::, Serial0/0
R   2345:6789:AAAA:3::/64 [120/2]
    via FE80::204:4EFF:FE22:2211, Serial0/0
L   FE80::/10 [0/0]
    via ::, Null0
L   FF00::/8 [0/0]
    via ::, Null0
RT1#
RT1#sh ipv6 rip
RIP process "1", port 521, multicast-group FF02::9, pid 102
    Administrative distance is 120. Maximum paths is 16
    Updates every 30 seconds, expire after 180
    Holddown lasts 0 seconds, garbage collect after 120
    Split horizon is on; poison reverse is off
    Default routes are not generated
    Periodic updates 53, trigger updates 2
    Interfaces:
      FastEthernet1/0
      Serial0/0
    Redistribution:
      None
RT1#
RT1#sh ipv6 rip next-hops
RIP process "1", Next Hops
    FE80::204:4EFF:FE22:2211/Serial0/0 [2 paths]
RT1#
=====
RT2#sh ipv6 int brief s0/0
Serial0/0 [up/up]
    FE80::204:4EFF:FE22:2211
    2345:6789:AAAA:2::2
RT2#

```

- Note that the next-hop address to reach 2345:6789:AAAA:3::/64 is the link-local address instead of global unicast address of RT2 Serial0/0. The **show ipv6 rip next-hops EXEC** command confirms that RIPng indeed uses link-local addresses as next-hop addresses.
- RIPng no longer performs automatic summarization as with RIPv2. It still sends out periodic full Update multicast packets every 30 seconds.
- Since IPv6 supports native authentication using the IPsec Authentication Header (AH), RIPng does not natively support authentication, instead rely on IPv6's inherent IPsec capabilities.

## Manually Configured Tunnel Configuration

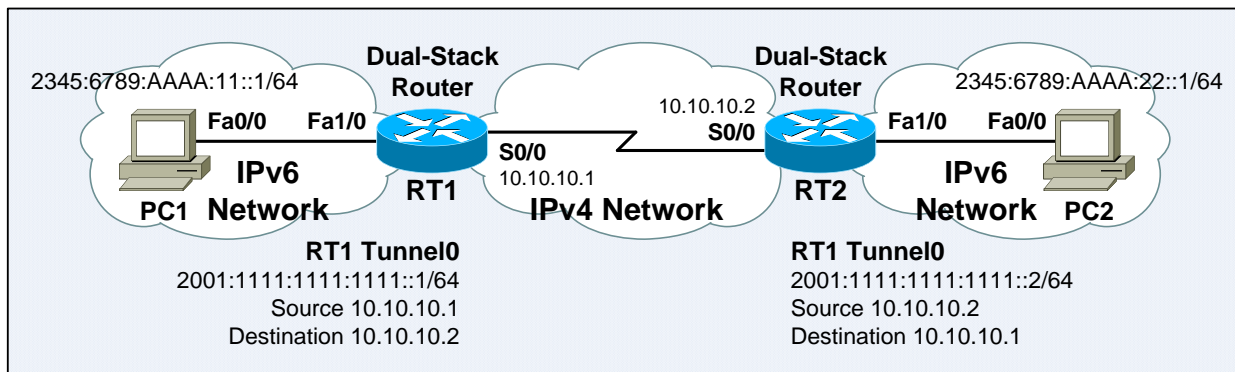


Figure 3-12: Sample IPv6 Tunnel Network

- IPv6 tunnels are configured on domain border routers that communicate with each other through an IPv4 network.
- Manually Configured Tunnel Configuration on RT1 and RT2:

```

RT1:
!
ipv6 unicast-routing
!
interface FastEthernet1/0
  ipv6 address 2345:6789:AAAA:11::1/64
  ipv6 rip tunnel-ripng enable
!
interface Serial0/0
  ip address 10.10.10.1 255.255.255.252
!
interface Tunnel0
  ipv6 address 2001:1111:1111:1111::1/64
  tunnel source Serial0/0
  tunnel destination 10.10.10.2
  tunnel mode ipv6ip
  ipv6 rip tunnel-ripng enable
!
=====
RT2:
!
ipv6 unicast-routing
!
interface FastEthernet1/0
  ipv6 address 2345:6789:AAAA:22::1/64
  ipv6 rip tunnel-ripng enable
!
interface Serial0/0
  ip address 10.10.10.2 255.255.255.252
!
interface Tunnel0
  ipv6 address 2001:1111:1111:1111::2/64
  tunnel source Serial0/0
  tunnel destination 10.10.10.1
  tunnel mode ipv6ip
  ipv6 rip tunnel-ripng enable
!

```

- After an IPv6 tunnel is created between the domain border routers, traffic need to be routed between the sites. This can be achieved using static routes or a dynamic routing protocol. Below shows that the route to the IPv6 network behind RT2 is learnt via RIPng:

```

RT1#sh ipv6 route
IPv6 Routing Table - 7 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
C   2001:1111:1111:1111::/64 [0/0]
    via ::, Tunnel0
L   2001:1111:1111:1111::1/128 [0/0]
    via ::, Tunnel0
C   2345:6789:AAAA:11::/64 [0/0]
    via ::, FastEthernet1/0
L   2345:6789:AAAA:11::1/128 [0/0]
    via ::, FastEthernet1/0
R 2345:6789:AAAA:22::/64 [120/2]
    via FE80::A0A:A02, Tunnel0
L   FE80::/10 [0/0]
    via ::, Null0
L   FF00::/8 [0/0]
    via ::, Null0
RT1#

```

- **Note:** The phrase *manually configured tunnels* refers to an RFC standard encapsulation of IPv6 inside IPv4 packets. There is no formal name for this feature; most documents refer to it as *manual tunnels*, *manual overlay tunnels*, *configured tunnels*, or *manually configured tunnels*.

## On-Demand Routing, RIPv2, and Routing Principles

- **On-Demand Routing** (ODR) requires minimal manual configuration and management overhead than static routing and provides IP routing information with minimal resource usage (eg: network bandwidth and router resources) compared to dynamic routing protocols.
- ODR is only applicable to **hub-and-spoke** topology, where each spoke (or stub) router is adjacent only to the hub router. The stub router typically has some LAN networks connected to it and a WAN connection to the hub router. The hub router needs to know the networks behind each spoke router, but the spoke routers only require a default route points back to the hub router.
- ODR utilizes **Cisco Discovery Protocol** (CDP) to carry network information between the spoke (or stub) and hub routers. Spoke routers use CDP to send IP prefixes information to the hub router; whereas the hub router sends a default route which points back to it to the spoke routers. ODR supports VLSM as **ODR routing updates carry subnet mask information**.
- ODR is not a routing protocol, as the information exchanged is limited to IP prefixes and default routes only. ODR does not report metric information – the hub router uses a hop count of 1 as the metric for all routes reported by ODR. However, ODR is able to dynamically obtain the routing information for stub networks without the overhead of a dynamic routing protocol, and default routes can be provided to the spoke routers without any manual configuration.
- The hub router inserts the stub networks learnt via ODR along with next-hop addresses to the spoke routers (based on the IP addresses learnt via CDP) into its routing table.
- If information about a stub network needs to be propagated to other parts of the network, the hub router can be configured to redistribute it into a dynamic routing protocol.
- The **router odr** global configuration command is used to enable ODR on the hub router; and no IP routing protocol configuration is required on the spoke routers.
- The **distribute-list** {*acl-num* | *acl-name*} **in** | **out** [*intf-type* *intf-num*] router subcommand can be used to limit the network prefixes that are permit to be learned via ODR in the hub router. The **timers basic** {*update-interval* *invalid* *holddown* *flush*} [*sleep-time*] router subcommand can be used to change the interval at which ODR routes are expired / flushed and being removed from the routing table.  
**Note:** Sleep timer is the interval (in ms) for postponing triggered (or flash) updates. Its value should be less than the update interval, or else the routing tables will become unsynchronized.
- ODR relies on CDP, hence CDP must be enabled. CDP is enabled by default on most interfaces, eg: Ethernet, Point-to-Point Serial, Frame Relay, etc.
- CDP updates are sent as Ethernet multicasts – 0100.0CCC.CCCC. On WAN links that require mapping statements, eg: ISDN dialer links and Frame Relay, the **broadcast** keyword is required in the mapping statements to allow broadcasts and multicasts to be propagated across the links.
- CDP updates are sent every 60 seconds by default, which may not be frequent enough to response to network topology changes. The **cdp timer** {*sec*} global configuration command can be used to change the CDP update interval. The **show cdp** and **show cdp interface EXEC** commands can be used to verify CDP configuration.

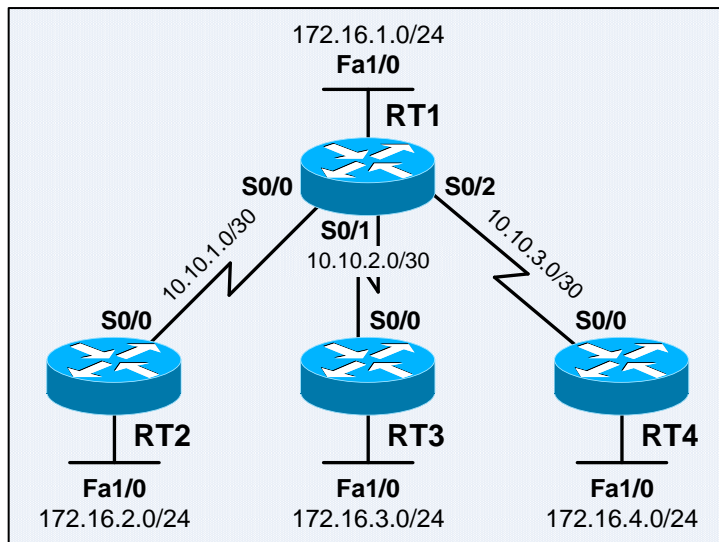


Figure 4-1: Sample ODR Network

- ODR configuration on RT1 and the routing table of RT2:

```

RT1#debug ip routing
IP routing debugging is on
RT1#
RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#router odr
RT1(config-router)#end
RT1#
00:10:11: RT: add 172.16.2.0/24 via 10.10.1.2, odr metric [160/1]
00:10:11: RT: NET-RED 172.16.2.0/24
00:10:25: RT: add 172.16.3.0/24 via 10.10.2.2, odr metric [160/1]
00:10:25: RT: NET-RED 172.16.3.0/24
00:10:36: RT: add 172.16.4.0/24 via 10.10.3.2, odr metric [160/1]
00:10:36: RT: NET-RED 172.16.4.0/24
RT1#sh ip route

Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 3 subnets
C       172.16.1.0 is directly connected, FastEthernet1/0
o       172.16.2.0 [160/1] via 10.10.1.2, 00:00:29, Serial0/0
o       172.16.3.0 [160/1] via 10.10.2.2, 00:00:15, Serial0/1
o       172.16.4.0 [160/1] via 10.10.3.2, 00:00:04, Serial0/2
    10.0.0.0/30 is subnetted, 2 subnets
C       10.10.1.0 is directly connected, Serial0/0
C       10.10.2.0 is directly connected, Serial0/1
C       10.10.3.0 is directly connected, Serial0/2
RT1#
=====
RT2#sh ip route

Gateway of last resort is 10.10.1.1 to network 0.0.0.0

    172.16.0.0/24 is subnetted, 1 subnets
C       172.16.2.0 is directly connected, FastEthernet1/0
    10.0.0.0/30 is subnetted, 1 subnets
C       10.0.1.0 is directly connected, Serial0/0
o*    0.0.0.0/0 [160/1] via 10.10.1.1, 00:00:22, Serial0/0
RT2#

```

- As soon as ODR is configured and running, routes from the stub routers will be shown in the routing table of the hub router. ODR routes are identified with the o character and not to be confused with the O character for OSPF routes.  
**Note:** Metric and administrative distance for ODR are 1 and 160 respectively.
- ODR routes are refreshed every 60 seconds due to the default CDP advertisement interval.

## RIPv2

- RIPv2 is a classless Distance-Vector routing protocol. It is designed to enhance the RIPv1 routing protocol. The most significant enhancement of RIPv2 is the advertisement of **subnet mask** information along with the subnet number in the routing updates. With this enhancement, RIPv2 supports classless routing, VLSM, route summarization, and discontinuous networks.  
**Note:** Distance-Vector algorithms are often being associated with Bellman-Ford algorithms. However, this is somewhat confusing and inaccurate as Link-State algorithms can also use the Bellman-Ford algorithm to perform route computation.
- RIPv2 route summarization improves routing scalability and efficiency in large networks by reducing the size of routing tables. However, RIPv2 is only able to summarize routes up to the classful network boundary. RIPv2 does not support CIDR-type summarization.
- RIPv2 advertises routing updates in a more efficient manner by using **multicasts**. RIPv1 updates are normally being discarded by end systems. RIPv2 advertises routing updates to other routers with the 224.0.0.9 multicast address. With multicasts, the packets can be discarded at either L2 or L3 instead of discarded based on invalid destination port number at the transport layer (L4).  
**Note:** RIP (both RIPv1 and RIPv2) updates are destined to UDP Port 520.
- 224.0.0.9 and 0100.5E00.0009 are the multicast IP and MAC addresses of RIPv2. Devices that can distinguish between a multicast and a broadcast at the data link layer (L2) can discard unwanted packets at the interface level. The worst case that will happen on devices that do not have such capability is that the RIPv2 update packets will be discarded at the network layer (L3) instead of the transport layer (L4), as those devices should never join the RIPv2 multicast group.
- Another enhanced feature in RIPv2 is security – **authentication**. Cisco RIPv2 implementation supports plain text and Message Digest 5 (MD5) authentications. The **ip rip authentication mode {md5 | text}** and **ip rip authentication key-chain {key-chain-name}** interface subcommands are used to enable RIPv2 authentication on an interface.

```

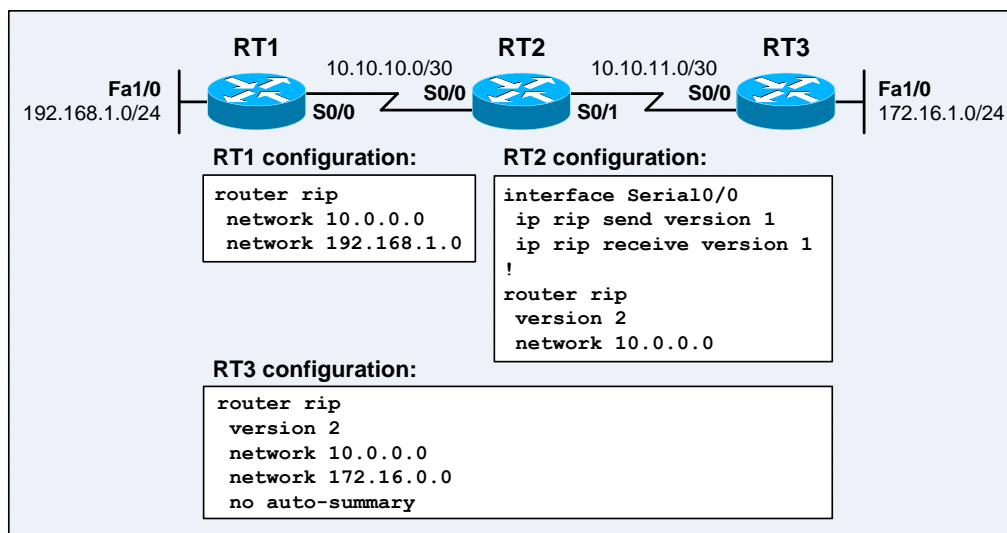
!
key chain kc01
  key 1
    key-string cisco123
!
interface Serial10/0
  ip address 10.10.10.1 255.255.255.0
  ip rip authentication mode md5
  ip rip authentication key-chain kc01
!

```

- The **router rip** global configuration command enables the RIPv1 process. By default, Cisco IOS receives versions 1 and 2 updates, but sends only version 1 updates.

- The **version {1 | 2}** router subcommand can be used to configure a router to send and receive routing updates **for the configured version only**.
- The **ip rip send version {ver}** and **ip rip receive version {ver}** interface subcommands can be used to control the version of the updates to be sent and received on a particular interface.

```
Router (config) #interface {intf-type intf-num}
Router (config-if) #ip rip send version {1 | 2 | 1 2}
Router (config-if) #ip rip receive version {1 | 2 | 1 2}
```

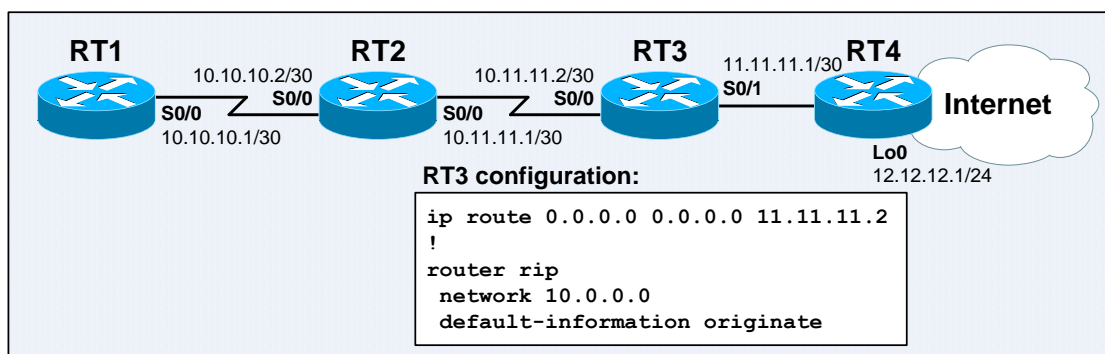


**Figure 4-2:** Sample RIPv2 Network

- A router configured with the **version 2** router subcommand sends and receives only RIPv2 updates. The **ip rip send version 1** and **ip rip receive version 1** interface subcommands configured on RT2's S0/0 interface override the default settings and tell Cisco IOS to both send and receive RIPv1 updates on the interface.

**Note:** RT1 will only have the 172.16.0.0/16 network in its routing table (instead of 172.16.1.0), as RIPv1 is a classful routing protocol which does not advertise subnet mask information along with routing updates – RT2 does not advertise the info to RT1 as they are communicating using RIPv1.

- The **default-information originate** router subcommand can be used to advertise a default route from a router through a routing protocol (RIPv1 for this case).
- **Note:** An RIP router configured the **default-information originate** command will generate a default route 0.0.0.0 even if it does not has a gateway of last resort.



**Figure 4-3:** RIP Default Route Advertisement

- Below shows the output of the **debug ip routing** and **show ip route** commands on RT1:

```

RT1#debug ip routing
IP routing debugging is on
RT1#
00:01:35: RT: add 0.0.0.0/0 via 10.10.10.2, rip metric [120/1]
00:01:35: RT: NET-RED 0.0.0.0/0
00:01:35: RT: default path is now 0.0.0.0 via 10.10.10.2
00:01:35: RT: new default network 0.0.0.0
00:01:35: RT: NET-RED 0.0.0.0/0
RT1#sh ip route

Gateway of last resort is 10.10.10.2 to network 0.0.0.0

    10.0.0.0/30 is subnetted, 1 subnets
R       10.11.11.0 [120/1] via 10.10.10.2, 00:00:14, Serial0/0
C       10.10.10.0 is directly connected, Serial0/0
R*    0.0.0.0/0 [120/1] via 10.10.10.2, 00:00:14, Serial0/0
RT1#
RT1#ping 12.12.12.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.12.12.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 52/76/112 ms
RT1#

```

- **Note:** RT2 would propagate the default route information from RT3 to RT1 without having the **default-information originate** RIP router subcommand configured on it.

## Static Routing and Default Routing

- Static routes are often being implemented in the following situations:
  - i) When it is undesirable to have dynamic routing updates forwarded across low-bandwidth links, eg: dial-up links.
  - ii) When a router is underpowered and has limited CPU or memory resources to run a dynamic routing protocol.
  - iii) When the administrator needs total control over the routes.
  - iv) When a backup route to a dynamically learned route is required.
  - v) When there is only one path to reach other network, eg: a stub network. This is also known as default routing.
  - vi) When a route should appeared as a **directly connected network** or locally attached link. This can be achieved by specifying an outgoing interface instead of a next-hop address in static routing configuration. The default administrative distance of static routes both specified with a next-hop address and an outgoing interface is 1.
    - Note:** A directly connected interface has an administrative distance of 0; a directly connected interface configured by specifying an outgoing interface in static routing configuration still has an AD of 1; not 0 as claimed by many sources!
- Default routes simplify the routing tables of Internet perimeter routers or internal stub routers. They can be manually configured or dynamically learned through routing protocol updates.



## Floating Static Routes

- Floating static routes static routes that appear in the routing table only when the primary route fails. They are often being used to configure **backup routes** or conditionally advertise a route.
- A floating static route is configured with an administrative distance that is higher than the default administrative distance of a dynamic routing protocol; hence it can be overridden by the dynamically learned route. A floating static route creates a **path of last resort** which is being used when the dynamically learned route is unavailable and removed from the routing table.

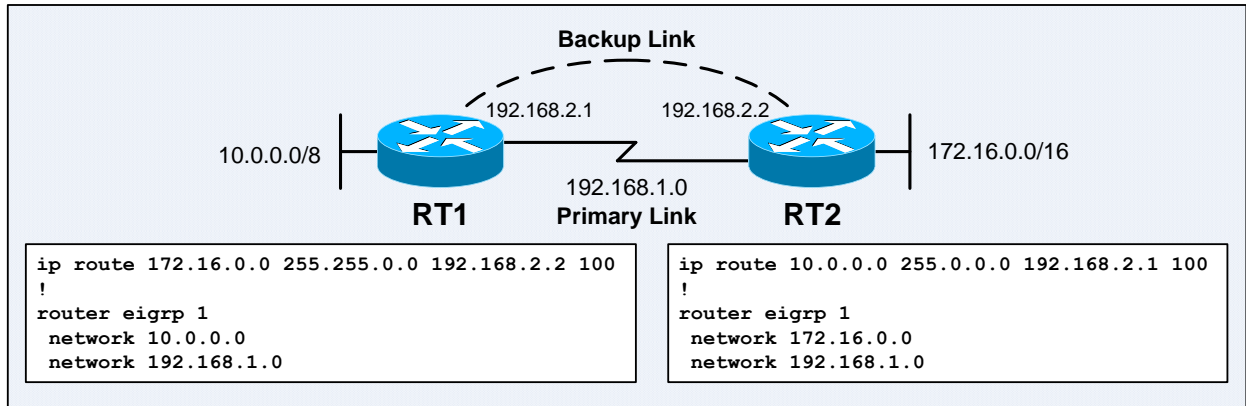


Figure 4-4: Floating Static Route

- The default AD of EIGRP is 90. The static routes are configured with an AD of 100 to make sure that the backup routes never override the primary routes. As long as the primary serial link is up, EIGRP will be used to exchange routing information. The static routes (backup routes) will only be inserted into the router routing tables when the serial link goes down and disables EIGRP.
- Below lists the default administrative distance values of different routing protocols. The lower the value, the higher the rating of trust of a routing information source – an inverse rating.

Route Type	Default AD
Directly connected	0
Static route out an interface	1 [1]
Static route to a next-hop address	1
EIGRP summary route	5
External BGP (EBGP)	20
EIGRP (internal)	90
IGRP	100
OSPF	110
IS-IS	115
RIPv1 and RIPv2	120
Exterior Gateway Protocol (EGP)	140
ODR	160
External EIGRP	170
Internal BGP (IBGP)	200
Unknown	255

[1] – It is not 0 as claimed by many sources!

- Based on the table above, routers believe static routes over any dynamically learned route. However, there are cases where this default behavior is not the desired behavior, eg: a static route that acts as the backup route to a dynamically learned route (floating static routes). The creation of floating static routes is an example of changing the administrative distance. Extra caution must be taken when changing the default AD values for routing protocols.

## Routes Insertion

- A Cisco router will consider the following criteria when inserting a route into its routing table:
  - i) **Valid next-hop IP address.** When a routing process receives a routing update, the router will first verify that the route has a valid next-hop IP address.
  - ii) **Administrative Distance.** If there are multiple routes learnt from **multiple routing protocols** to the same network, the route with the **lowest administrative distance** will be inserted into the routing table.
  - iii) **Metric.** If there are multiple routes learnt from the **same routing protocol** to the same network, the route with the **lowest metric** will be inserted into the routing table.
  - iv) **Prefix.** If there is no exact match to the advertised prefix (or subnet) in the routing table, the route will be inserted into the routing table.  
 Ex: A router with 3 routing processes running in it receives the following updates: EIGRP – 172.16.0.0/24, OSPF – 172.16.0.0/20, and RIP – 172.16.0.0/16. Since all routes have different prefix lengths or subnet masks, the routes are considered destined to different destinations and will be inserted into the routing table.

## IP Routing Protocols Comparison

- Protocol numbers, port numbers, and update reliabilities of common routing protocols:

Routing Protocol	Protocol Number	Port Number	Update Reliability
IGRP	9	–	Best-effort delivery
EIGRP	88	–	1-to-1 window
OSPF	89	–	1-to-1 window
RIP	–	UDP 520	Best-effort delivery
BGP	–	TCP 179	Uses TCP windowing

- IGRP, EIGRP, and OSPF are **transport layer protocols** that reside directly above the IP layer. IGRP uses connectionless (best-effort) delivery for its routing updates and do not required to acknowledge routing updates; whereas EIGRP and OSPF have more reliability built into their update processes – both require the acknowledgment of an update before sending another update – **1-to-1 window**, one update and one acknowledgment).
- RIP and BGP are **application layer protocols**. RIP uses UDP as its transport protocol. RIP updates are unreliable and being sent with best-effort delivery.
- BGP uses TCP as its transport layer protocol. It takes advantages of the reliability and windowing mechanisms of TCP. BGP routers often have more than 100000 routes in their routing tables. It is impractical for OSPF and EIGRP to send such amount of updates, as this will take a long time due to the overhead of the acknowledgment process (1-to-1 window). TCP 65536-byte window limit allows 65536 bytes to be sent without waiting for an acknowledgment, which allows BGP to send more routes per update than OSPF and EIGRP.
- IS-IS is a network-layer protocol. It does not use the services of IP to advertise its routing updates. IS-IS packets are encapsulated directly into data-link layer frames.

The maximum hop count for EIGRP (and IGRP) is 255, with the default value of 100. The **metric maximum-hops** `{hop-count}` router subcommand changes this value.

- Routing protocols comparison chart:

Characteristic	RIPv2	EIGRP	OSPF	IS-IS	BGP
Distance-vector	✓	✓ <sup>[1]</sup>	–	–	✓ <sup>[2]</sup>
Link-state	–	–	✓	✓	–
Hierarchical topology required	–	–	✓	✓	–
Classless	✓	✓	✓	✓	✓
VLSM support	✓	✓	✓	✓	✓
Autosummarization <sup>[3]</sup>	✓	✓	–	–	✓
Manual route summarization	✓	✓	✓	✓	✓
Metric	Hops	Composite metric	Cost	Cost	Path attributes
Convergence time	Slow	Very fast	Fast	Fast	Slow

[1] EIGRP is an advanced distance-vector routing protocol with some attributes or features that found in link-state routing protocols.

[2] BGP is a path-vector routing protocol.

[3] Autosummarization is enabled by default, but can be disabled. Disabling autosummarization would provide support for discontinuous networks. Route summarization in OSPF and IS-IS requires manual configuration.

- OSPF and IS-IS require a hierarchical topology with a backbone that carries inter-area traffic. OSPF and IS-IS use the Dijkstra Shortest Path First (SPF) algorithm for route calculation. **Cost** is used as the metric for route calculation. The cost of OSPF is based on **bandwidth** of the links.
- IS-IS uses **Complete Sequence Number PDU (CSNP)** multicasts for link-state database synchronization.
- **Holddown** has a different meaning for link-state routing protocols compared to holddown for distance-vector routing protocols. If a link-state router does not receive a Hello packet from a neighbor within the specified holddown interval, that neighbor is considered down and routes through that neighbor are considered unavailable. The equivalent term in OSPF is **dead interval**.
- Generally, the holddown or dead interval are 3 to 4 times the Hello interval.
- Several issues must be considered when changing the default timer intervals:
  - i) Smaller Hello interval results in faster detection of network topology changes, but can consume more bandwidth.
  - ii) Longer holddown interval might delay network convergence, as routers would take longer time to notice and react upon network topology changes.
  - iii) Holddown or dead interval must not be set too low until neighbor relationships can be torn down unnecessarily.
- BGP is a routing protocol that keeps track of paths to autonomous systems. BGP uses a very complicated metric based on the evaluation of the attributes of all paths. The main concern of BGP is **reliability**. Convergence speed is not important as with other routing protocols.
- BGP works differently than the other routing protocols. The BGP **network** *{network-addr}* [**mask** *{net-mask}*] router subcommand permits BGP to advertise a network only if the route to the network is in the routing table. The command supports CIDR-type prefixes (subnet masks) – individual subnets, or supernets can be advertised.
 

**Note:** At least one subnet of the specified major network must exist in the routing table in order for the classful network to be advertised. If a subnet mask is specified, the exact network (both address and mask) must exist in the routing table in order for the network to be advertised.

- Routing protocol default timers comparison chart:

Routing Protocol	Update Frequency	Hello Frequency	Other Timers
RIPv1 and RIPv2	Every 30 seconds. Triggered.	–	Invalid and holddown timers – 180 sec. Flush timer – 240 sec.
IGRP	Every 90 seconds. Triggered.	–	Invalid timer – 270 sec. Holddown timer – 280 sec. Flush timer – 630 sec.
EIGRP	Triggered.	60 sec for T1 or less, 5 sec for others.	Holddown timer – 180 sec for T1 or less, 15 sec for others.
OSPF	Triggered. LSAs flood every 30 min.	30 sec for NBMA, 10 sec for others.	Dead timer – 120 sec for NBMA, 40 sec for others.
IS-IS	Triggered. Link-state database synchronization every 10 seconds on LANs, and at startup on point-to-point links.	10 sec.	Hold timer – 30 sec.
BGP	Triggered.	60 sec.	Holddown timer – 180 sec.
ODR	60 seconds (default CDP update interval).	–	Invalid timer – 180 sec. Holddown timer – 0 sec. Flush timer – 240 sec. Sleep timer – 0 milliseconds.

- Route summarization support in various routing protocols:

Protocol	Autosummarization up to the Classful Network Boundary?	Ability to Turn off Autosummarization?	Ability to Summarize to other than Classful Network Boundary?
RIPv1	Yes	No	No
RIPv2	Yes	Yes	No
IGRP	Yes	No	No
EIGRP	Yes	Yes	Yes
OSPF	No	–	Yes
IS-IS	No	–	Yes

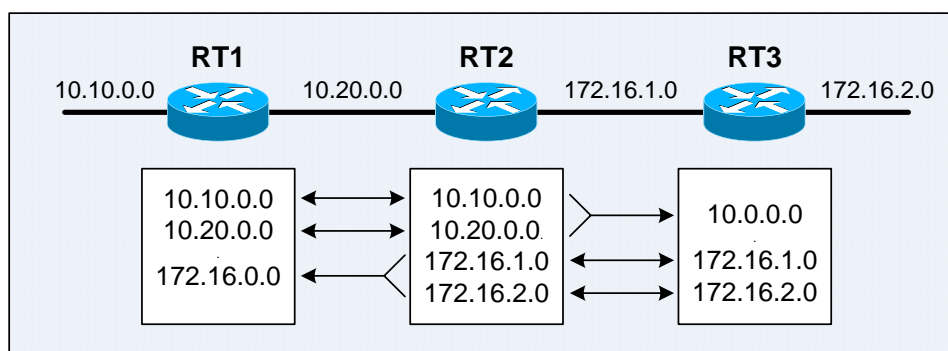


Figure 4-5: Autosummarization in Classful Routing

- Cisco IOS also provides an **IP unnumbered** feature that permits discontinuous networks to be separated by an unnumbered link.

## Network Convergence

- **Convergence time** is the time taken for the synchronization of routing tables of all routers upon a topology change. It should be as short as possible, as all routers must have the same view of the converged network in order to route data correctly.
- The convergence time for RIP and IGRP upon a topology change is slow, due to the detection (with invalid timer) and holddown mechanisms in distance-vector routing protocols.
- The convergence time for link-state routing protocols (eg: OSPF, IS-IS) upon a topology change is fast, as a router should have known all the links in its area.
- EIGRP has the fastest convergence time, as it maintains feasible successors (backup routes) in its topology table. When the best route fails, EIGRP can immediately use the feasible successor without performing additional best-path calculation, and without the worry of causing a loop.
- The **clear ip route** *{network-addr} {net-mask}* privileged command can be used to delete the route to a destination network from the routing table and forces the router to relearn the information about the network. This command is very useful for troubleshooting.

## Cisco IOS Routing and Switching Functions

- Both the routing and switching functions must be performed within a router to move a packet from an incoming (input) interface to one or more outgoing (output) interfaces. The destination path is first determined with the **routing process**, while the sending of the packet to the outgoing interface(s) is performed by the **switching process**.
- The **routing function** is responsible for learning the logical topology of the network and making decisions based on the knowledge – determine whether the incoming packet can be routed, and how to route it. Routing is more processing intensive and has higher latency than switching.
- The **switching function** is responsible for moving data within a router – from an inbound interface to an outbound interface, which is performed after making the routing decisions. Switching is very fast and has much lower latency than routing as it does not require extra header processing and routing table lookup.
- A packet destined to a router is accepted into the router if the frame header that encapsulated the packet contains the L2 address of one of the router's interfaces. After the framing is checked, the frame and its content (the packet) are buffered in memory and pending for further processing.
- If the router has not seen the source and destination L3 address of the packet before (the first packet that is being routed), the packet will be **process switched**, in which the first packet must be handled by both the routing and switching processes. A routing table lookup is first performed to determine which outgoing interface(s) the packet should be forwarded, and followed by the switching process which move the packet to the output queue of the outgoing interface(s). Finally the packet will be encapsulated in data link layer frame and transmitted to the medium.
- If **fast switching** is enabled, the packet is examined again, and an entry that consists of an IP prefix, the output interface, and the L2 layer header to be used when forwarding the packet is stored into a route cache. The effort spent for making the routing decision can be reused later.

- Subsequent packets destined to the same destination are forwarded (switched) using the routing information stored in the route cache. The routing function is not initiated and hence no CPU cycle is consumed for processing such packets.
- Fast switching is enabled by default with the **ip route-cache** interface subcommand (not shown in configuration files). The **show ip int {intf-type intf-num} EXEC** command displays the route cache information for a particular interface.

```

Router#sh ip int s0/0
Serial0/0 is up, line protocol is up
  Internet address is 10.10.10.1/24
  Broadcast address is 255.255.255.255
  Address determined by setup command
  MTU is 1500 bytes
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound access list is not set
  Proxy ARP is enabled
  Local Proxy ARP is disabled
  Security level is default
  Split horizon is enabled
  ICMP redirects are always sent
  ICMP unreachable are always sent
  ICMP mask replies are never sent
  IP fast switching is enabled
  IP fast switching on the same interface is enabled
  IP Flow switching is disabled
  IP CEF switching is enabled
  IP CEF Fast switching turbo vector
  IP multicast fast switching is enabled
  IP multicast distributed fast switching is disabled
  IP route-cache flags are Fast, CEF
  Router Discovery is disabled
  IP output packet accounting is disabled
  IP access violation accounting is disabled
  TCP/IP header compression is disabled
  RTP/IP header compression is disabled
  Policy routing is disabled
  Network address translation is disabled
  WCCP Redirect outbound is disabled
  WCCP Redirect inbound is disabled
  WCCP Redirect exclude is disabled
  BGP Policy Mapping is disabled
Router#

```

- The type of route cache being used and the contents in the cache depend on the router hardware and the type of interrupt content switching, eg: **fast switching**, **optimum switching**, **autonomous switching**, **silicon switching**, and **Cisco Express Forwarding (CEF)**. Different switching methods affect system performance and how load balancing is performed.
- There are several additional features that require additional processing and can affect switching performance, eg: **queuing**, **Random Early Detection (RED)**, **compression**, **filtering (ACLs)**, **encryption**, and **accounting**.

*This page is intentionally left blank*

## Chapter 5

# EIGRP

- EIGRP is a Cisco-proprietary hybrid routing protocol that contains features of distance-vector and link-state routing protocols. Some of its features are:
  - i) **Rapid convergence.** EIGRP uses the **Diffusing Update Algorithm** (DUAL) to achieve rapid convergence. DUAL not only calculates the best loop-free routes, but also calculates backup routes in advanced before they are actually being needed. An EIGRP router stores all available backup routes for fast react upon network topology changes. If no backup route exists in the routing table, an EIGRP router will query its neighbors until an alternative route is found.
  - ii) **Reduced bandwidth usage.** EIGRP does not send periodic updates as with DV protocols. It sends **partial updates** upon the route information changes (eg: path, metric). Additionally, the update is propagated only to routers that require it, instead of all routers within an area as with LS routing protocols.
  - iii) **Multiple routed protocols support.** EIGRP has been extended from IGRP to be network-layer independent. It supports IP, IPX, and AppleTalk with protocol-dependent modules (PDMs), which are responsible for protocol requirements specific to the corresponding routed protocols. EIGRP offers superior performance and stability when implemented in IPX and AppleTalk networks. EIGRP maintains a **neighbor table**, a **topology table**, and a **routing table** for each running routed protocols (PDMs).
  - iv) **Support all LAN and WAN data link protocols and topologies.** EIGRP does not require special configuration across any L2 protocols. OSPF requires different configurations for different L2 protocols, eg: Ethernet and Frame Relay. EIGRP was designed to operate effectively in both LAN and WAN environments. EIGRP supports all multi-access networks, eg: Ethernet, Token Ring, FDDI, and all WAN topologies – leased lines, point-to-point links, and non-broadcast multiaccess (NBMA) topologies, eg: X.25, SMDS, ATM, and Frame Relay.
- EIGRP has its roots as a distance-vector routing protocol (EIGRP is based on IGRP). It is considered an advanced DV routing protocol with traditional DV features, eg: autosummarization, easy configuration; and LS features, eg: dynamic neighbor discovery. Another distance-vector rule is that if a neighbor is advertising a destination, it must also be using that route to forward packets to the particular destination.
- EIGRP (Enhanced IGRP) provides many enhancement features over IGRP, a traditional DV routing protocol, mainly in convergence properties and operating efficiency. Traditional DV routing protocols send periodic full routing updates, which consume unnecessary bandwidth.
- EIGRP utilizes multicasts and unicasts only; broadcasts are not being used. As a result, end systems will not affected by the routing updates and queries.
- EIGRP is a **transport layer** protocol that relies on IP packets to deliver its routing information. EIGRP packets are encapsulated in IP packets with the Protocol Number field value 88 (0x58) in the IP header. Some EIGRP packets are sent as multicasts (destination IP address 224.0.0.10), while others are sent as unicasts.
- A significant advantage of EIGRP (and IGRP) over other routing protocols is the support for **unequal-cost load balancing**.
- EIGRP performs autosummarization by default, but this behavior can be disabled with the **no auto-summary** router subcommand.



- **Neighbor table** lists the directly connected adjacent EIGRP routers to ensure bidirectional communication with the neighbors. It is similar to the neighborship database in LS routing protocols. It maintains information such as address, hold time, and interface which an adjacent router connected to. An EIGRP router keeps a neighbor table for each running routed protocol. EIGRP routers must form neighbor relationships before exchanging EIGRP updates.
- **Topology table** maintains **all advertised routes to all destinations**, along with the advertising neighbors and advertised metric for each destination. The term “topology table” is confusingly named, as it does not actually store the complete network topology, but rather the routing tables from the directly connected neighbors. All successors and feasible successors to all destinations will be maintained in this table.
- The best routes to a destination will be selected from the EIGRP topology table and placed into the **routing table**. An EIGRP router maintains 1 routing table for each running routed protocol. It contains all best routes selected from the EIGRP topology table and other routing processes. Successors and feasible successors (when unequal-cost load balancing is enabled with the **variance** router subcommand) will be selected from the topology table and stored in this table.
- The **show ip eigrp neighbors**, **show ip eigrp topology**, and **show ip route EXEC** commands display the EIGRP neighbor table, EIGRP topology table, and routing table.
- **Successor** is the **lowest-metric best path** to reach a destination. EIGRP successor routes will be placed into the routing table.
- **Feasible Successor (FS)** is the **best alternative loop-free backup path** to reach a destination. Because it is not the least-cost or lowest-metric path, therefore it is not being selected as the primary path to forward packets and not being inserted into the routing table. Feasible successors are important as they allow an EIGRP router to recover immediately upon network failures and hence reduce the number of DUAL computations and therefore increase performance. The convergence time upon a successor failure with a feasible successor exists is in the range of 2 to 4 seconds (1 ping drop). Feasible successor routes are maintained in the topology table only.
- EIGRP routers use the following procedures to populate their routing tables:
  - i) Each router advertises its IP routing table to all adjacent neighbors in the neighbor table.
  - ii) Each router stores the routing tables of the adjacent neighbors in the topology table.
  - iii) Each router examines its topology database to determine the successor and feasible successor routes to every destination network.
  - iv) The best route to a destination network as selected from the EIGRP topology table or other routing processes is inserted into the routing table

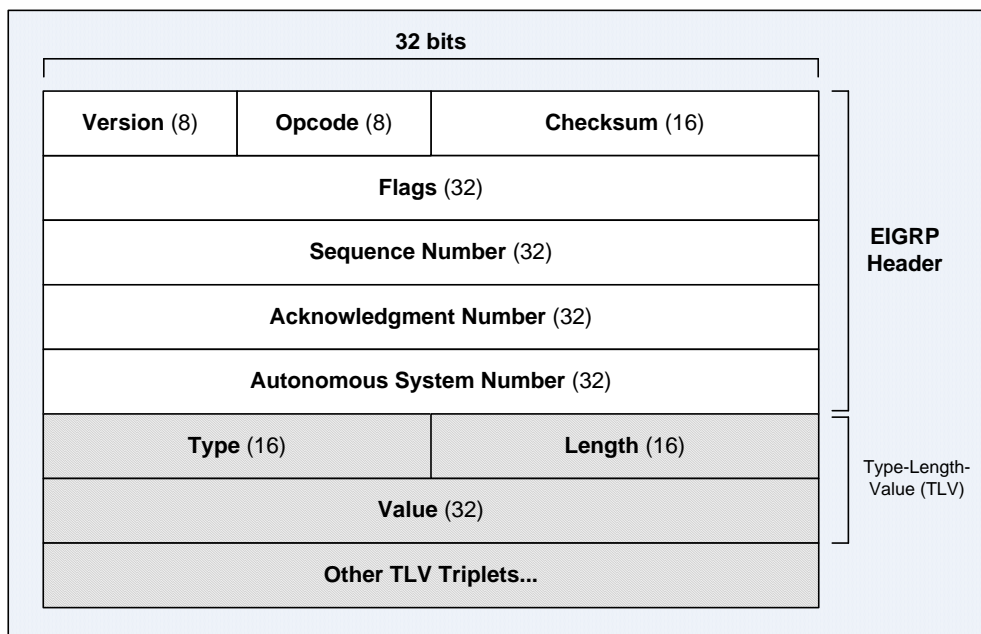
## EIGRP Packet Format

- EIGRP sends out the following 5 types of packets:

<b>Hello</b>	Used to discovery neighbor before establishing adjacency. EIGRP Hellos are sent as <b>multicasts</b> and contain an acknowledgment number of 0. EIGRP routers must form neighbor relationships before exchanging EIGRP updates.
<b>Update</b>	Used to communicate the routes that a particular router has used to converge. EIGRP Updates are sent as <b>multicasts</b> when a new route is discovered or when convergence is completed (the route becomes passive); and are sent as <b>unicasts</b> when synchronizing topology tables with neighbors upon the EIGRP startup. They are sent <b>reliably</b> between EIGRP routers.
<b>Query</b>	Used to query other EIGRP neighbors for a feasible successor when DUAL is re-computing a route in which the router does not have a feasible successor. EIGRP Queries are sent <b>reliably</b> as <b>multicasts</b> .
<b>Reply</b>	Sent as the response to an EIGRP Query packet. EIGRP Replies are sent <b>reliably</b> as <b>unicasts</b> .
<b>Acknowledge</b>	Used to acknowledge EIGRP Updates, Queries, and Replies; Hello and ACK packets do not require acknowledgment. ACKs are Hello packets that contain no data and a <b>non-zero</b> acknowledgment number and are sent as <b>unicasts</b> .

- An EIGRP router sends Hello packets out all EIGRP-enabled interfaces. The EIGRP multicast address is 224.0.0.10. An EIGRP router only establishes neighbor relationships (adjacencies) with other routers within the **same autonomous system**.
- EIGRP Hello packets are sent every **5 seconds** on LANs (eg: Ethernet, Token Ring, and FDDI) and point-to-point links (eg: PPP, HDLC, Frame Relay and ATM point-to-point subinterfaces, and multipoint circuits with bandwidth greater than T1 (eg: ISDN PRI, ATM, and Frame Relay), and **60 seconds** on T1 or low-speed interfaces (eg: ISDN BRI, X.25, ATM, and Frame Relay). The **ip hello-interval eigrp** {as-num} {sec} interface subcommand configures the Hello interval for an EIGRP routing process running upon an interface.
- The EIGRP neighbor table also maintains the **hold time** – the amount of time a router considers a neighbor is up without receiving a Hello or any EIGRP packet from the particular neighbor. The **ip hold-time eigrp** {as-num} {sec} interface subcommand configures the hold time interval for an EIGRP routing process. The hold time interval is recommended to be at least 3 times the Hello interval. In fact, the hold time interval is 3 times the Hello interval by default. The hold time interval is not automatically adjusted upon the change of the Hello interval. Once the Hello interval is changed, the hold time interval must be manually configured according to the new Hello interval.  
**Note:** The newly configured hold time value affects neighbor routers instead of local router! Verify the newly configured hold time interval with the **show ip eigrp neighbors EXEC** command on neighbor routers. The hold time value is a parameter in the Hello packet; therefore a neighbor router which receives the Hello packet will be in effect.
- If an EIGRP packet is not received before the expiration of the hold time interval, the neighbor adjacency is deleted, and all topology table entries learnt from the neighbor will be removed, as well as send out an update indicating that the routes are unreachable. If the neighbor is a successor for some destination networks, those networks are removed from the routing table, and alternative paths will be recomputed via DUAL.

- A route is considered **passive** when the router is not performing recomputation for that route; while a route is considered **active** when the router is performing recomputation to seek for a new successor when the existing successor has become invalid.

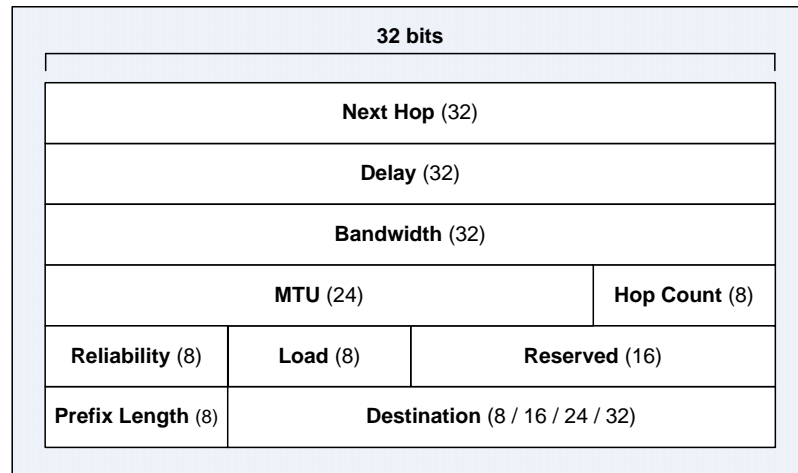


**Figure 5-1: EIGRP Packet Format**

- The EIGRP header comprises of the following fields:

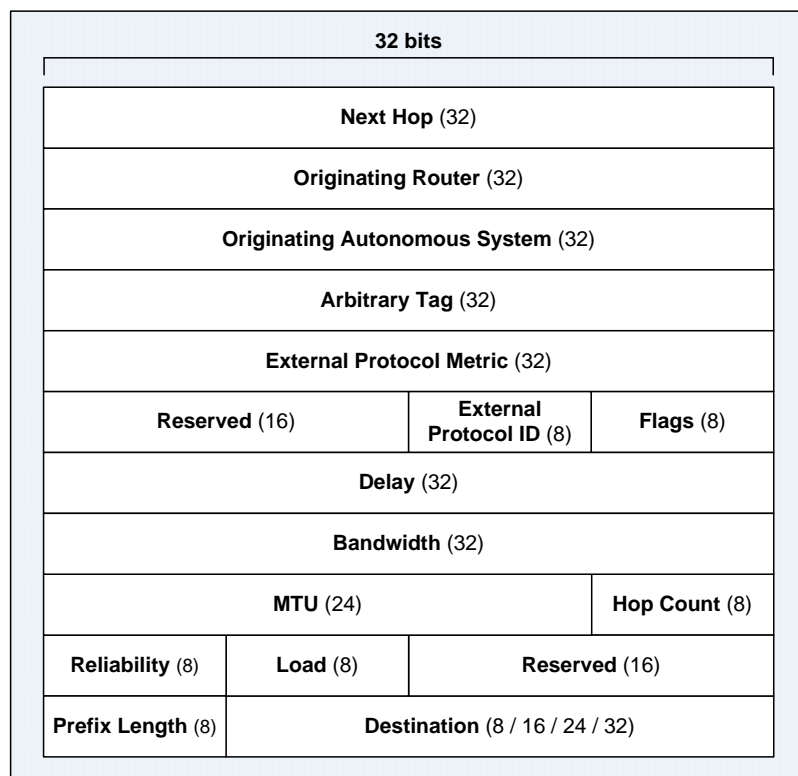
Field	Description
<b>Version</b>	Identifies the EIGRP process version. The current EIGRP version is 2.
<b>Opcode</b>	Identifies the EIGRP packet type – Update (0x01), Query (0x03), Reply (0x04), Hello (0x05). It determines the TLVs that follow the EIGRP header. <b>Note:</b> ACKs are Hello packets that contain a <b>non-zero</b> ACK number.
<b>Checksum</b>	The checksum of the <b>entire EIGRP packet</b> , excluding the IP header.
<b>Flags</b>	1st LSB bit (0x00000001) – Init bit, used indicate the first set of routing updates upon establishing a new neighbor relationship. 2nd LSB bit (0x00000002) – Conditional Receive bit, used in the Cisco-proprietary reliable multicast protocol – <b>Reliable Transport Protocol (RTP)</b> . Other bits are not being used.
<b>SEQ and ACK</b>	Used by RTP for reliable EIGRP message exchange.
<b>AS Number</b>	Identifies the autonomous system of an EIGRP packet. An EIGRP process only process EIGRP packets within an EIGRP domain (same AS number).
<b>Type / Length / Value (TLV)</b>	TLVs are comprise of a 16-bit Type field, a 16-bit Length field, and a vary number of fields depends on the type of TLV. <b>General TLVs:</b> 0x0001 – EIGRP parameters – K values and hold time. Size of 12 bytes. 0x0002 – Message Digest 5 (MD5) authentication data. Size of 40 bytes. 0x0003 – Sequence. Used by RTP. 0x0004 – Software versions – IOS and EIGRP release versions. Size 8 bytes. 0x0005 – Next Multicast Sequence. Used by RTP. 0x0006 – EIGRP stub parameters. <b>IP TLVs:</b> 0x0102 – IP internal route. Size of 28 bytes. 0x0103 – IP external route. Size of 48 bytes.

- An **internal route** contains a destination network within an EIGRP domain; while an **external route** contains a destination network outside an EIGRP domain, eg: redistributed routes from other routing processes into an EIGRP domain.
- EIGRP IP internal routes have a Type field of 0x0102. EIGRP metric information is similar to IGRP, with 2 new fields – Next Hop, which can specify a different next-hop router than the advertising router to send packets destined to the Destination/Prefix Length; and Prefix Length (for VLSM support). The TLV triplets Value fields contain the information as in Figure 5-2.



**Figure 5-2: EIGRP IP Internal Route Packet Format**

- In IGRP, when RT2 sends an update to RT1 with a destination network number 10.0.0.0, the next-hop of RT1 to the 10.0.0.0 network is RT2. With EIGRP, RT2 can send an update to RT1 with a Next Hop field of RT3. Ex: RT2 and RT3 are running RIP and RT2 is redistributing RIP routes into EIGRP. When RT2 sends an update to its neighbors on a shared network, RT2 can tell them (RT1) to send traffic directly to RT3 instead of RT2. This saves RT2 from having to accept and reroute the packets to RT3. **Note:** This setup is not being verified yet!

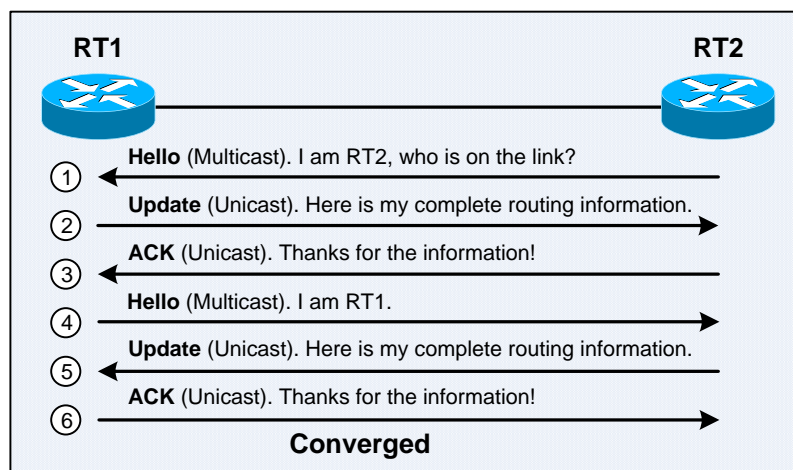


**Figure 5-3: EIGRP IP External Route Packet Format**

- The Originating Router, Originating AS, External Protocol Metric, and External Protocol ID fields specify the information about the source (router and routing protocol) from which an external route is derived. The common external routing protocol identifiers are 0x01 – IGRP, 0x02 – EIGRP (from different AS), 0x03 – Static Route, 0x04 – RIP, 0x06 – OSPF, 0x07 – IS-IS, 0x08 – EGP, 0x09 – BGP, 0x0A – ISO IDR, and 0x0B – directly connected link.  
**Note:** The **Inter-Domain Routing Protocol (IDRP)** which is defined for OSI-based environments is similar to **Border Gateway Protocol (BGP)** in the TCP/IP environments. Cisco IOS Software does not support IDR!
- **Note:** The **eigrp router-id {ip-addr}** EIGRP router subcommand configures the Router ID for an EIGRP router. However, the EIGRP **show** commands seldom display the Router ID. The EIGRP Router ID is used only when redistributing external routes into EIGRP.
- The Arbitrary Tag field is used to carry a tag set by route maps – **route tagging**.
- A delay of 0xFFFFFFFF or 4294967295 (infinite distance) indicates an unreachable route.

## EIGRP Neighbors

- EIGRP routers can become neighbors even if the Hello and hold time values do not match, in which the Hello and hold time intervals can be set independently on different routers. The EIGRP hold time has the same function as the OSPF dead time.
- EIGRP cannot form neighbor relationships using secondary addresses, as only primary addresses are used as the source IP addresses of all EIGRP packets. A neighbor relationship is formed between EIGRP routers when their primary addresses reside in the same subnet, they reside in the same AS domain, and the metric calculation constants – **K values** for the link are same.
- EIGRP routers send out multicast Hello packets to discover neighbors and form neighborships. An EIGRP router builds a neighbor table with Hello packets received from adjacent EIGRP routers running the same routed protocol. Only adjacent routers will exchange routing updates.
- When an EIGRP router discovers a new neighbor, it will send an update about its known routes to the new neighbor and receives the same information from the new neighbor. These updates are used to populate the topology table, which maintains the advertised metric from neighbors for all destinations and the metric that the local router uses to reach the destinations – **feasible distance**. The feasible distance metric values will be advertised to other neighbors.



**Figure 5-4:** EIGRP Initialization – Neighbor and Route Discovery

- Below describes the steps in the EIGRP initial neighbor and route discovery process:
  - i) An EIGRP router (RT2) sends out a Hello packet through all its interfaces.
  - ii) RT1 which receives the Hello packet reply with a **startup update** packet that contain all its routes in its routing table (RT1 considers RT2 as its neighbor). Even though RT2 received an Update packet from RT1, it does not consider RT1 as its neighbor until it receives a Hello packet from RT1 – the EIGRP: Neighbor not yet found message will be displayed in the output of the **debug eigrp neighbors** privileged command. An EIGRP router considers another router as a neighbor and accepts the update information only after it received a Hello packet from the neighboring router.
  - iii) RT2 replies RT1 with an ACK packet, indicating the receipt of the update information.
  - iv) RT2 inserts the update information, which includes all destinations along with their associated metrics as advertised by neighboring routers – **advertised distance**.
  - v) Step 4, 5, and 6 are similar to Step 1, 2, and 3.

**Note:** Above shows a sample EIGRP neighbor establishment scenario. There are very high chances where both routers send Hello, Update, or ACK packets to each other at the same time. However, the sequence of receive Hello, reply with Update, and receive ACK will still follow.
- The **show ip eigrp topology EXEC** command displays only the successor and feasible successor routes, as well as DUAL states which are very useful for debugging EIGRP problems. The **show ip eigrp topology all-links** and **show ip eigrp topology detail-links EXEC** commands display all entries in the topology table.

```

Router#sh ip eigrp topology | b 172.16.1.0
P 172.16.1.0/24, 1 successors, FD is 2172416
   via 10.10.10.2 (2172416/28160), Serial0/0
Router#
Router#sh ip eigrp topology all-links | b 172.16.1.0
P 172.16.1.0/24, 1 successors, FD is 2172416, serno 24
   via 10.10.10.2 (2172416/28160), Serial0/0
   via 10.10.10.6 (2684416/2172416), Serial0/1
Router#

```

**Note:** The highlighted path is not a feasible successor as its advertised distance is not less than the feasible distance – both values are the same.

- EIGRP distinguishes itself from other routing protocols with its route selection process. EIGRP maintains successor (primary) and feasible successor (backup) routes and inserts them into the topology table (up to 6 per destination). The primary route is then inserted into the routing table.

## EIGRP Metric Computation

- EIGRP supports the following types of routes:

<b>Internal</b>	Routes that are originated within an EIGRP autonomous system.
<b>External</b>	Routes that are learnt from another routing protocol or another EIGRP AS.
<b>Summary</b>	Routes that encompass multiple subnets. EIGRP summary routes have an administrative distance value of 5 (better than any dynamically learned route). AD values are <b>locally significant</b> and hence are not propagated to other routers.

- EIGRP uses **Diffusing Update Algorithm (DUAL)** to calculate and select **loop-free** primary and backup routes to a destination. When the primary routes fails, EIGRP can immediately uses a backup route without the need for holddown, and hence results in fast convergence.
- The EIGRP composite metric calculation can use up to 5 variables, but only the following 2 are used by default (K1 and K3):

<b>K1 – Bandwidth</b>	The minimum or lowest bandwidth between the source and destination.
<b>K3 – Delay</b>	The cumulative interface delay values along the path.

- The following variables are not commonly used, as they often cause **frequent recalculation** of the topology table.

<b>K2 – Load Utilization</b>	The worst load on a link between the source and destination based on the packet rate and the configured interface bandwidth.
<b>K4 – Reliability</b>	The worst reliability between the source and destination.
<b>K5 – Maximum Transmission Unit</b>	The smallest MTU along a path. MTU is included in the EIGRP Update packets but was never included in the formula used for metric calculation.

- EIGRP calculates the metric to a network by adding weighted values for variables of the links. Below shows the weights attributed to the K variables:

- i) K1 = Bandwidth (1)
- ii) K2 = Load Utilization (0)
- iii) K3 = Delay (1)
- iv) K4 = Reliability (0)
- v) K5 = MTU (0)

- The EIGRP metric calculation formula is as below:

$$metric = \left[ \left( K1 \times \frac{10^7}{BW_{min}} + \frac{K2 \times BW_{min}}{256 - load} + K3 \times \sum delays \right) \times \frac{K5}{K4 + reliability} \right] \times 256$$

- When the weight for K5 as 0, the  $\frac{K5}{K4 + reliability}$  will not be in effect and will be taken as 1.

- The EIGRP metric calculation formula with default weighted K values will be simplified as:

$$\begin{aligned} metric &= \left[ \left( 1 \times \frac{10^7}{BW_{min}} + \frac{0 \times BW_{min}}{256 - load} + 1 \times \sum delays \right) \times 1 \right] \times 256 \\ &= \left( 1 \times \frac{10^7}{BW_{min}} + \frac{0 \times BW_{min}}{256 - load} + 1 \times \sum delays \right) \times 256 \\ &= \left( \frac{10^7}{BW_{min}} + \sum delays \right) \times 256 \end{aligned}$$

- K values are transmitted in the EIGRP Parameters TLV in the EIGRP Hello packets. They should be modified only after careful planning and must be set identically on all routers within an EIGRP domain. Changing the weight for these constants is not recommended.
- The format of the bandwidth and delay values is different from those displayed in the **show interfaces EXEC** command. The EIGRP bandwidth metric value is calculated with the formula **1000000 / minimum bandwidth along the path (in kbps)**; while the EIGRP delay metric value is calculated as the **cumulative delays along the path (in tens of microseconds / usec)**. **Notes:** tens of microseconds = 10 usec. The **show interfaces EXEC** command displays delay value in microseconds. By issuing the **delay 100** interface subcommand, the delay for an interface will be displayed as 1000 usec in the **show interfaces EXEC** command.

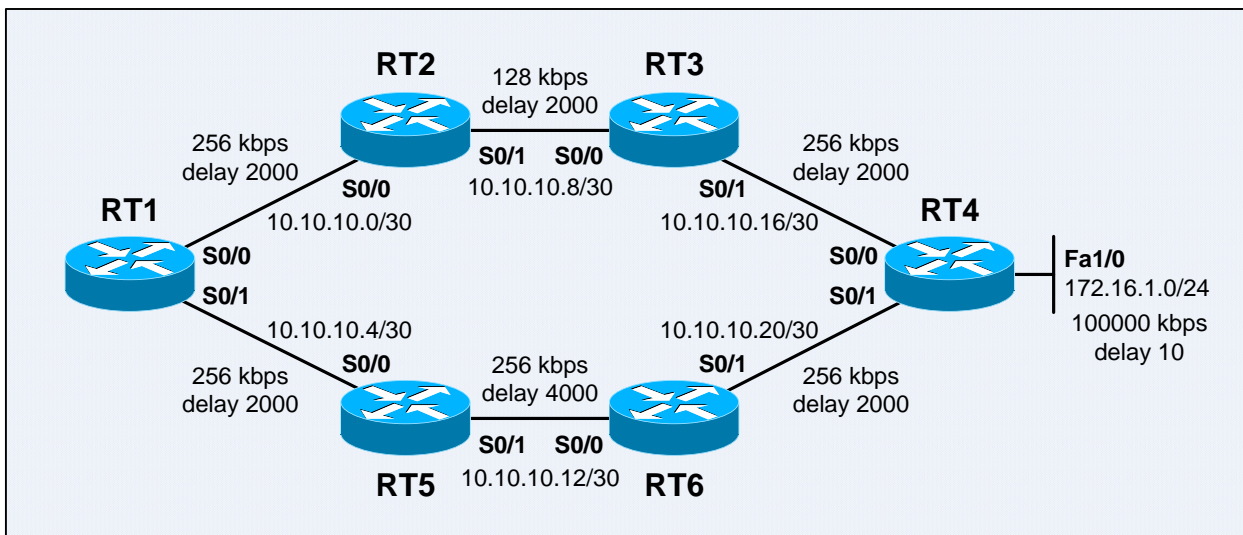
```

Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#int s0/0
Router(config-if)#delay ?
    <1-16777215> Throughput delay (tens of microseconds)

Router(config-if)#delay 100
Router(config-if)#do sh int s0/0 | in DLY
    MTU 1500 bytes, BW 1544 Kbit, DLY 1000 usec,
Router(config-if)#

```

- EIGRP metrics are represented in a 32-bit format (**1 – 4’294’967’296**) instead of the 24-bit format (**1 – 16’777’216**) as with IGRP. This provides additional granularity when determining the successor and feasible successor. Present-day bandwidth ranges from 9600bps to 10Gbps. EIGRP 32-bit metric format accommodates this range better than IGRP 24-bit metric format.
- EIGRP uses the same algorithm used by IGRP for metric calculation. The EIGRP metric is the IGRP metric multiplied by 256. When redistributing IGRP routes into an EIGRP domain, the IGRP metric is multiplied by 256 to compute the EIGRP-equivalent metric. When redistributing EIGRP routes to an IGRP routing domain, the EIGRP metric is divided by 256 to compute the IGRP-equivalent metric.



**Figure 5-5: Sample Network for EIGRP Metric Calculation**

- Figure 5-5 shows RT1 has 2 paths to reach 172.16.1.0/24. The bandwidth (in kbps) and delay (in tens of microseconds) of all links are also shown.



- The minimum bandwidth along the upper path (RT1 – RT2 – RT3 – RT4) is 128kbps. The EIGRP bandwidth calculation for this path is:

$$\text{bandwidth} = \frac{10000000}{128} = 78125$$

The delay through this path is:

$$\begin{aligned} \text{delays} &= (\text{RT1} - \text{RT2 delay}) + (\text{RT2} - \text{RT3 delay}) + (\text{RT3} - \text{RT4 delay}) + \text{RT4 Fa1/0 delay} \\ &= 2000 + 2000 + 2000 + 10 \\ &= 6010 \end{aligned}$$

Finally, the EIGRP metric for this path is:

$$\begin{aligned} \text{metric} &= (\text{min-bandwidth} + \text{delays}) \times 256 \\ &= (78125 + 6010) \times 256 \\ &= 21538560 \end{aligned}$$

- The minimum bandwidth along the upper path (RT1 – RT5 – RT6 – RT4) is 256kbps. The EIGRP bandwidth calculation for this path is:

$$\text{bandwidth} = \frac{10000000}{256} = 39062.5 = 39062$$

The delay through this path is:

$$\begin{aligned} \text{delays} &= (\text{RT1} - \text{RT5 delay}) + (\text{RT5} - \text{RT6 delay}) + (\text{RT6} - \text{RT4 delay}) + \text{RT4 Fa1/0 delay} \\ &= 2000 + 4000 + 2000 + 10 \\ &= 8010 \end{aligned}$$

Finally, the EIGRP metric for this path is:

$$\begin{aligned} \text{metric} &= (\text{min-bandwidth} + \text{delays}) \times 256 \\ &= (39062 + 8010) \times 256 \\ &= 47072 \times 256 \\ &= 12050432 \end{aligned}$$

- RT1 chooses the lower path over the upper path due to lower metric value. RT1 installs the route through the lower path with a next-hop of RT5 and a metric of 12050432 in its IP routing table.

```
RT1#sh ip eigrp topo 172.16.1.0/24
IP-EIGRP (AS 100): Topology entry for 172.16.1.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 12050432
  Routing Descriptor Blocks:
  10.10.10.6 (Serial0/1), from 10.10.10.6, Send flag is 0x0
    Composite metric is (12050432/11538432), Route is Internal
  Vector metric:
    Minimum bandwidth is 256 Kbit
    Total delay is 80100 microseconds
    Reliability is 255/255
    Load is 1/255
    Minimum MTU is 1500
    Hop count is 3
```

- The bottleneck along the upper path is the 128kbps link between RT2 and RT3, as this means that the throughput rate through this path would be at a maximum of 128kbps. While the lowest speed along the lower path is 256kbps, allowing the throughput rate up to that speed.

**Note:** Bandwidth is the largest contributor upon the EIGRP metric. The delay value provides granularity when choosing the best path out from multiple paths with equivalent bandwidth.

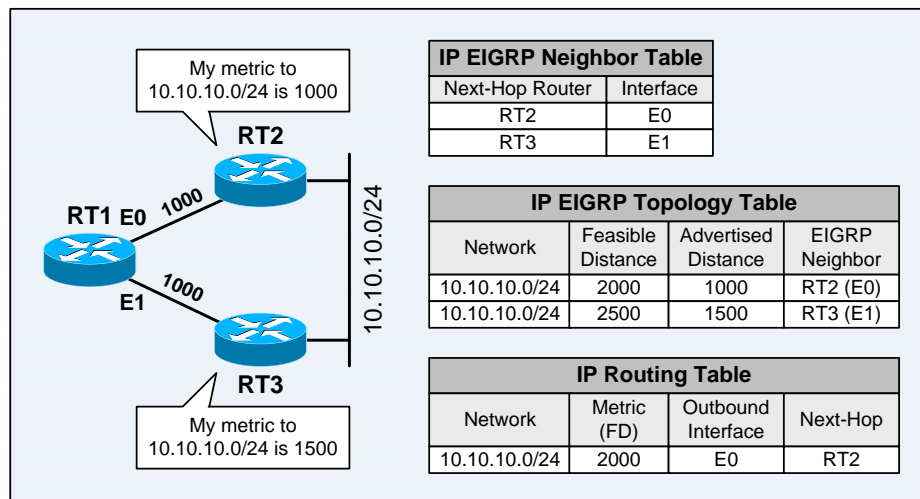
## EIGRP DUAL – Diffusing Update Algorithm

- EIGRP uses the DUAL finite-state machine to tracks all routes advertised by all neighbors with the topology table, performs route computation on all routes to select an efficient and loop-free path to all destinations, and inserts the lowest metric route into the routing table.
- EIGRP uses the **advertised distance** and **feasible distance** to determine the **successor** (best route) and **feasible successor** (backup route) to a destination network.

<b>Advertised Distance</b>	The EIGRP metric for a next-hop EIGRP neighboring router to reach a destination network. Also known as <b>Reported Distance</b> .
<b>Feasible Distance</b>	The EIGRP metric for the local router to reach a destination network. Theoretically it is the sum of the advertise distance of an EIGRP neighbor and the metric to reach the neighbor, but actually the local router would recalculate the EIGRP metric to the destination network.

The xxx and yyy in the via A.B.C.D (xxx/yyy), *interface* entry in the **show ip eigrp topology EXEC** command represent **feasible distance** and **advertised distance** respectively.

- An EIGRP router compares all feasible distances to reach a destination network in its topology table. The lowest-metric route(s) will be placed in its IP routing table as the successor route(s).



**Figure 5-6:** EIGRP Neighbor Table, Topology Table, and Routing Table

- RT1's topology table has 2 paths to reach 10.10.10.0/24. The EIGRP metric for RT1 to reach RT2 and RT3 is 1000. The metric (1000) will be added to the respective AD from each router, which represents the FD for RT1 to reach network 10.10.10.0/24 via a particular neighbor. RT1 chooses the least-cost FD (2000) as the best route and inserts it into its IP routing table. An EIGRP route in the routing table is the best FD selected from the EIGRP topology table. **Note:** Advertised distance is used only to calculate the feasible distance (the least-cost path); it does not affect the selection of the best route. Because there could be cases where the advertised distance from a neighbor is promising, but the metric to the neighbor is disappointing. Hence the path via the neighbor would not be selected as the feasible distance.
- **Successor** is a neighboring router that has the least-cost or lowest-metric (best) path of all possible paths to reach a destination network. After the best path to reach a network is selected, EIGRP inserts the destination network, metric to reach the network, outbound interface to reach the next-hop router, and the IP address of the next-hop router into the IP routing table. If the EIGRP topology table has multiple equal-cost FD entries to a destination network, all successors (4 by default) for the destination network will be inserted into the routing table.

- **Feasible Successor** is a neighboring router that does not provide the least-cost path but provides a backup route. The route through the feasible successor must be **loop-free**. Successors and feasible successors are selected at the same time during the metric computation process. Feasible successor routes are maintained in the topology table, as well as in the routing table when unequal-cost load balancing is enabled with the **variance** router subcommand.

To be qualified as a feasible successor, the AD from a neighboring router must be **less than** (and not equal to) the FD of the successor route to a destination network. This ensures an FS is **loop-free** and will never route packets to the destination network through the local router, or else this would result in a routing loop. The AD of the local router (RT1) would often be greater than the FD of the neighboring feasible successor router (RT2) and will eventually notify the neighboring feasible successor router (RT2) that the local router is not a feasible successor for the destination network and never route packets to the destination network via the local router (RT1). When the AD from a neighboring router is greater than or same as the FD of the local router to the destination network, it means that the neighboring router could route packets to the destination network via the local router. Considering the path via the neighboring router as a feasible successor (backup route) and immediately uses the path to forward packets upon the failure of the successor route would result in routing loop.

- Every neighbor that satisfies the relation of  $AD < FD$  for a particular destination network is a loop-free route to that destination. This feasibility condition is proven in Loop-free Routing using Diffusing Computations by Dr. J. J. Garcia-Lunes-Aceves published by IEEE on Feb 1993.
- Below shows an excerpt from the output of the **show ip eigrp topology EXEC** command. The FD (successor) to the 10.10.10.12/30 network is via Fa0/0. The route via S1/0 is considered a FS as the AD from 10.10.10.1 (30720) is less than the FD (284160) to the network.

```
P 10.10.10.12/30, 1 successors, FD is 284160
   via 10.10.10.10 (284160/281600), FastEthernet0/0
   via 10.10.10.1 (2174976/30720), Serial1/0
```

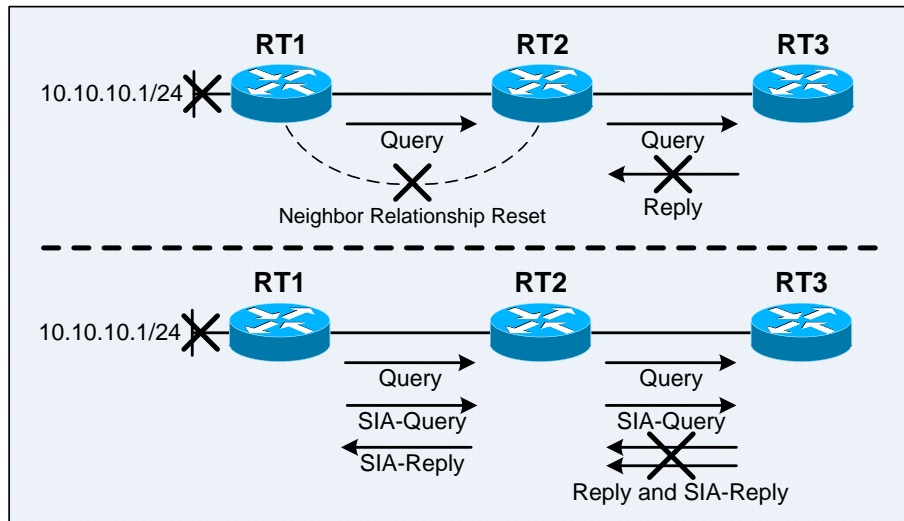
- Refer to Figure 5-6, RT3 is an FS to 10.10.10.0/24, as the AD through RT3 (1500) is less than the FD of the current successor, RT2 (2000).
- All routing protocols can insert only the next-hop router information in the routing table, information about the subsequent routers along the path can never appear in the routing table. Each router counts on the next-hop router to reach a destination network. Each router makes a path selection to reach a network and installs the best next-hop address to reach the network. A router trusts a successor (the best next-hop router) to forward traffic to the destination network.
- The routing table is a subset of the topology table. The topology table contains more detailed information about each route, backup routers, and information used exclusively by DUAL.
- When a router loses a route, it first looks at the topology table for an FS. If there is an FS, the route does not go into active state, and the best FS is promoted as the successor and inserted into the routing table – an FS is used immediately without any route computation. If there is no FS, a route goes into active state. A new successor will be determined through the route computation process, in which an EIGRP router will begin the process by sending queries to all its neighbors. The queried neighbors may also go through the same process, which creates a cascading of queries through the network to search the network for a path to the destination. The amount of route computation time affects the convergence time.

- An EIGRP router would perform the following tasks upon the receipt of a query for a route:
  - i) If the EIGRP topology table of the router does not contain an exact entry for the queried route, it immediately replies with a **network is unreachable** message to state that there is no path for the queried route through this neighboring router.
  - ii) If the EIGRP topology table of the router lists the querying router as the successor for the queried route and contains one or more FSs, the FS with the lowest metric will be installed into the routing table and the router immediately replies the information of the FS to the querying router.
  - iii) If the EIGRP topology table of the router lists the querying router as the successor for the queried route and does not contain a FS, the router propagates the query to all its neighboring routers except the querying router to seek for another alternative successor. The router will not reply to the querying router until it has received a reply for each query it propagated to its neighboring routers.
  - iv) If the querying router is not the successor for the queried route, the router immediately replies the information of its successor to the neighboring router.
- For each neighbor to which a query is sent, an EIGRP router will set a **reply status flag (r)** to keep track of all outstanding queries that are waiting for replies. The DUAL computation is completed when the router has received a reply for every query sent out earlier.

<pre>Router&gt;sh ip eigrp topology IP-EIGRP Topology Table for AS(100)/ID(10.10.10.1)  Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,        r - reply Status, s - sia Status</pre>	
<b>Passive (P)</b>	Indicates an available and stable network which can be installed into the routing table.
<b>Active (A)</b>	Indicates an unavailable network which has outstanding queries and cannot be installed into the routing table.
<b>Update (U)</b>	Indicates that a network is being updated and the router is waiting for an acknowledgment for the update packet.
<b>Query (Q)</b>	Indicates that a network has an outstanding query packet or the router is waiting for an acknowledgment for a query packet.
<b>Reply (R)</b>	Indicates that the router is generating a reply for the network or is waiting for an acknowledgment for the reply packet.
<b>Reply Status (r)</b>	Indicates that the router is waiting for the reply to a query packet from a neighboring router.

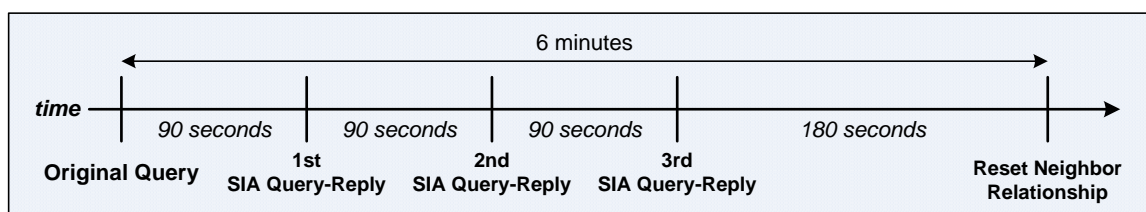
- At the beginning of the DUAL computation, an Active timer will be set for 3 minutes. If the expected reply for each outstanding query is not received before the Active timer expires, the route will enter into the **Stuck-in-Active (SIA)** state. The router will reset the neighbor relationships for a neighbor that failed to reply, and cause the router to go active on all routes known through the neighbor and re-exchange routing information with the neighbor.
- SIA should not occur in well-designed and stable networks. If a neighboring router or a link fails, it should be detected by the expiry of the hold timer instead of the expiry of the active timer. Such problems are often not the root cause of SIA.
- SIA may occur due to looping of queries, heavily congested and/or low bandwidth links, and/or overloaded routers in a large network. SIA may lead to flushing of valid neighbors and routes which would affect the stability of a network.

- The **EIGRP SIA Rewrite** or **Active Process Enhancement** feature was introduced to provide improved network reliability by preventing unintended resets of neighbor relationships between querying and queried routers due to the SIA conditions.



**Figure 5-7: EIGRP SIA Rewrite**

- Figure 5-7 shows the differences of the SIA conditions with and without the SIA Rewrite feature. When RT1 loses the route to 10.10.10.0/24, it would send a query for the network to RT2. RT2 would propagate the query to RT3 as it has no feasible successor for the network. As RT3 has no other neighbors to query, it would reply RT2 that it has no feasible successor. If RT2 never receives the reply from RT3 due to an errored link between RT2 and RT3, RT1 would reset the neighbor relationship between RT1 and RT2 after its Active timer expired. The question is why interrupts RT1 and RT2 when RT2 and RT3 are experiencing problems?
- With the SIA Rewrite feature, RT1 would query RT2 about the status of the route with a SIA-Query when the SIA-retransmit timer expires (90 seconds). RT2 would respond to the SIA-Query from RT1 with a SIA-Reply indicating that it is still searching for a new successor. Upon the receipt of SIA-Reply from RT2, RT1 would acknowledge the status of RT2 and reset the Active and SIA-retransmit timers. RT2 would also send a SIA-Query to RT3. If no SIA-Reply or reply to the original query for the active route is received from RT3 within the SIA-retransmit timer due to network link problem, RT2 would terminate the neighbor relationship with RT3 and inform RT1 that 10.10.10.0/24 is unreachable. RT1 and RT2 would remove the active route from their topology and routing tables and the neighbor relationship between them remains intact. **Note:** When a router starts querying for an active route, it would activate the Active timer (3 minutes) and SIA-retransmit timer, which is half of the value of the Active timer (90 seconds).
- Assuming that no reply to the original query is received, an EIGRP router will send up to 3 SIA-Query messages as long as SIA-Reply messages are received before resetting a neighbor. Hence as long as a neighbor router responds to every SIA-Query within the SIA-transmit timer, it won't be declared SIA and reset for 6 minutes, assuming a default Active timer of 180 seconds. This gives more time for large networks to respond to the query-reply processes for active routes.



**Figure 5-8: 3 SIA Query-Reply Cycles Maintain Neighbor Relationship up to 6 Minutes**

## EIGRP Reliability

- EIGRP reliability mechanism ensures the delivery of important routing information – Update, Query, and Reply packets, to neighboring routers in order to maintain a loop-free topology. A sequence number is assigned to every packet and requires an explicit acknowledgment for the sequence number. Acknowledgments are not necessarily sent via ACK packets, as the ACK field in any RTP unicast packet is sufficient to acknowledge the received EIGRP packets. For efficiency purpose, only certain EIGRP packets are being transmitted reliably.
- **Reliable Transport Protocol (RTP)** is responsible for guaranteed and ordered delivery of EIGRP packets with the use of sequence and acknowledge numbers, but without any fancy windowing or congestion control mechanism (because only one packet will be sent at a time). It supports transmission of both multicast and unicast packets.
- Sending Hello packets to all neighbors individually is inefficient on a multi-access networks that provide multicast capabilities (eg: Ethernet), therefore Hello packets do not require acknowledgement are sent as **unreliable multicasts**. All packets that carry routing information – Update, Query, and Reply packets are sent reliably and require explicit acknowledgment.  
**Note:** Hello and ACK packets which are not being transmitted reliably have no sequence number.
- RTP ensures ongoing communication is maintained between neighboring routers by maintaining/a retransmission list for each neighbor. The list is used to track all the reliable packets that were sent but not acknowledged within the **Retransmission Time Out (RTO)**. If the RTO timer expires before an ACK packet is received, EIGRP will transmit another copy of the reliable packet until the hold timer expires and terminate the neighbor relationship.
- The use of reliable multicast packets is efficient. However, delays are potential to exist on multi-access media with multiple neighbors, as a reliable multicast packet cannot be transmitted until all peers have acknowledged the previous multicast packet. If a router is slow to respond, it would delay the transmission of next packet and affects all other routers. RTP is designed to handle such situation – neighbors that respond slow to multicasts would have the unacknowledged multicast packets retransmitted as unicasts when the RTO timer expires. This allows the reliable multicast operation to proceed without delaying communications with others and ensure low convergence time in the environments with variable-speed links.
- The multicast flow timer determines how long to wait for an ACK packet before switching from multicast to unicast; while the RTO determines how long to wait between subsequent unicasts. The EIGRP process for each neighbor calculates both the multicast flow timer and RTO timer based on the **Smooth Round-Trip Time (SRTT)**. The formulas for the SRTT, RTO, and multicast flow timer are Cisco-proprietary.
- RTO is a dynamically adjusted over time. It is based on the SRTT, which specifies the average time in milliseconds between the transmission of a packet and the receipt of an acknowledgment. As more unacknowledged updates are sent, the SRTT would get higher and higher, which causes the RTO to increase exponentially. The maximum RTO value is 5000 ms (5 seconds).
- In a steady-state network where no routes are flapping, EIGRP waits for the specified hold-time interval to expire before determining that an EIGRP neighbor adjacency is down. By default, EIGRP waits up to 15 seconds for high-speed links and up to 180 seconds for low-speed links. When EIGRP determines that a neighbor is down and the router cannot reestablish the adjacency, the router will remove all reachable networks through that neighbor from the routing table. The router will attempt to find alternative paths to those networks when the convergence occurs.

- The 180-second hold time interval for low-speed links seems excessive, but it accommodates the links which are generally connected to less-critical remote sites. Additionally, 15 seconds is considered too long for some networks with high-speed links and serving mission-critical and time-sensitive applications. Always remember that there are situations and reasons in which modifying the default hold time interval is necessary in order to achieve faster convergence.
- When a local router sends an update packet to a remote router and the remote router does not acknowledge the packet, the router would retransmit the update every time the RTO expires for up to 16 times, or until the hold timer expires, whichever is longer, and eventually terminate the neighbor relationship; not as claimed by some sources that the neighbor relationship will be terminated after 16 retransmissions rather than wait until the hold timer expires!
- EIGRP packets are only being generated at the moment of transmission. The transmit queues contain small and fixed-size structures that indicate which parts of the topology table to include in an EIGRP packet. As a result, the queues do not consume large amounts of memory and only the latest information will be transmitted. Ex: If a route changes state several times, only the last state is transmitted in the packet. This approach reduces bandwidth utilization.
- The **show ip eigrp traffic EXEC** command shows the numbers EIGRP packets sent and received on a router.

```

Router#sh ip eigrp traffic
IP-EIGRP Traffic Statistics for AS 100
  Hellos sent/received: 26/22
  Updates sent/received: 3/3
  Queries sent/received: 0/0
  Replies sent/received: 0/0
  Acks sent/received: 2/0
  Input queue high water mark 2, 0 drops
  SIA-Queries sent/received: 0/0
  SIA-Replies sent/received: 0/0
  Hello Process ID: 72
  PDM Process ID: 71

Router#

```

## Chapter 6

# EIGRP Lab

- Below are the steps for configuring EIGRP for IP:
  - 1) Enable EIGRP and define the autonomous system with the **router eigrp** {*as-num*} global configuration command. The *as-num* value must be matched on all EIGRP routers within an autonomous system or an EIGRP routing domain.
  - 2) Specify the networks that belong to the EIGRP autonomous system with the **network** {*ip-addr*} [*wildcard-mask*] router subcommand. The *ip-addr* can be either the network address, the subnet, or the address of an interface. It determines which links to advertise routing updates, which links to listen and process routing updates, and which networks to be advertised in routing updates. The *wildcard-mask* is an **optional** inverse mask used to determine how to interpret the *ip-addr*. It contains wildcard bits, where 0 is a match and 1 is don't care. Ex: 0.0.0.255 indicates an exact match in the first 3 bytes.

**Note:** If the optional *wildcard-mask* is not being specified, the EIGRP process assumes that all directly connected networks that are part of the major network will participate in the EIGRP routing process, and EIGRP will attempt to establish EIGRP neighbor relationship for each interface that is part of the Class A, B, or C network. Use the *wildcard-mask* to identify an IP address, a subnet, or a network. Ex: Use the mask 0.0.0.0 if specifying an interface to be participated in the EIGRP routing process. The combination of 0.0.0.0 255.255.255.255 matches all interfaces on a router.

**Note:** The **network 0.0.0.0** EIGRP router subcommand is equivalent to the **network 0.0.0.0 255.255.255.255** EIGRP router subcommand. However, configuring the combination of **0.0.0.0 0.0.0.0** will be automatically changed to **0.0.0.0 255.255.255.255**.
  - 3) When configuring EIGRP on WAN links, remember to define the appropriate bandwidth value for the links for EIGRP to perform metric and route computations precisely. For serials interfaces (eg: PPP and HDLC), set the bandwidth to match the line speed. For Frame Relay point-to-point interfaces, set the bandwidth of the subinterfaces to their corresponding provisioned CIRs. For Frame Relay multipoint interfaces, set the bandwidth to the sum of all CIRs. The **bandwidth** {*kbits*} interface subcommand defines the bandwidth of a particular interface. The **bandwidth** and **delay** interface subcommands only configures informational parameters for EIGRP to make better metric calculation, they are unable to adjust the actual bandwidth and delay of an interface to achieve better performance. It is recommended to use the **bandwidth** command over the **delay** command whenever possible, because it is must easier to get the desired result.

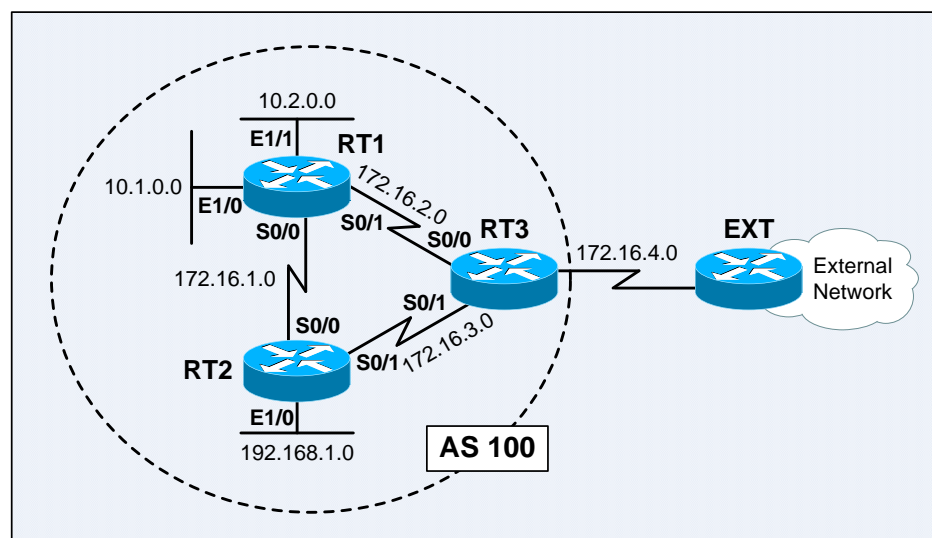


Figure 6-1: Sample EIGRP Network



- EIGRP configuration on RT1, RT2, and RT3:

```

RT1 (config) #router eigrp 100
RT1 (config-router) #network 10.1.0.0
RT1 (config-router) #network 10.2.0.0
RT1 (config-router) #network 172.16.1.0
RT1 (config-router) #network 172.16.2.0
RT1 (config-router) #no auto-summary
RT1 (config-router) #^Z
RT1 #sh run | b router eigrp
router eigrp 100
  network 10.0.0.0
  network 172.16.0.0
  no auto-summary
!
-----

RT2 (config) #router eigrp 100
RT2 (config-router) #network 172.16.1.0 0.0.0.255
RT2 (config-router) #network 172.16.3.0 0.0.0.255
RT2 (config-router) #network 192.168.1.0
RT1 (config-router) #no auto-summary
RT2 (config-router) #^Z
RT2 #sh run | b router eigrp
router eigrp 100
  network 172.16.1.0 0.0.0.255
  network 172.16.3.0 0.0.0.255
  network 192.168.1.0
  no auto-summary
!
-----

RT3 #sh run | b router eigrp
router eigrp 100
  network 172.16.2.0 0.0.0.255
  network 172.16.3.0 0.0.0.255
  no auto-summary
!

```

**Note:** The **network** {*classful-network-addr*} router subcommand is a classful command, it will change the network number entered along with the command to its **classful entry** or **major network number**. Ex: 10.1.1.0 → 10.0.0.0; 172.16.1.0 → 172.16.0.0.

- Wildcard mask is not used in RT1 EIGRP configuration. All RT1 interfaces that are part of network 10.0.0.0/8 and network 172.16.0.0/16 are participating in the EIGRP routing process. **Note:** Network 192.168.1.0/24, a RT2's directly connected network, is not configured in RT1, as RT1 does not have any interface resides in that network.
- RT2 EIGRP configuration uses wildcard mask to determine which directly connected interfaces to participate in the EIGRP routing process. In this case, all the 3 RT2 interfaces are participating in the EIGRP routing process.
- RT3 EIGRP configuration also uses the wildcard masks, as it connects to an external router, and EIGRP should not run over there. RT3 has 3 interfaces of 172.16.0.0 Class B network. Without using wildcard mask, RT3 would send EIGRP to the external network, which would waste bandwidth and CPU resources as well as provide unnecessary information to the network. With the configuration above, RT3 would only establish a relationship with EIGRP routers from interfaces that are part of network 172.16.2.0/24 and 172.16.3.0/24.

## EIGRP Default Routing

- The EIGRP default route can be created with the **ip default-network** {*network-number*} global configuration command. An EIGRP router which is configured with this command will consider the *network-number* as the gateway of last-resort and will announce to other routers. The network must be reachable by the router that is configured with this command before it is announced as a candidate default route to other routers; which means that the network must either be an EIGRP-derived network in the routing table or generated with a static route and redistributed into EIGRP. The network number will be propagated to other routers so that they can utilize this network as the default network and eventually the gateway of last-resort.
- In a complex topology where multiple default networks are configured as candidate default, downstream routers will use the EIGRP metric to dynamically determine the best default route, rather than requiring manually configured static routes.

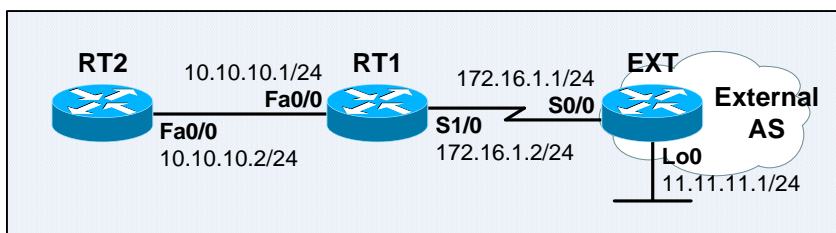


Figure 6-2: Network Setup for EIGRP Default Routing

- EIGRP Default Network configuration on RT1:

```
!  
router eigrp 100  
  network 10.0.0.0  
  network 172.16.0.0  
  auto-summary  
!  
ip default-network 172.16.0.0  
ip route 172.16.0.0 255.255.0.0 172.16.1.1  
!  
RT1#sh ip route  
  
Gateway of last resort is 172.16.1.1 to network 172.16.0.0  
  
* 172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks  
S* 172.16.0.0/16 [1/0] via 172.16.1.1  
C 172.16.1.0/24 is directly connected, Serial1/0  
C 10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks  
C 10.10.10.0/24 is directly connected, FastEthernet0/0  
D 10.0.0.0/8 is a summary, 00:06:04, Null0  
RT1#
```

**Note:** The **auto-summary** EIGRP router subcommand is required for this sample to work.

**Note:** For EIGRP to propagate the route, the network specified by the **ip default-network** command must be known to EIGRP. This means that the network must be an EIGRP route in the routing table, or the static route that generates the route to the network must be redistributed into EIGRP, or advertised into EIGRP using the **network** command.

- Below shows the routing table on RT2. The default network is advertised from RT1 to RT2.

```

RT2#sh ip route

Gateway of last resort is 10.10.10.1 to network 172.16.0.0

D* 172.16.0.0/16 [90/2172416] via 10.10.10.1, 00:05:12, FastEthernet0/0
    10.0.0.0/24 is subnetted, 1 subnets
C    10.10.10.0 is directly connected, FastEthernet0/0
RT2#
RT2#ping 11.11.11.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 11.11.11.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 60/79/100 ms
RT2#

```

- RT1 is configured with the 172.16.0.0/16 network as a candidate default network using the **ip default-network 172.16.0.0** global configuration command. RT1 also has that network configured with the **network** router subcommand, and therefore will advertise it to RT2. RT1 routing table does not set the gateway of last resort – the **ip default-network** command has no effect on RT1. On RT2, the 172.16.0.0/16 network learnt via EIGRP is flagged as a candidate default network as indicated by the \* besides the candidate default route. RT2 also sets its gateway of last resort as 10.10.10.1 (RT1) to reach the default network 172.16.0.0.
- The **ip default-network** command is classful. When the network specified with this command is a subnet of a major network (eg: 10.10.0.0, 172.16.1.0), the router will configure a static route to the major network number with the **ip route** command instead of configuring a default network with the **ip default-network** command. The static route is configured without the notice of the user, as no message will be displayed. Deleting this route requires the **no ip default-network** command instead of the **no ip route** command!
- EIGRP propagates default route differently than RIP when using the **ip route 0.0.0.0 0.0.0.0** command. EIGRP does not redistribute the 0.0.0.0 0.0.0.0 default route by default. However, if the **network 0.0.0.0** EIGRP router subcommand is configured, EIGRP will redistribute a default route as a result of the **ip route 0.0.0.0 0.0.0.0 {outgoing-intf}** global command, but not as a result of the **ip route 0.0.0.0 0.0.0.0 next-hop-addr** or **ip default-network** command. Below shows the alternative configuration on RT1 for it to propagate the 0.0.0.0 default route information to RT2. RT1 and RT2 are able to ping 11.11.11.1.

```

RT1 configuration:
!
router eigrp 100
 network 0.0.0.0
  auto-summary | no auto-summary
!
ip route 0.0.0.0 0.0.0.0 Serial1/0
!
-----
RT2#sh ip route | in Gateway|D[*]
Gateway of last resort is 10.10.10.1 to network 0.0.0.0
D* 0.0.0.0/0 [90/2172416] via 10.10.10.1, 00:06:25, FastEthernet0/0
RT2#

```

- The best solution is simply configure and redistribute a static default route into EIGRP on RT1. 🤖

## EIGRP Route Summarization

- Summarizing routes automatically up to the major network boundary is an EIGRP feature that inherited from traditional distance vector routing protocols (the predecessor of EIGRP is IGRP, a DV routing protocol). Traditional DV routing protocols are classful routing protocols that must summarize at the network boundaries. They have no idea regarding the subnet masks for non-directly connected networks, as subnet masks are not being exchanged in routing updates.
- EIGRP performs autosummarization by default. This feature can be disabled and must be disabled for **discontiguous networks**. Additionally, EIGRP summary routes within a network can be configured on any bit boundary as long as a more-specific route exists in the routing table. EIGRP supports summarization at any router; unlike OSPF, which must be performed on an Area Border Router (ABR) or Autonomous System Border Router (ASBR).
- When route summarization is configured on an interface, a summary route destined to the **Null0 interface**, a directly connected, software-based interface as the next-hop interface will be added to the routing table. The Null0 interface is often used to prevent routing loop in a network. Ex: If the summarizing router receives a packet to an unknown subnet that is part of the summarized range, the packet matches the summary route based on the longest match. The packet is forwarded to the Null0 interface and discarded. This prevents the summarizing router from forwarding the packet to a default route which could possibly create a routing loop. The null route will only be seen on the summarizing router.
- The **ip summary-address eigrp {as-num} {network-address} {summary-mask}** interface subcommand creates a summary route. The summary route is advertised only if an internal component route (a more-specific entry) of the summary route is exists in the routing table of the summarizing router <sup>[1]</sup>. The component route does not need to be a directly connected subnet; it may be learnt from other routers via a routing protocol.  
[1] – This means that if all components that make up the summary disappear, or only external (redistributed) components exist, the summary route is not installed and advertised.
- IP EIGRP summary routes have an administrative distance value of 5. Administrative distance values are **locally significant** and therefore are not propagated to other routers; the neighbor routers which receive the summary route still have an Internal EIGRP route with an AD of 90. The EIGRP summary route with an AD of 5 will only be shown on the summarizing router with the **show ip route {summary-route} EXEC** command.

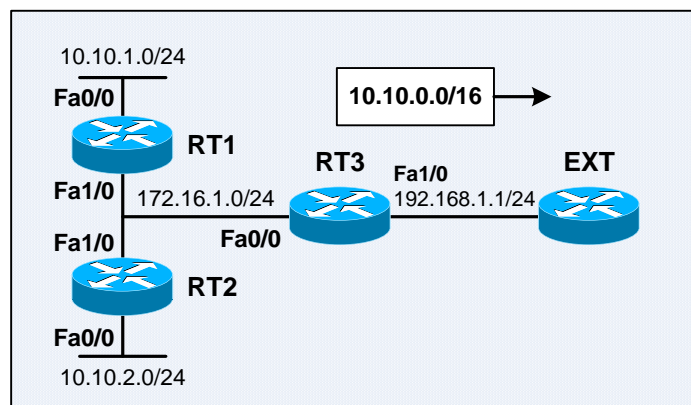


Figure 6-3: Network Setup for EIGRP Route Summarization

- Autosummarization on has been disabled RT1 and RT2 with the **no auto-summary** router subcommand, therefore RT3's routing table contains all the subnets on RT1 and RT2. Configuring a summary route on RT3 Fa0/0 consolidates the 10.1.1.0/24 and 10.1.2.0/24 routes and advertises a single 10.1.0.0/16 summary route to EXT.
- Below shows the EIGRP route summarization configuration on RT3 and routing table of EXT:

```
EXT#sh ip route
```

```
Gateway of last resort is not set
```

```

    172.16.0.0/24 is subnetted, 1 subnets
D       172.16.1.0 [90/30720] via 192.168.1.1, 00:00:35, FastEthernet0/0
    10.0.0.0/24 is subnetted, 2 subnets
D       10.10.1.0 [90/33280] via 192.168.1.1, 00:00:35, FastEthernet0/0
D       10.10.2.0 [90/33280] via 192.168.1.1, 00:00:35, FastEthernet0/0
C       192.168.1.0/24 is directly connected, FastEthernet0/0
EXT#
```

```
RT3#conf t
```

```
Enter configuration commands, one per line.  End with CNTL/Z.
```

```
RT3(config)#int fa1/0
```

```
RT3(config-if)#ip summary-address eigrp 100 10.10.0.0 255.255.0.0
```

```
RT3(config-if)#
```

```
00:06:14: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 192.168.1.2
(FastEthernet1/0) is down: summary configured
```

```
00:06:15: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 192.168.1.2
(FastEthernet1/0) is up: new adjacency
```

```
RT3(config-if)#^Z
```

```
RT3#sh ip route
```

```
Gateway of last resort is not set
```

```

    172.16.0.0/24 is subnetted, 1 subnets
C       172.16.1.0 is directly connected, FastEthernet0/0
    10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
D       10.10.0.0/16 is a summary, 00:05:22, Null0
D       10.10.1.0/24 [90/30720] via 172.16.1.1, 00:06:42, FastEthernet0/0
D       10.10.2.0/24 [90/30720] via 172.16.1.2, 00:06:42, FastEthernet0/0
C       192.168.1.0/24 is directly connected, FastEthernet1/0
RT3#
```

```
RT3#sh ip route 10.10.0.0
```

```
Routing entry for 10.10.0.0/16
```

```
Known via "eigrp 100", distance 5, metric 30720, type internal
```

```
Redistributing via eigrp 100
```

```
Routing Descriptor Blocks:
```

```
* directly connected, via Null0
```

```
Route metric is 30720, traffic share count is 1
```

```
Total delay is 200 microseconds, minimum bandwidth is 100000 Kbit
```

```
Reliability 255/255, minimum MTU 1500 bytes
```

```
Loading 1/255, Hops 1
```

```
RT3#
```

```
EXT#sh ip route
```

```
Gateway of last resort is not set
```

```

    172.16.0.0/24 is subnetted, 1 subnets
D       172.16.1.0 [90/30720] via 192.168.1.1, 00:00:15, FastEthernet0/0
    10.0.0.0/16 is subnetted, 1 subnets
D       10.10.0.0 [90/33280] via 192.168.1.1, 00:00:15, FastEthernet0/0
C       192.168.1.0/24 is directly connected, FastEthernet0/0
EXT#
```

## EIGRP Load Balancing

- Load balancing is a router capability to distribute traffic over multiple network paths that have the same metric to a destination network.
  - Good load balancing algorithms utilize both line speed and reliability information. Load balancing increases the utilization of network links and eventually increases the effective bandwidth and throughput.
  - Cisco IOS perform load balancing across of 4 equal-cost paths by default. The **maximum-paths** *{max-paths-num}* router subcommand defines the number of same metric parallel paths to a same destination network that can be added to the routing table and perform load balancing across the links in round-robin basis. Set *max-paths-num* to 1 to disable load balancing. **Note:** When packets are process-switched, load balancing occurs on a per-packet basis. When packets are fast-switched, load balancing occurs on a per-destination basis.
  - **Note:** Load balancing is performed only on traffic that passes through a router, not traffic that originated by the router. Hence when testing load balancing, do not issue ping and observe the results from the router that connects to the same metric parallel paths, as the locally generated ping packets are process-switched rather than fast-switched and might produce confusing results.
  - EIGRP and IGRP are able to perform load balancing across multiple routes with different metrics – **unequal-cost load balancing**. The **variance** *{multiplier}* router subcommand defines the variance value (between 1 and 128) which defines routes with metric value less than or equal to **lowest metric × variance multiplier** are considered equal and will be used for load balancing. The default is 1, which means equal cost load balancing. Ex: With a lowest metric of 100 and variance of 2, all routes with a metric  $\leq 200$  are considered equal.
  - Only paths that meet the **feasibility condition** can be used for unequal-cost load balancing. A path or route can only be considered as a feasible successor when:
    - The advertised distance from the next router (the feasible distance of the next router) must be less than the lowest-metric (feasible distance) of the local router to the destination. This means that the next router in the path must be closer to the destination than the local router in order to prevent routing loops.
    - The feasible distance through the next router must be less than or equal to the lowest metric (feasible distance) of the local router multiplied with the variance multiplier.
- \* If a path isn't a feasible successor, it isn't used in load balancing, even if variance is configured!

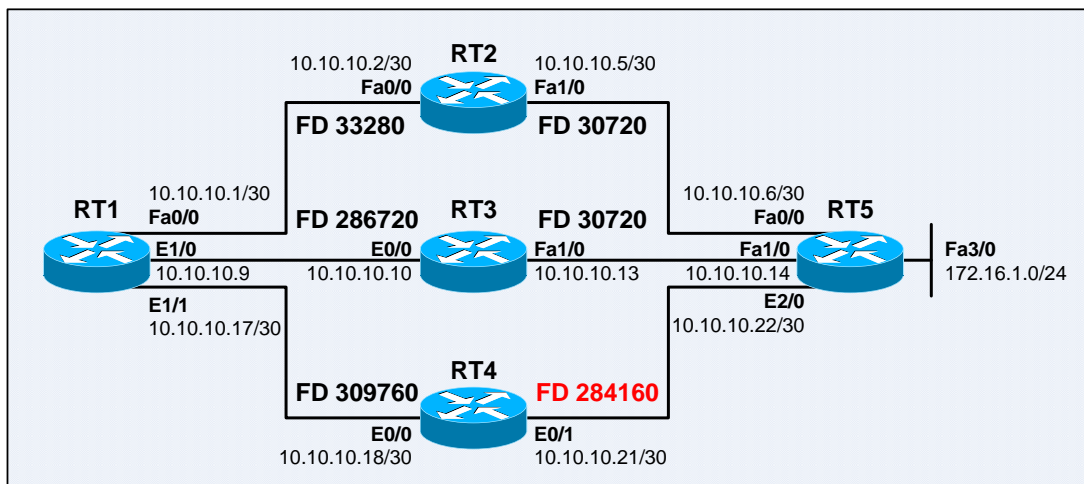


Figure 6-4: Network Setup for EIGRP Load Balancing

- Below shows that after changing the variance multiplier to 10 on RT1, the path to 172.16.1.0/24 via RT4 is not considered feasible, even the FD via RT4 – 309760 is less than lowest metric multiplied with the variance multiplier – 332800; because the AD of RT4 – 284160, is not less than the FD of RT1 to 172.16.1.0/24 – 33280.

```

RT1#sh ip route

Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 1 subnets
D       172.16.1.0 [90/33280] via 10.10.10.2, 00:01:00, FastEthernet0/0
    10.0.0.0/30 is subnetted, 6 subnets
C       10.10.10.8 is directly connected, Ethernet1/0
D       10.10.10.12 [90/33280] via 10.10.10.2, 00:01:00, FastEthernet0/0
C       10.10.10.0 is directly connected, FastEthernet0/0
D       10.10.10.4 [90/30720] via 10.10.10.2, 00:01:00, FastEthernet0/0
C       10.10.10.16 is directly connected, Ethernet1/1
D       10.10.10.20 [90/286720] via 10.10.10.2, 00:01:00, FastEthernet0/0
RT1#
RT1#sh ip eigrp topology all-links | b 172.16.1.0
P 172.16.1.0/24, 1 successors, FD is 33280, serno 10
   via 10.10.10.2 (33280/30720), FastEthernet0/0
   via 10.10.10.18 (309760/284160), Ethernet1/1
   via 10.10.10.10 (286720/30720), Ethernet1/0
RT1#
RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#router eigrp 100
RT1(config-router)#variance 10
RT1(config-router)#^Z
RT1#sh ip route

Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 1 subnets
D       172.16.1.0 [90/33280] via 10.10.10.2, 00:00:10, FastEthernet0/0
           [90/286720] via 10.10.10.10, 00:00:10, Ethernet1/0
    10.0.0.0/30 is subnetted, 6 subnets
C       10.10.10.8 is directly connected, Ethernet1/0
D       10.10.10.12 [90/33280] via 10.10.10.2, 00:00:10, FastEthernet0/0
           [90/284160] via 10.10.10.10, 00:00:10, Ethernet1/0
C       10.10.10.0 is directly connected, FastEthernet0/0
D       10.10.10.4 [90/30720] via 10.10.10.2, 00:00:10, FastEthernet0/0
C       10.10.10.16 is directly connected, Ethernet1/1
D       10.10.10.20 [90/286720] via 10.10.10.2, 00:00:10, FastEthernet0/0
           [90/307200] via 10.10.10.18, 00:00:10, Ethernet1/1
           [90/309760] via 10.10.10.10, 00:00:10, Ethernet1/0
RT1#

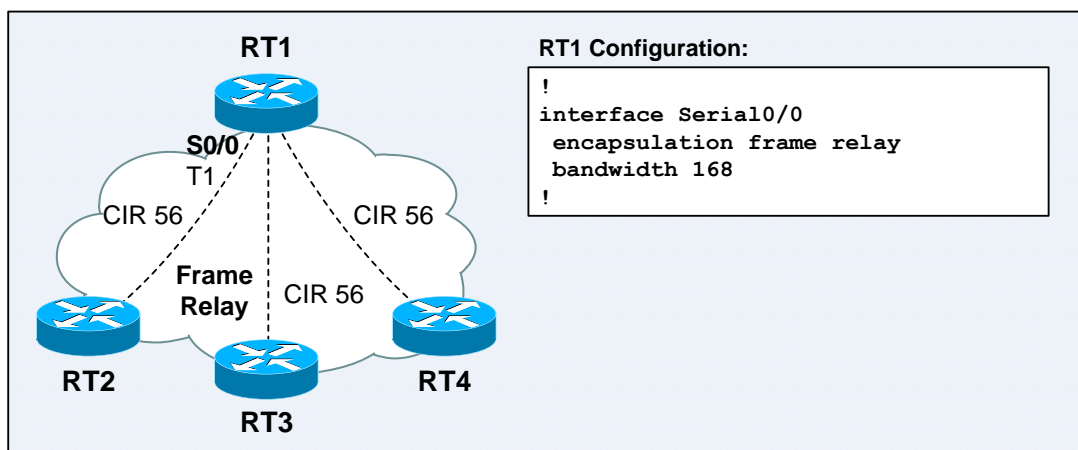
```

**Tips:** Always write out the FD and AD of all routes in a network diagram before deciding whether the **variance** command allows the use of all paths.

- The **traffic-share {balanced | min across-interfaces}** router subcommand controls how traffic is being distributed among routes when multiple different costs routes exist for the same destination network – **traffic sharing**. With the **balanced** keyword, the router distributes traffic proportionately to the ratios of the metrics associated with the different routes. With the **min across-interfaces** keyword, the router uses only routes that have minimum costs – all routes that are feasible and within the variance are maintained in the routing table, but only those with the minimum cost will be used for forwarding packets.

## EIGRP and WAN Links

- EIGRP operates efficiently in WAN environments. EIGRP is scalable on both point-to-point links and NBMA multipoint and point-to-point links, eg: X.25, Frame Relay, and ATM. Due to the inherent differences of the operational characteristics, the default configuration of WAN connections might not be optimal. A solid understanding of EIGRP operation and the knowledge of link speeds can produce an efficient, reliable, scalable router configuration.
- By default, EIGRP is allowed to use up to 50% of the bandwidth of an interface or subinterface. EIGRP uses the default bandwidth of the link, or the bandwidth set by the **bandwidth** interface subcommand when calculating how much bandwidth to use.
- The EIGRP **pacing** feature allows the fine-tuning of the bandwidth usage of a link for transmitting EIGRP traffic. The **ip bandwidth-percent eigrp {as-num} {percent}** interface subcommand adjusts the EIGRP link utilization for an [sub]interface based on the interface's bandwidth value. The *percent* value is ranging from 1 to 999999. A *percent* value greater than 100 is **reasonable** if the bandwidth is configured artificially low for some routing policy reasons or to accommodate an oversubscribed multipoint Frame Relay interface.  
Ex: A 512kbps interface is configured with a logical bandwidth of 56kbps. 200% is configured when 112kbps (2 x 56) is allocated for EIGRP traffic.  
**Note:** Make sure that the physical link is provisioned to handle the configured capacity.
- Cisco IOS treats Frame Relay point-to-point subinterfaces in the same manner as serial interfaces in terms of bandwidth. It assumes both of them are operating at full T1 link speed. However, only fractional T1 speeds are available in most implementations. Therefore, always remember to set the bandwidth to match the provisioned CIR when configuring these subinterfaces.
- When configuring Frame Relay multipoint interfaces, remember that the bandwidth is shared equally by all neighbors – EIGRP uses the bandwidth of the physical interface divided by the number of Frame Relay neighbors connected on the physical interface to obtain the bandwidth attributed to each neighbor. EIGRP configuration should be fine-tuned to reflect the appropriate percentage of the actual available bandwidth for the link.



**Figure 6-5:** Frame Relay Multipoint with all VCs Share the Bandwidth Evenly

- Figure 6-5 shows a Frame Relay network where RT1 has a Frame Relay multipoint interface and all VCs share the bandwidth evenly. The RT1 Frame Relay multipoint interface is configured for a bandwidth of 168kbps (3 x 56kbps CIR).



- The sample network in Figure 6-6 have the same configuration as for Figure 6-5, where one of the circuits has been provisioned for a 56kbps CIR, and the other circuits have a higher CIR. RT1 Frame Relay multipoint interface is configured for a bandwidth that represents the lowest CIR multiplied by the number of circuits –  $56 \times 3 = 168$ . This approach might not fully utilize the higher-speed circuits, but it protects the circuit with the lowest CIR from being overwhelmed.

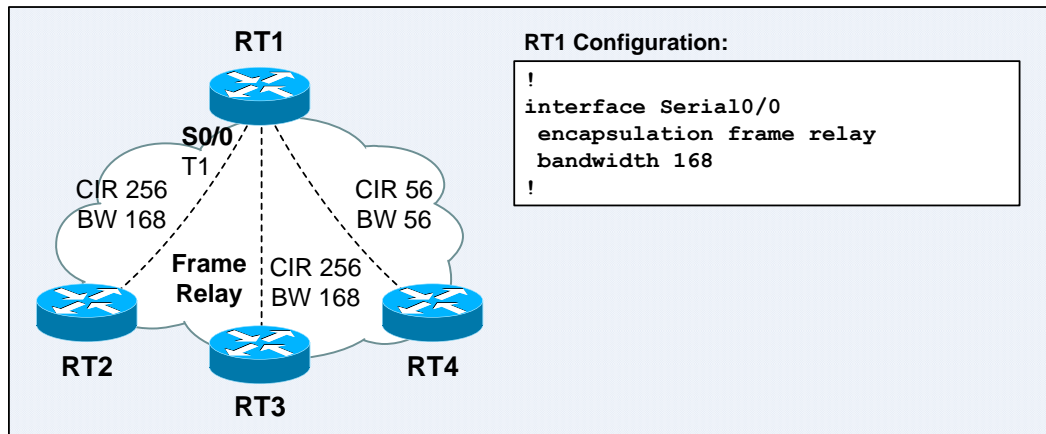


Figure 6-6: Frame Relay Multipoint with all VCs Have Different CIRs

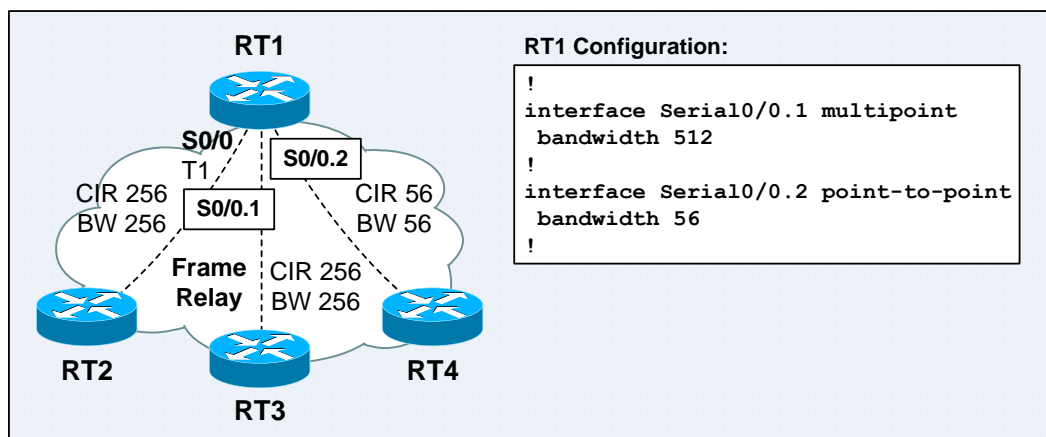


Figure 6-7: Hybrid Frame Relay Network with Multipoint and Point-to-Point

- Figure 6-7 shows a hybrid Frame Relay network, where there is 1 low-speed circuit and other VCs provisioned for a higher CIR. The lowest CIR VC is configured as point-to-point with the bandwidth equals to CIR; the higher CIR VCs are configured as multipoint with combined CIRs.

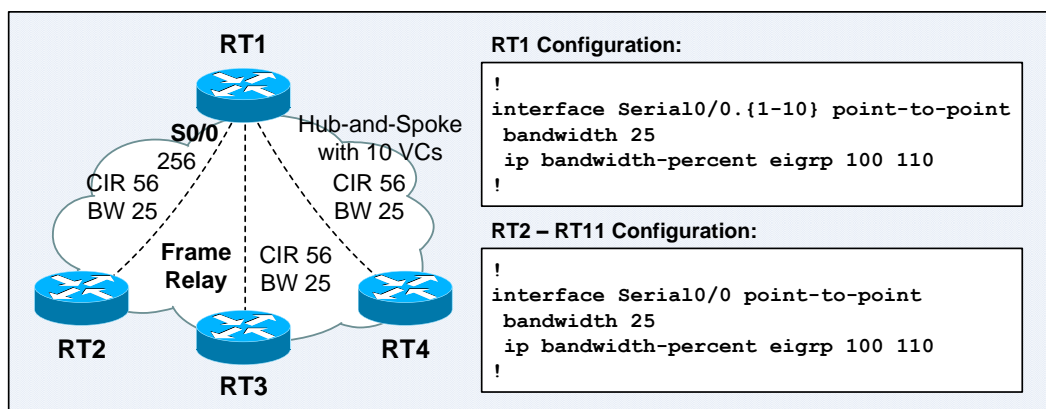


Figure 6-8: Frame Relay Hub-and-Spoke Network

- Figure 6-8 shows a common oversubscribed Frame Relay hub-and-spoke topology with 10 VCs to the remote sites (only 3 remote sites are shown in the figure). The circuits are provisioned with 56kbps CIR, but there is insufficient bandwidth at RT1 interface to support the allocation.
- All VCs are treated equally and are configured for about 1/10 (25kbps) of the RT1 link speed, and EIGRP link utilization is increased to 50% of the actual point-to-point VC capacity (56kbps). By default, EIGRP uses 50% of the bandwidth of a circuit. EIGRP configuration should reflect the correct percentage of the actual available bandwidth on the line. Therefore, each subinterface has the EIGRP allocation percentage raised up to 110% (~28kbps) of the specified bandwidth in order to ensure enough bandwidth to deliver EIGRP packets through the Frame Relay network. This configuration restores the 50-50 ratio when the bandwidth value was set artificially low for some reasons, eg: manipulation of the routing metric or to accommodate an oversubscribed multipoint Frame Relay configuration on the hub router interface.

**Note:**  $\frac{50}{100} \times 56\text{kbps} = 28\text{kbps}$      $\frac{110}{100} \times 25\text{kbps} = 27.5\text{kbps}$

- The **ip bandwidth-percent** command relies upon the value set by the **bandwidth** command.
- EIGRP can further conserve WAN bandwidth by suppressing ACKs. An ACK will not be sent if a unicast data packet is ready for transmission, as the ACK field in any RTP unicast packet is sufficient to acknowledge the EIGRP packets received from a neighboring router, therefore the ACK packet will be suppressed to conserve bandwidth. This is an essential feature for point-to-point links and NBMA networks, as on those environments, all data packets are sent as unicasts, and hence can carry an acknowledgement themselves (also known as **piggyback ACK**). In those environments, there is no need for ACK packets.
- The **show ip eigrp interfaces detail [intf-type intf-num]** EXEC command shows the numbers of ACK packets suppressed for a particular interface.

```
Router#sh ip eigrp interfaces detail s0/0
IP-EIGRP interfaces for process 100

          Xmit Queue   Mean   Pacing Time   Multicast
Pending
Interface      Peers  Un/Reliable  SRTT    Un/Reliable   Flow Timer
Routes
Se0/0          0      0/0          0       0/10         0          0
  Hello interval is 5 sec
  Next xmit serial <none>
  Un/reliable mcasts: 0/0  Un/reliable ucasts: 0/0
  Mcast exceptions: 0  CR packets: 0  ACKs suppressed: 0
  Retransmissions sent: 0  Out-of-sequence rcvd: 0
  Authentication mode is not set
Router#
```

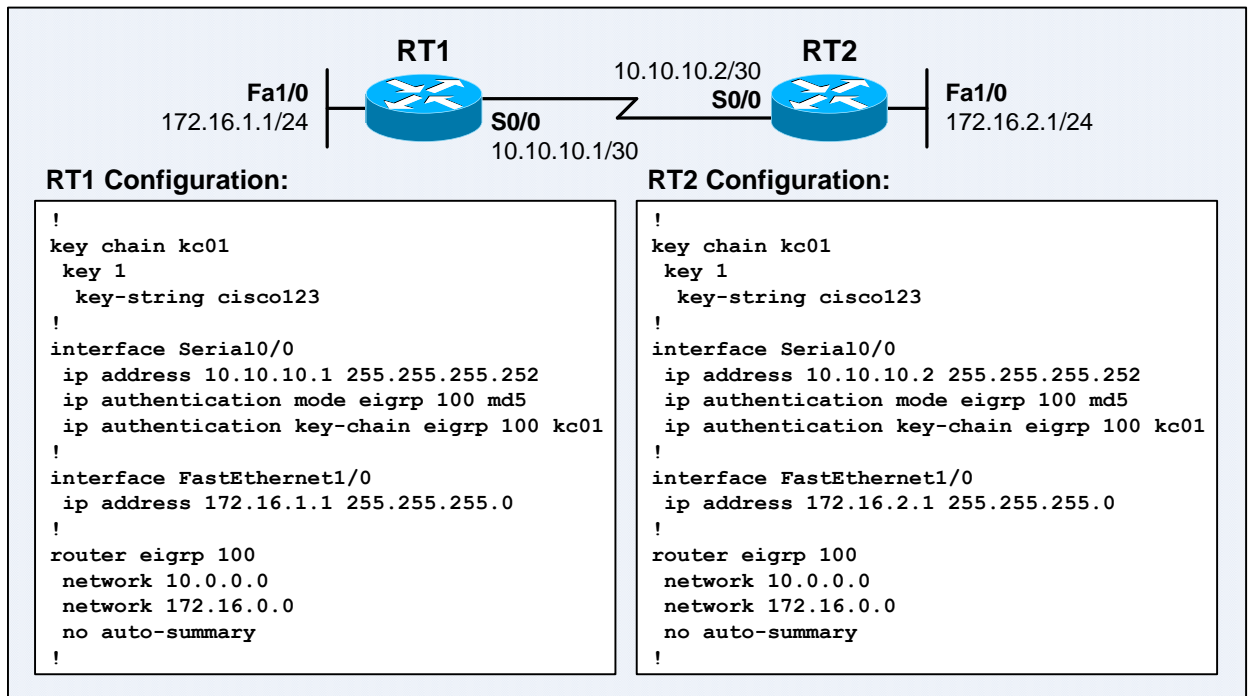
## EIGRP Authentication

- EIGRP neighbor authentication (also known as **neighbor router authentication** or **route authentication**) configures EIGRP routers to only participate in routing domain based on predefined passwords for preventing an EIGRP router from receiving unauthorized or fraudulent routing updates from unknown sources. An EIGRP router configured with neighbor authentication will authenticate the source of all types of EIGRP packets except ACK packets.
- Cisco IOS supports the following types of authentication for common routing protocols:

<b>Simple password authentication</b>	Also known as plain text authentication. Sends the authentication key over across the network and therefore vulnerable to passive attacks. Supported by RIPv2, OSPF, and IS-IS.
<b>Message Digest 5 (MD5) authentication</b>	Sends a message digest or hash instead of the authentication key. The message digest or hash is appended to routing update packets. Supported by RIPv2, EIGRP, OSPF, and BGP.

**Note:** EIGRP only supports the MD5 authentication method.

- Authentication keys can be managed using **key chains**. A key defined within a key chain can specify a time interval for which the key will be activated, referred to as the **lifetime** of the key. Routing update packets will be sent with the valid or activated key based on the lifetime of a key. The 1st valid key that is encountered in the series of keys with the lowest to highest key ID number will be used at a time regardless of the number of valid keys. The key ID numbers do not need to be consecutive. However, at least 1 key must be defined within a key chain.



**Figure 6-9:** EIGRP MD5 Authentication Configuration

- The **accept-lifetime** *{start-time} {infinite | end-time | duration secs}* and the **send-lifetime** *{start-time} {infinite | end-time | duration secs}* key chain key configuration subcommands are optional for specifying the time period in which a particular key will be accepted for received packets and used for sending packets respectively. When intended to set lifetimes on keys, ensure that the time of network devices are synchronized via NTP.

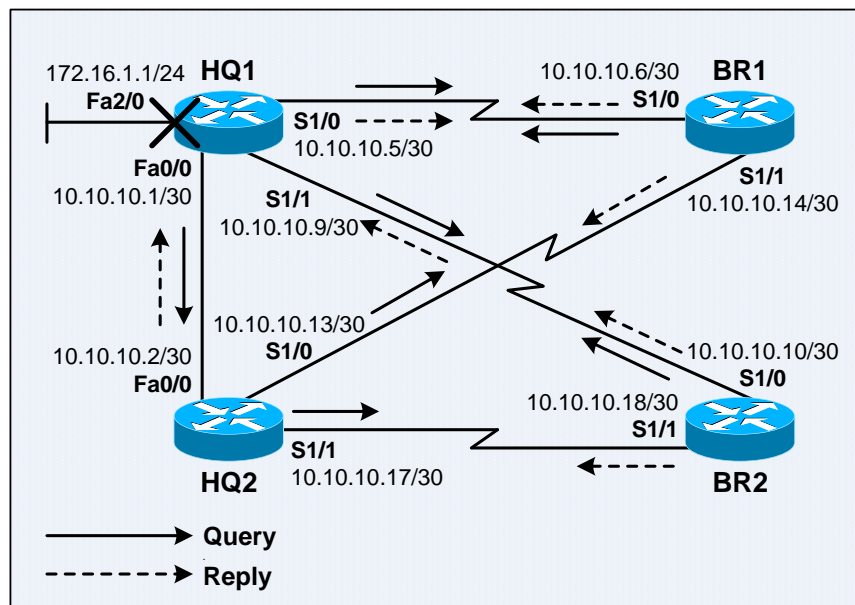
## Scaling EIGRP Internetworks

- Below are some variables that affect network scalability:
  - i) **The amount of information exchanged between neighbors** – The more information to be exchanged between EIGRP routers, the more work the routers need to perform upon a topology change.
  - ii) **The number of routers** – The more EIGRP routers that must be involved in a topology change, the more resources will be consumed by the routers upon a topology change.
  - iii) **The depth of the topology** – Depth is referred to as the number of hops that information must travel to reach all routers. An organization network without route summarization often has a large depth and would increase the convergence time upon a topology change. A 3-tier network design is highly recommended for all IP routing environments. There should never be more than 7 hops between any 2 routing devices on an internetwork. The propagation delay and the query process across multiple hops would increase the convergence time upon a topology change.
  - iv) **The number of alternative paths** – A network should be built with alternative paths to avoid single points of failure. However, EIGRP convergence would take longer duration to complete for networks with too many alternative paths, as EIGRP needs to perform query to explore all possible paths for lost routes without feasible successors. Such complexity often causes routers to become **stuck-in-active** as they await responses to queries that are being propagated through other EIGRP routers on the alternative paths.
- As an advanced DV routing protocol, EIGRP inherits the routing by rumor feature from traditional DV routing protocols. EIGRP relies on its neighbors to provide routing information. An EIGRP router would query its neighbors when a route is lost and no FS is available.
- Summarization in the 3-tier hierarchical network model provides the following benefits:
  - i) Summarized routes at **the core** reduce the size of the routing tables of the core routers. Smaller routing tables allow efficient lookups and hence provide a fast-switching core.
  - ii) Summarized routes at **the regional head offices** reduce the number of topology table entries and hence provide faster convergence upon the alternative paths.
  - iii) Proper and well-planned allocation of addressing blocks to **the remote offices** allows local traffic to remain local and does not burden other portions of the network.

## EIGRP Query-Reply Process and Stuck-in-Active Routes

- When an EIGRP router loses a route that does not have a FS in its topology table, it will look for an alternative path to the destination. The route is now considered **active**. A route is considered passive when a router is not performing computation for the particular route. Re-computing an active route involves sending query packets to all neighbors on interfaces other than the one used for the previous successor (due to the split horizon rule), inquiring whether they have a route to the particular destination. If a neighbor has an alternative route, it will answer the query with the path in a reply packet and does not further propagate the query packet. If a neighbor does not have an alternative path, it will continue to query its own neighbors for an alternative path. The queries would be propagated through the network and create an expanding tree of queries.
- When a router replies a query, it stops spreading of the query through that branch of the network. However, the query can still spread through other portions of the network as other routers attempt to find alternative paths which might not exist.

- Due to the nature of the reliable multicast approach implemented by EIGRP, a reply must be received for each query generated when searching for alternative paths for a lost route. When a route enters the active state and queries are initiated, the only way the route can come out of the active state and transition to passive state is after the router received a reply for every query generated! Besides that, every query and reply is acknowledged using an ACK message!
- For each neighbor to whom a query is sent, an EIGRP router will set a **reply status flag (r)** to keep track of all outstanding queries that are waiting for replies. The DUAL computation is completed when the router has received a reply for every query sent out earlier.
- By default, if an EIGRP router does not receive a reply to for each outstanding query within 3 minutes (the default Active timer value), the route will enter into the **Stuck-in-Active (SIA)** state. When a route is in the SIA state, the router will reset the neighbor relationships for neighbors that failed to reply, and cause the router to go active on all routes known through the neighbors and re-advertise all its known routes to the neighbors. **Query scoping**, which limits the scope of query propagation through a network – query range, helps to reduce the occurrences of SIA. Keeping the query packets close to the source reduces the chance that a failure in a part of the network to involve routers in other parts of the network in the query-reply convergence process.
- The best methods for limiting the scope of query range and building scalable EIGRP networks:
  - i) Configure route summarization on the outbound interfaces of the appropriate routers.
  - ii) Configure the remote routers as EIGRP stub routers.
- Other methods used for limiting query range include route filtering and interface packet filtering. Ex: When specific EIGRP routing updates to a router are filtered, upon the router receives a query about a filtered network, the router would reply and indicate that the network is unreachable and does not further propagate the query to other neighbors.
- The **timers active-time [time-limit | disabled]** EIGRP router subcommand changes the active-state time limit in minutes. The default value which is 3 minutes – **timers active-time 3** will not be shown in the router configuration files. The **timers active-time** command is equivalent to the **timers active-time disabled** command. With the active timer disabled, SIA route which does not receive a reply within 3 minutes would not cause the reset of the neighborhood between the querying and queried routers.
- The **eigrp log-neighbor-changes** router subcommand enables the logging of neighbor adjacency events monitors the stability of the routing system and detects SIA-related problems. **Note:** This command is enabled by default. **Note:** The **show ip eigrp sia-event** hidden EXEC command displays the SIA events.
- An erroneous approach which is often used to decrease the chance of occurrence of a SIA route is implementing multiple EIGRP ASs to somewhat simulate OSPF areas in order to bound the query range, with mutual redistribution between the different ASs. However, this approach will not achieve the intended results. When a query reaches the edge of an AS, where routes are redistributed into another AS, the original query is first answered, followed by the edge router initiates a new query in the other AS. As a result, the query process has not been stopped; the querying process continues into other AS, and the route can eventually enter into SIA.
- Another misconception about AS boundaries is implementing multiple ASs to protect one AS from route flapping in another AS. If routes are being redistributed between ASs, route transitions occur in an AS will be detected in the other ASs.



**Figure 6-10: The EIGRP Query-Reply Process on Redundant Topology**

- Figure 6-10 is used for the discussion of the EIGRP Query-Reply process upon a lost route on redundant topology. The convergence process is considered complex even with only 2 head quarter routers and 2 branch routers. In networks with hundreds of branch routers, the process would become even more complex and can cause severe problems.
- HQ1 advertises 172.16.1.0/24 to all other routers. The successor path for HQ2 to reach the network is via the Fast Ethernet link to HQ1. The successor paths for branch routers (BR1 and BR2) to reach the network are via their serial links to HQ1. Additionally, they have also learnt the feasible successor path through HQ2.
- Assume that the EIGRP metrics for Fast Ethernet and Serial are 100 and 1000 respectively.
- Below shows the EIGRP topology table for BR1 and BR2 for network 172.16.1.0/24. BR1 and BR2 have determined that the path to HQ1 is the successor while the path to HQ2 is the feasible successor (the AD is 200 through HQ2, which is less than the FD through HQ1) for the network.

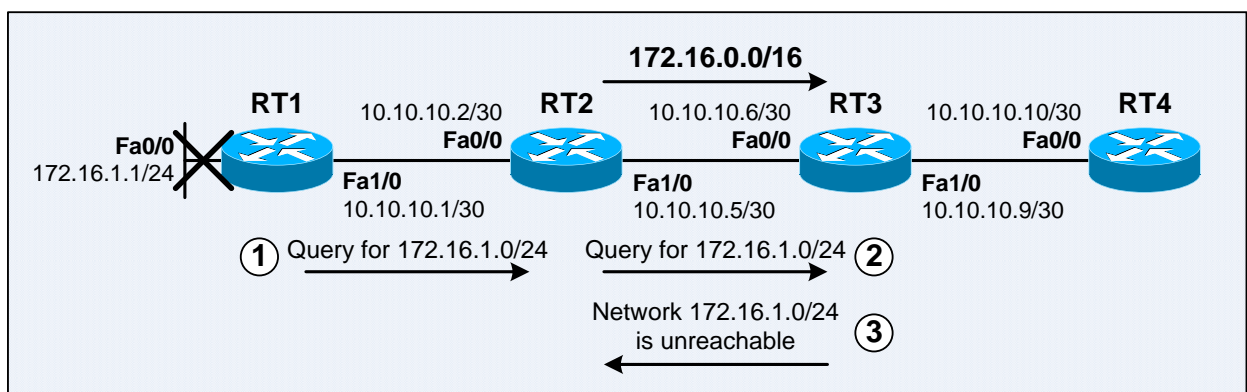
Neighbor	Feasible Distance	Advertised Distance
HQ1	1100	100
HQ2	1200	200

- Below shows the EIGRP topology table for HQ2 for network 172.16.1.0/24. HQ2 does not have a FS, as all paths through the branch routers have an AD greater than the FD through HQ1.

Neighbor	Feasible Distance	Advertised Distance
HQ1	200	100
BR1	2100	1100
BR2	2100	1100

- At a particular time, HQ1 lost the route to 172.16.1.0/24 due to a network device failure. It sends a query to HQ2, BR1, and BR2 to seek for a feasible successor as it does not have any. When the branch routers receive the query, they automatically install the feasible successor path through HQ2 into their routing tables and respond to HQ1 with their new successor through HQ2. They also remove the unreachable path through HQ1 from their routing tables and install the path through HQ2 into their routing tables.

- Now HQ1 has received the responses for 2 out of its 3 queries, but it is still waiting for the response from HQ2. When HQ2 receives the query from HQ1 for 172.16.1.0/24, it propagates the query to BR1 and BR2, as it does not have a FS but knows that a path exists through each branch router to reach 172.16.1.0/24.
- BR1 and BR2 receive the query from HQ2 and check their topology tables for alternative paths. However, both branch routers do not have another path at the moment, as HQ1 has just informed that it has lost the path to this network. As the branch routers do not have an answer to the query from HQ2, they create a query and send it to all neighbors except the neighbor that they received the query from (due to split horizon). In this case, the branch routers send the query to HQ1.
- HQ1 would reply to BR1 and BR2 that the route has an infinite metric and is unreachable. Once BR1 and BR2 receive the reply, the edge of the network is reached and the edge routers do not have any more neighbors to query. BR1 and BR2 would reply back to HQ2 and eventually HQ2 would reply back to HQ1.  
**Note:** The longest query range in this sample network is HQ1 → HQ2 → BR1, BR2 → HQ1.
- Below summarizes the events of the Query-Reply process in this scenario:
  - 1) HQ1 sends queries to HQ2, BR1, and BR2.
  - 2) BR1 and BR2 reply to HQ1 with their feasible successor paths via HQ2.
  - 3) HQ2 propagates the query from HQ1 to BR1 and BR2 as it does not have a feasible successor path.
  - 4) BR1 and BR2 propagate the query from HQ2 to HQ1.
  - 5) HQ1 replies to BR1 and BR2 that it does not have a feasible successor path.
  - 6) BR1 and BR2 reply back to HQ2.
  - 7) HQ2 replies back to HQ1.
- This scenario shows that in a network with redundant links between the head quarter and branch offices, not only the branch routers are required to respond to queries from the head quarter, but they also continue the search for a successor by propagated the queries back to the head quarter – HQ2 in this case. The Query-Reply packets which may traverse from the head quarter to the branch offices and back to the head quarter is often overlooked by network architects.
- Route summarization can limit the query range by limiting the knowledge of a router regarding the subnets in a network. When a subnet is down, queries will only be propagated to as far as the routers that receive the summary route and do not have knowledge about the specific subnet, as a router propagates the query about a network only if it has an exact match in the routing table.



**Figure 6-11:** Route Summarization Limits the Scope of EIGRP Query Range

- Figure 6-11 shows a sample scenario in which RT2 advertises a summary route of 172.16.0.0/16 to RT3. When the network 172.16.1.0/16 is down, RT3 would receive a query from RT2 as propagated from RT1. As RT3 has received only a summary route and the specific queried network is not in its routing table, it would reply with a “Network 172.16.1.0/24 is unreachable” message with an infinite metric of 4294967295 and does not further propagate the query to RT4.
- However, RT2 would send a query for the summary route 172.16.0.0/16 to RT3 and the query will be propagated to RT4 as the only component subnet 172.16.1.0/24 is unreachable while it requires at least one valid component subnet for the summary route to remain in its routing table. In production environments, RT1 should have multiple directly connected subnets in the 172.16.0.0/16 major network and hence this additionally query-reply process will never occur. **Note:** The FD of a summary route is same as the FD to the lowest-metric component subnet. If RT2 have different metric to the component subnets as advertised from RT1, it would send out a query to RT3 and the query will be propagated to RT4 when the lowest-metric component subnet which determines the metric for the summary route is unreachable, as it would like to query for any metric that is lower or better than its next lowest-metric component subnet that determines the metric for the 172.16.0.0/16 summary route.
- Route summarization can be implemented on the network as shown in Figure 6-10 to reduce the convergence traffic (queries and replies) on the redundant topology. The **ip summary-address eigrp 100 172.16.0.0 255.255.0.0** interface subcommand which configured on the outbound interfaces of the head quarter routers allows HQ1 and HQ2 to advertise the 172.16.0.0/16 summary route instead of the specific route to the branch routers – BR1 and BR2. Eventually, BR1 and BR2 will not propagate the query for 172.16.1.0/24 back to HQ1, but will reply to a query for 172.16.1.0/24 with the “network is unreachable” message instead.

## EIGRP Stub Routing

- In hub-and-spoke environments, spoke routers (branch routers) often do not need to maintain complete routing tables of the network, as the paths to other portions of the network are always through the hub routers (head quarter routers). Additionally, the massive query-reply processes between the hub and spoke routers upon a loss route would affect the stability of the network.
- A typical connection between a hub router and a spoke router has much less bandwidth than a connection in the network core. Therefore attempting to use such connections as transit paths would typically results in excessive congestion. EIGRP stub routing can prevent this problem by restricting the spoke router from advertising the routes from a hub router to another hub router. As a result, the hub routers would not notice there are alternative paths through the spoke routers. However, stub routing is unable to stop the hub router from advertising routes to spoke router. Manual summarization is required on the hub to advertise only a default route to the spokes. **Note:** A stub router would not advertise routes received from a neighbor to another neighbor.
- Without stub routing implemented on dual-homed spoke routers, route filtering configuration is required on the spoke routers to prevent them from appearing as transit paths to the hub routers.
- When the remote sites are not acting as a transit sites between the regional sites, the regional routers (hubs) should be configured to advertise only a default route to the remote routers. The remote routers (spokes) should be configured to advertise only their directly connected stub networks back to the regional routers to reduce the complexity of EIGRP convergence (the query-reply process) and the sizes of the EIGRP topology table and IP routing table.



- EIGRP stub routing an efficient method for limiting the query range, therefore conserves bandwidth due to unnecessary queries, prevents SIA events, and improves network stability.
- An EIGRP stub router would inform its stub status to upstream routers via the Hello packets. Any neighboring upstream router that receives such Hello packets will not query the stub router for lost routes, as the stub router has no downstream EIGRP neighbors and hence would not have alternative paths for a lost route. The upstream routers which connected to the stub router would answer any query on behalf of the stub router, which results in improved convergence time.
- The **eigrp stub [receive-only | connected , static , summary , redistributed]** router subcommand configures an EIGRP router as an EIGRP stub router. An EIGRP stub router would advertise its connected and summary routes to other neighboring routers by default. Below describes the optional keywords that can be used to modify this behavior:

Keyword	Description
<b>receive-only</b>	Restricts an EIGRP stub router from advertising any route to other routers. This option cannot be used with any other option as it prevents the advertisement of any type of route, which is not really useful; the other options can be configured in any combination. Use this option when the stub router has only a single interface.
<b>connected</b>	Allows an EIGRP stub router to advertise its connected routes. The <b>network</b> statement is required to include the connected interfaces into the EIGRP routing process. The <b>redistribute connected</b> router subcommand can also be used to redistribute connected subnets into the EIGRP routing process. This option is enabled by default and is the most widely used option.
<b>static</b>	Allows an EIGRP stub router to advertise its static routes. The <b>redistribute static</b> router subcommand is required to redistribute static routes.
<b>summary</b>	Allows an EIGRP stub router to advertise its summary routes. Summary routes can be configured manually with the <b>ip summary-address eigrp</b> interface subcommand or automatically with the <b>auto-summary</b> router subcommand. This option is enabled by default.
<b>redistributed</b>	Allows an EIGRP stub router to advertise external EIGRP routes learnt from other routing protocols or other EIGRP autonomous systems.

**Note:** Configuring EIGRP stub routing would teardown existing EIGRP neighbor relationships!

- By implementing EIGRP stub routing on BR1 and BR2 in Figure 6-10, HQ1 and HQ2 would notice both branch routers as stub routers and will suppress the queries to these neighbors. HQ1 and HQ2 will no longer query BR1 and BR2 when the successor to 172.16.1.0/24 is lost; they will send update packets to BR1 and BR2 instead. However, BR1 and BR2 will still query HQ1 and HQ2 as they lost both the successor and feasible successor to 172.16.1.0/24.

```

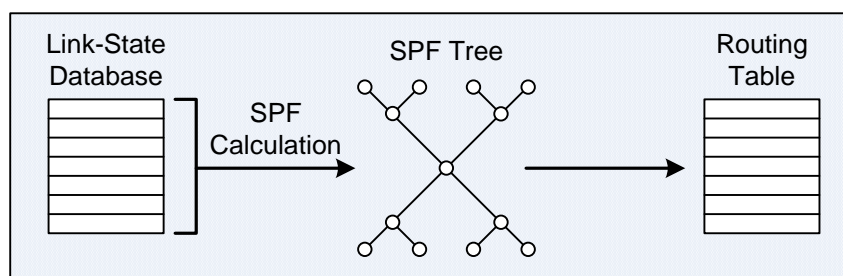
HQ1#sh ip eigrp neighbors detail
IP-EIGRP neighbors for process 100
H   Address           Interface           Hold Uptime      SRTT   RTO   Q   Seq
                               (sec)              (ms)          Cnt  Num
2   10.10.10.10        Se1/1              14 00:02:28     1   4500  0   22
  Version 12.3/1.2, Retrans: 2, Retries: 0
  Stub Peer Advertising ( CONNECTED ) Routes
  Suppressing queries
1   10.10.10.6         Se1/0              13 00:02:29     1   4500  0   22
  Version 12.3/1.2, Retrans: 2, Retries: 0
  Stub Peer Advertising ( CONNECTED ) Routes
  Suppressing queries
0   10.10.10.2         Fa0/0              12 00:02:30    424   3816  0   25
  Version 12.3/1.2, Retrans: 1, Retries: 0
HQ1#

```

## OSPF in a Single Area

- **Open Shortest Path First (OSPF)** is the most common open-standard interior gateway protocol. OSPF is considered a complex routing protocol due to the use of various packet types, the neighbor relationship establishment process, and the database exchange mechanism. OSPF and **Integrated Intermediate System-to-Intermediate System (IS-IS)** are classified as link-state routing protocols due to the nature in which they distribute routing information and calculate routes to the destination networks.
- Link-State routing protocols are developed to overcome the limitations of Distance-Vector routing protocols, eg: slow convergence, prone to routing loops, holddown, counting to infinity. The following are the overall characteristics of Link-State routing protocols:
  - i) Respond quickly to network changes.
  - ii) Send triggered updates upon a network change.
  - iii) Send periodic updates – **link-state refresh** regularly, eg: every 30 minutes.
- DV routing protocols employ a **distributed computation** approach, in which every router will perform route computation based on the information provided by other routers individually; LS routing protocols employ a **replicated distributed database** approach, in which every router contributes information into a database that will be distributed and shared among all routers.
- With DV routing protocols, routers rely on routing information and decisions made by neighboring routers. Routers do not have the full picture of the network topology. With LS routing protocols, routers have the full picture of the network topology, and can independently make routing decisions based on the accurate picture of the network topology.
- OSPF is a LS routing protocol intentionally designed and developed for TCP/IP. Other LS routing protocols that have been developed for all other major protocol suites include Intermediate System-to-Intermediate System (IS-IS) for OSI, NetWare Link Services Protocol (NLSP) for Novell NetWare, Advanced Peer-to-Peer Networking (APPN) for IBM SNA, and Private Network-to-Network Interface (PNNI) for ATM.
- A TCP/IP routing protocol can run directly over the data-link layer, over the IP network layer, or over the IP transport layer protocols – TCP or UDP. Running directly over the data-link layer requires a routing protocol to implement its own fragmentation mechanism as most data-link layers do not provide fragmentation and reassembly services; Besides that, code points must also be registered and assigned for the various data-link layers that the protocol runs over.
- Running over the IP network layer allows a routing protocol to utilize the fragmentation and reassembly services which were built into the IP network layer without reinventing the wheel; while the benefits of using UDP include a checksum mechanism for verifying packet integrity, a greater multiplexing capability with the 16-bit Destination Port field than the 8-bit IP Protocol field, and easier deployment on many operating systems, as sending and receiving packets directly over the IP network layer may require superuser privileges, whereas UDP interface is usually available to all users and applications.
- The reason that OSPF does not run over the IP transport layer protocols are that it does not require the reliability of TCP, as it has its own reliability built into the flooding mechanism with the use of checksum; and the small benefits of UDP (UDP also has a checksum field) were outweighed by the extra 8 bytes of UDP header overhead that would appear in every OSPF packet.

- Router interfaces are considered as **links** in OSPF. Link-State routing protocols advertise the status of the link (interface) – up or down to other routers. The link (or interface) will have state information associated with it (up or down) as well as one or more IP addresses.
- LS routing protocols only generate and send routing updates upon a network topology change. When a link changes its state, the router that detected the change would generate and send a **link-state advertisement (LSA)** about the link to all neighboring routers with a multicast packet. Each router would receive a copy of the LSA, updates its **link-state database (LSDB)**, and propagates the LSA to neighboring routers. Finally, the LSA **reliable flooding** feature ensures that all routers update and synchronize their LSDBs to have the same LSDB to reflect the new network topology before performing route computation and updating their routing tables.
- The LSDB is used to calculate the best paths to all networks in an internetwork. A LS router applies Dijkstra's SPF algorithm upon its LSDB to build a SPF tree. The best paths are then selected from the SPF tree and are inserted into the routing table.



**Figure 7-1: Components of Link-State Routing Protocols**

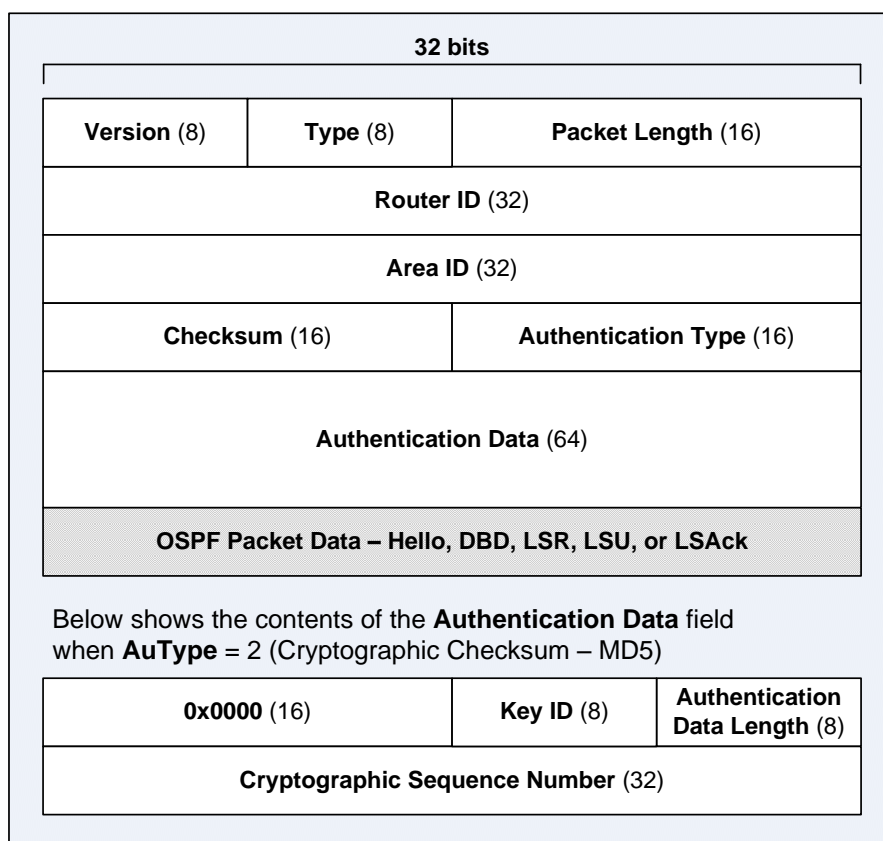
- LS routers collect routing information from all other routers within a defined area of the network. Each router **independently calculates** its best paths to all networks with the SPF algorithm. Therefore, incorrect information received from a router is very unlikely to cause problems, as each router would maintain its own view of the network.
- An OSPF router must maintain the following information for making correct routing decisions:
  - Its neighboring routers.** If a router loses contact with a neighbor router for a duration, it removes all paths through that router and recalculates the paths throughout the network. The neighbor adjacency information is stored in the neighbor table or adjacency database. An adjacency is referred to as a relationship between 2 OSPF routers that permits the direct exchange of routing information and routing updates.
  - All other routers within an area of the network and their attached networks.** A router recognizes other routers and networks within the same area through LSAs, which are flooded through the network. LSAs are stored in the topology table or LSDB.
  - The best paths to all destination networks.** Instead of advertising distances to known destinations, link-state algorithm advertises and distributes the states of its local network links to all routers. As a result, all routers will obtain the same link-state database that describes the network topology. Each router independently calculates the best paths to all destination networks in the network with SPF algorithm. The best paths are then inserted into the routing table.
- The main drawback of LS routing protocols is the memory consumption for maintaining the link-state databases or topology tables. However, as OSPF routers can maintain full information about all the routers and links within an area, each router can independently calculate and select loop-free and efficient paths to all destination networks in the area. This overcomes the routing by rumor problem as in DV routing protocols.

## OSPF Packet Format

- Below lists the 5 types of OSPF packets:

Message Type and Packet Type	Purposes
Type-1 – Hello	Discovers neighbors, negotiate capabilities, establishes and maintains adjacencies between neighbors.
Type-2 – Database Description (DBD)	Elects the Master and Slave, determines the initial sequence number for database exchange, and summarizes the LSDB contents using LSA headers during the database exchange process between routers.
Type-3 – Link-State Request (LSR)	Requests specific LS records (LSAs) that are seen during the database exchange process.
Type-4 – Link-State Update (LSU)	Sends specifically requested LS records to a neighbor and sends triggered updates upon a topology change. LSU packets are also used in the flooding process.
Type-5 – Link-State Acknowledgment (LSAck)	Acknowledges the receipt of LSU packets. A single LSAck packet can acknowledge multiple LSU packets.

- OSPF packets are encapsulated directly into an IP payload without the use of TCP or UDP. OSPF packets are consisting of **multiple encapsulations**. The IP header encapsulates 1 of the 5 OSPF packet types – Hello, DBD, LSR, LSU, and LSAck. All types of OSPF packets begin with an OSPF packet header, which is same for all packet types. The OSPF packet data following the header varies according to the packet type. Each type of OSPF packet has a number of fields, followed by more data.
- Most OSPF packets travel only a **single hop** between neighboring routers; the TTL field in the IP header is often set to 1 to ensure misconfigured routers not mistakenly forwarding OSPF packets. The exception for setting the TTL to 1 exists in certain OSPF hierarchical routing configurations.
- The Destination IP address field in the IP header is the IP address of a neighbor or either one of the OSPF multicast IP addresses – AllSPFRouters (224.0.0.5) and AllDRouters (224.0.0.6). All OSPF routers must be prepared to receive IP packets destined to 224.0.0.5; while both the DR and BDR must be prepared to receive IP packets destined to 224.0.0.6. **Note:** The corresponding Ethernet MAC addresses for the AllSPFRouters and AllDRouters multicast IP addresses are 0100.5e00.0005 and 0100.5e00.0006 respectively.
- OSPF Hello packets contain a list of known neighbors, while other types of OSPF packets contain a series of LSAs. These LSAs would also have their own specific types, headers, and data fields. Examples of LSAs are Router-LSAs, Network-LSAs, and Network-Summary-LSAs.
- OSPF Hello packets are sent to the AllSPFRouters IP multicast address (224.0.0.5) every 10 seconds for broadcast and point-to-point networks, and every 30 seconds for NBMA networks. The **ip ospf hello-interval** {sec} interface subcommand changes the interval of sending Hello packets out a particular router interface.
- The OSPF Hello mechanism ensures neighboring routers can send and receive packets among them (bidirectional link) in order to prevent routing problems that caused by unidirectional links.
- OSPF requires a reliable transmission mechanism. Since TCP is not used, it uses its own acknowledgment packets (OSPF packet Type-5) for acknowledging the receipt of OSPF packets.

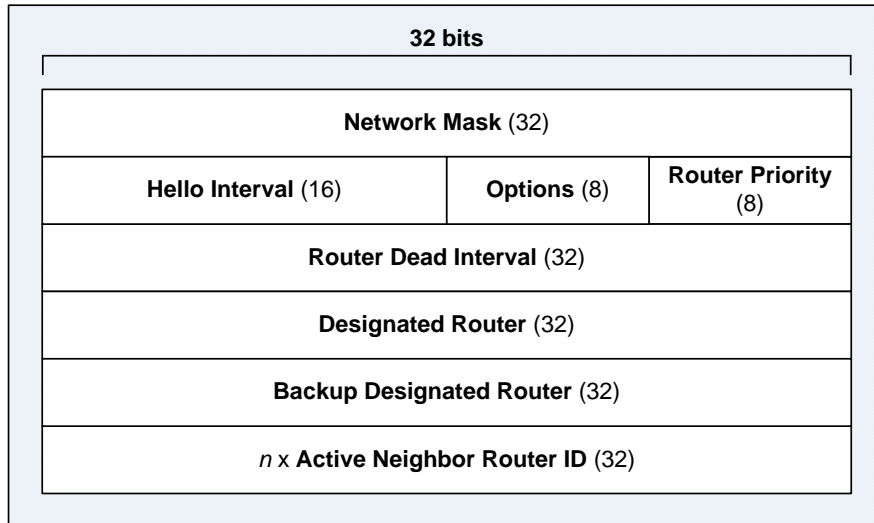


**Figure 7-2: OSPF Packet Format**

- Below lists the fields in the OSPF header:

<b>Field</b>	<b>Description</b>
<b>Version</b>	The values of 2 and 3 indicate OSPF versions 2 and 3 respectively.
<b>Type</b>	Differentiates the 5 types of OSPF packets.
<b>Packet Length</b>	The length of an OSPF packet, including LSA header and contents, in bytes.
<b>Router ID</b>	Uniquely identifies the source or originating OSPF router of an OSPF packet. It is not necessary to be reachable or exist in the routing table.
<b>Area ID</b>	Identifies the area from which an OSPF packet is originated and allows the receiving router to associate the packet to the proper level of OSPF hierarchy and ensure that OSPF hierarchy is configured consistently. The neighboring router interfaces must reside on the same subnet and area to form adjacencies. The Area ID for an OSPF packet sent over a virtual link is 0.0.0.0 (the backbone's Area ID), as virtual links are considered part of the backbone.
<b>Checksum</b>	Specifies a standard IP checksum of the entire packet including the header.
<b>Authentication Type</b>	The values of 0, 1, and 2 indicate <b>Null</b> (no authentication), <b>Simple Password</b> (plain text), and <b>Message Digest 5 (MD5) Cryptographic Checksum</b> OSPF router authentication methods respectively.
<b>Key ID</b>	Identifies the authentication algorithm and the secret key used to create the message digest.
<b>Authentication Data Length</b>	Specifies the length of message digest appended to an OSPF packet, in bytes. The message digest is not considered part of the OSPF packet.
<b>Cryptographic Sequence Num</b>	Specifies a non-decreasing number used to prevent replay attacks.

## OSPF Neighbor Adjacency Establishment and Database Synchronization Process

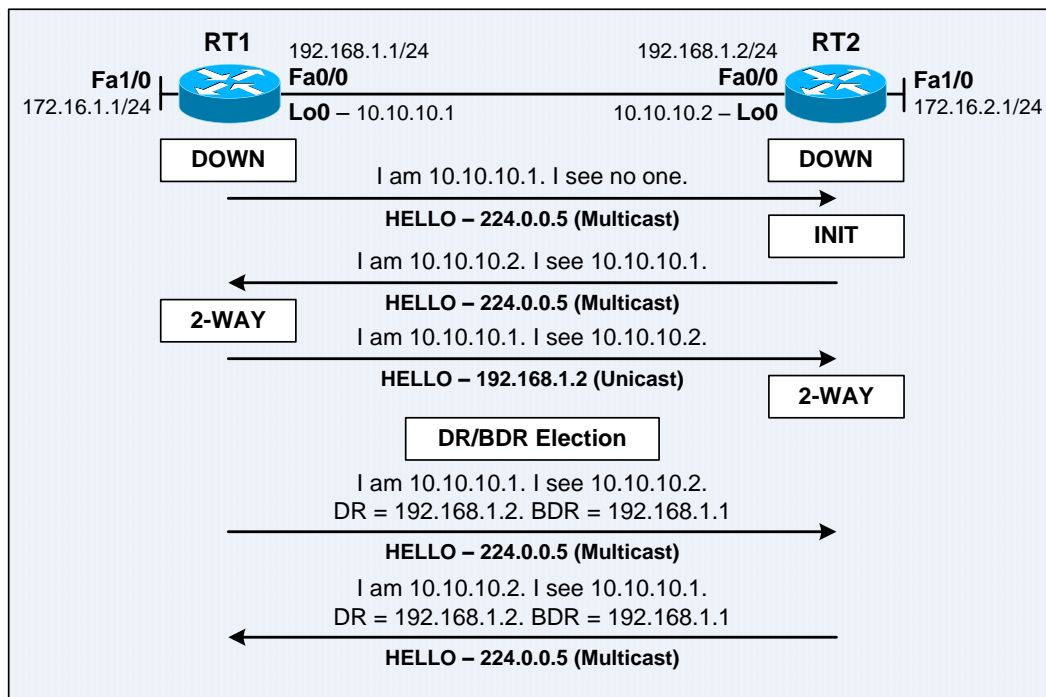


**Figure 7-3: OSPF Hello Packet Format**

- Below lists the fields in the OSPF Hello header:

Field	Description
<b>Network Mask</b>	Indicates the network mask of the interface from which the packet was sent. The Hello packet will be discarded if it contains a network mask that does not match the network mask of the interface on which the Hello is received.
<b>Hello Interval</b>	Indicates the period between the transmissions of Hello packets, in seconds.
<b>Options</b>	Allows OSPF routers to communicate their optional capabilities. OSPF routers may not form adjacency due to compatibility issues of capabilities.
<b>Router Priority</b>	Used in the election of DR and BDR. A value of 0 indicates that the originating router is a DROTHER and will never become a DR or BDR.
<b>Router Dead Interval</b>	Indicates the duration (in seconds) that the originating router will wait for a Hello from a neighbor before declaring the neighbor is dead / out of service.
<b>Designated Router and Backup Designated Router</b>	The physical IP address (not the Router ID) of the interface of the DR and BDR on the broadcast network. During the DR/BDR election process, this field may indicate the DR/BDR from the perspective of the originating router and not the finally elected DR/BDR. This field is set to 0.0.0.0 if there is no DR/BDR as the DR/BDR has not been elected yet or the network type does not require DR/BDR.
<b>Active Neighbor Router ID</b>	A recurring field that lists all the Router IDs of the neighbors which have established bidirectional communication with the originating router. Bidirectional communication is established when a router sees itself listed in the active neighbor list in the Hello packet received from a neighbor.

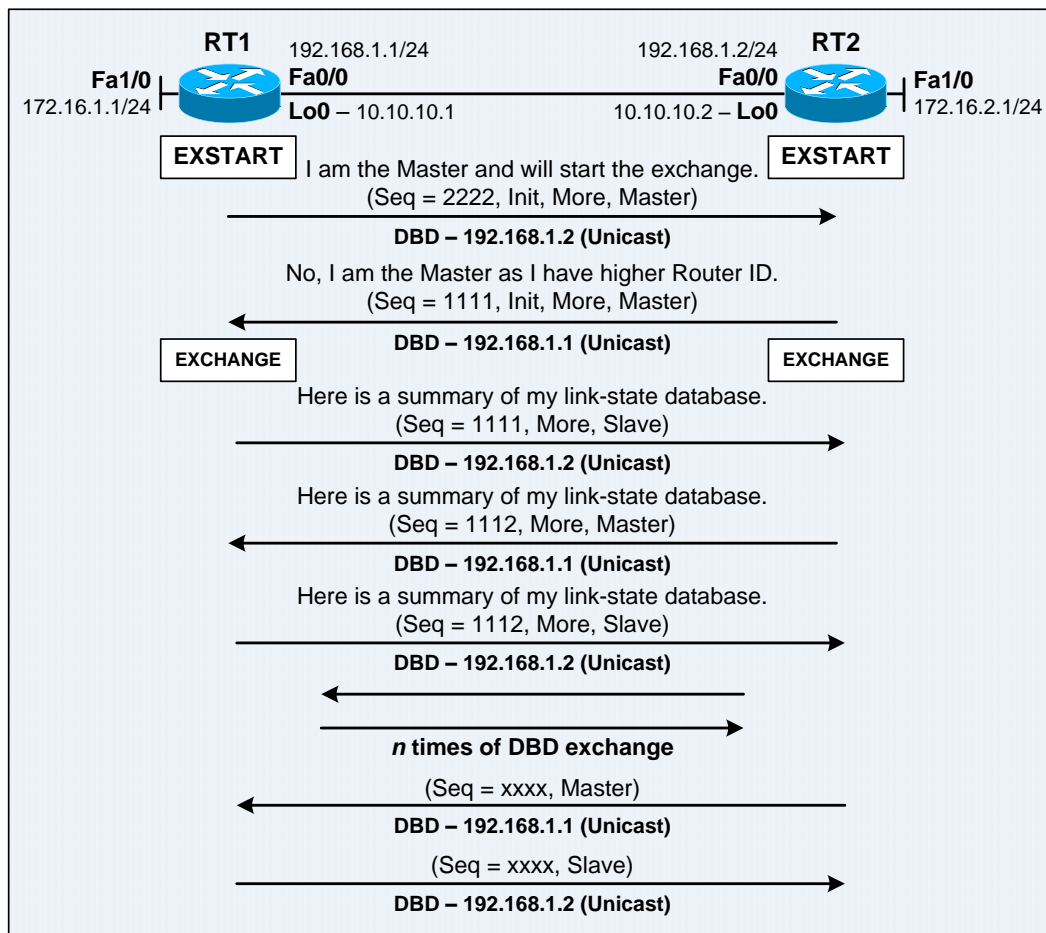
- An OSPF router must establish a neighbor relationship or adjacency with another neighboring router before exchanging routing information with the neighboring router.
- The OSPF Hello packets are being used to discover neighbors, and establish and maintain neighbor relationships between neighbors.
- The interfaces of neighboring routers must reside on the same network and have the same Hello and dead intervals, network masks, area ID, MTU, stub area flags in the Options field, and authentication password (if any) in order to form a neighbor relationship or adjacency.



**Figure 7-4:** Establishing Bidirectional Communication

- This section discusses how 2 neighboring OSPF routers on LAN establish adjacency, exchange link-state database information, and eventually route packets between the networks behind them.
- At the beginning, the OSPF neighbor state on both routers is DOWN as they haven't received any Hello packet each other. The OSPF process is enabled on RT1 at a particular time and triggered RT1 to send multicast Hello packets through all its interfaces participating in OSPF.
- RT2 which was running OSPF and resides on the same subnet and area as RT1 received the Hello packet from RT1 and entered into the INIT state. RT2 added RT1 into its OSPF neighbor list.
- Subsequently, RT2 sent a **multicast** Hello packet which lists RT1 in the OSPF neighbor list. When RT1 received the Hello packet from RT2, it noticed that another router has received its Hello packet as it is being listed in the neighbor list of RT2's Hello packet. RT1 entered into the 2-WAY state and added RT2's Router ID into its neighbor table. At the same time, RT1 immediately sent a **unicast** Hello packet that lists RT2 in the neighbor list to RT2 in order to speed up RT2 to enter the 2-WAY state as soon as possible. Both routers have established bidirectional communication as they see each other in their own neighbor lists.  
**Note:** Receiving a DBD packet from a neighbor in the INIT state would also cause a transition to the 2-WAY state. This is often seen in Point-to-Point network setups over serial links.
- The DR/BDR election occurs during the 2-WAY state for OSPF neighboring routers reside on broadcast network, eg: Ethernet. The DR/BDR election must occur before the routers can begin to exchange link-state information.
- **Note:** When an OSPF router joins a broadcast network which has a DR and BDR elected, it will only establish adjacency and enter into the FULL state with the DR and BDR. The neighbor state with other non-DR/BDR neighboring routers (DROTHERs) would stays in the 2-WAY state.
- **Note:** When an OSPF router has a higher Router ID or priority than the existing DR or BDR, it **does not preempt** the existing DR or BDR. This prevents the DR/BDR election process from occurring whenever a new OSPF router joins a broadcast network.

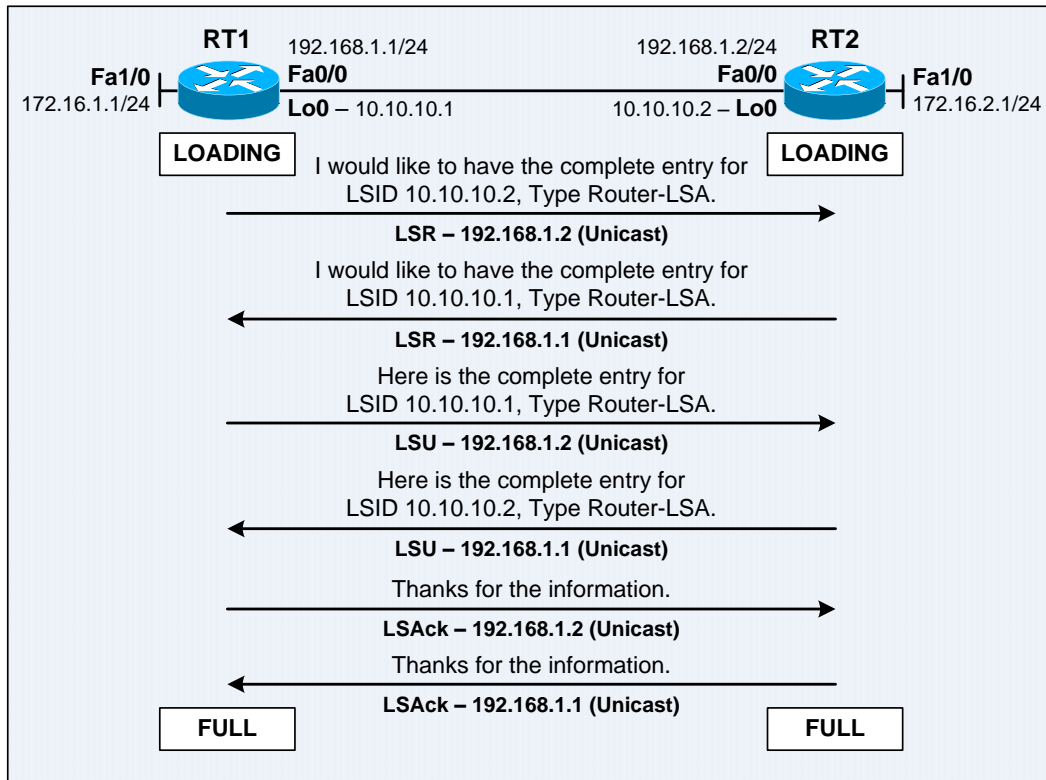




**Figure 7-5: Exchanging Basic Link-State Information**

- After the DR and BDR have been elected, both routers entered into the EXSTART state, in which the Master and Slave will be elected. Both routers will first claim to be the Master by sending empty DBD packets with a sequence number the Init and Master/Slave (MS) bits set. RT2 which has the higher Router ID would become the Master and controls the database synchronization process, eg: deciding the initial DD sequence number used for DBD exchange. The EXSTART state ends once the Master/Slave relationship is determined.
- During the EXCHANGE state, both routers exchange link-state information with multiple DBD packets for them to determine whether they have the same LSAs in their link-state databases. A DBD packet would include one or more LSA headers in the link-state database of the sender. An OSPF router would send **only LSA headers instead of the entire LSDB** to a neighbor during the EXCHANGE state. The LSA Type, Link-State ID, and Advertising Router fields in an LSA header are enough to uniquely identify a particular LSA. Sequence numbers are being used to determine the newness of link-state information.
- The transmission and reception of DBD packets resembles the operation of the TFTP protocol, in which only one DBD packet can be outstanding at a time. The Master will send the next DBD packet only when the previous DBD packet is acknowledged through a DBD packet with the same sequence number from the Slave. If the Master does not receive an acknowledgment for an outstanding DBD packet within the RxmtInterval, it would retransmit the previous DBD packet.
- The Slave would send a DBD packet with the same sequence number to acknowledge the receipt of a DBD packet from the Master. Therefore, the last DBD packet is always sent by the Slave.





**Figure 7-6:** Exchanging Complete Link-State Information

- During the **LOADING** state, an OSPF router would use LSR packets to request **more specific, recent, and complete LSAs** from a neighbor router in which the link-state information received during the **EXCHANGE** state is not in its LSDB or is more recent than the entry in its LSDB. Upon receiving an LSR packet, an OSPF router would reply with an LSU packet which contains the specific and complete LSAs. An LSAck packet is used to acknowledge the LSU sent from a neighbor router. Each LSA must be acknowledged separately to ensure reliable flooding. An LSA is being acknowledged by including its header in the LSAck packet, and multiple LSAs can be acknowledged in a single LSAck packet. To ensure reliability, a router will periodically retransmit an LSA sent to a neighbor until the neighbor acknowledges the receipt of the LSA.
- OSPF routers generally delay the acknowledgement of LSAs to fit more LSA acknowledgements into a single LSAck packet in order to conserve bandwidth and router processing resources.
- **Note:** Not all LSAs require explicit acknowledgment. When routing update cross, in which 2 neighboring routers send each other the same instance of LSA at about the same time, the received LSA will be treated as an **implicit acknowledgment** and no corresponding LSAck packet is required.
- Once the database synchronization process ends, both routers conclude that they have identical LSDB and are in fully adjacency state (**FULL**) with each other. Routers must be in the **FULL** state before they can forward packets to each other. Once adjacent routers are in the **FULL** state, they do not repeat the database synchronization process unless the **FULL** state changes.  
**Note:** At the beginning of the LSDB exchange process, both routers were in **merely adjacent** state.

- Below describes the OSPF **neighbor states** in the order of progressing functionality:

<b>DOWN</b>	Indicates that no Hello packets have been received from the neighbor yet. The router can send Hello packets and waiting to enter into the INIT state. If a router does not receive Hello packets from a neighbor within the RouterDeadInterval (4 missed Hellos) during the fully adjacent state, the neighbor state will change from FULL to DOWN.
<b>ATTEMPT</b>	This state only applicable for neighbors on NBMA networks. When a neighbor is in this state, it means that no information is received from the neighbor yet, but the local router is constantly sending a unicast Hello packet upon every HelloInterval (instead of the PollInterval) to contact the neighbor.
<b>INIT</b>	A router enters this state once received the 1st Hello packet from a neighbor. Upon receiving the Hello packet, the router declares a one-way state because it doesn't see itself (its Router ID) in the neighbor list in the Hello packet yet. As a result, the 2-way or bidirectional communication is not established yet.
<b>2-WAY</b>	A router enters this state once received the Hello packet that includes itself in the neighbor list, which means that the neighbor has received its Hello packet. This state indicates that <b>bidirectional communication</b> has been established and is the initial stage of neighbor relationship between neighboring routers. The routers cannot exchange routing information yet. The DR/BDR election occurs in this state. As both routers have received the Hello packets from each other and know their <b>physical IP addresses</b> on the broadcast network, subsequent Hello packets would include the DR and BDR information.
<b>EXSTART</b>	This state is used for the initialization of the database synchronization process. The Master-Slave membership between the router and the DR and BDR of the network are elected in this state. The router that has the higher Router ID will be elected as the Master. The initial DD sequence number used for DBD exchange is also being decided in this state. The sequence number is set by the Master to a unique value in its first DD packet, and it is incremented in subsequent DBD packets exchanged between the neighboring routers.
<b>EXCHANGE</b>	This state is when an OSPF router uses DBD packets to describe all LSAs in its link-state database to a neighbor that is in the EXCHANGE state too for the neighbor to determine whether it has the same LSAs in its link-state database. As multiple DBD packets will be exchanged during the database synchronization process, the Init, More and Master/Slave flags are being used to manage the LSDB exchange via a master-slave polling mechanism. <b>Note:</b> The DR/BDR is not necessary the Master in an exchange process!
<b>LOADING</b>	This state is when the exchange of link-state information really occurs. An OSPF router would send <b>only LSA headers instead of the entire LSDB</b> to a neighbor during the EXCHANGE state. An OSPF router would use LSR packets to request <b>more specific, recent, and complete LSAs</b> from a neighbor router in which the link-state information discovered during the EXCHANGE state is not in its LSDB or is more recent than the entry in its LSDB. When a neighbor router received an LSR packet, it would reply with an LSU packet which contains the specific and complete LSAs. An LSAck packet is used to acknowledge the LSU sent from a neighbor router.
<b>FULL</b>	The final OSPF neighbor state after the complete database information has been distributed and the link-state databases of routers are fully synchronized. <b>Note:</b> On broadcast and NBMA networks, a router would only establish FULL state with the DR and BDR. The neighbor state with other non-DR/BDR neighboring routers (DROTHERs) would stay in the 2-WAY state.

## OSPF Link-State Database

- OSPF maintains a database with only the most recent and up to date link-state information or records with the use of MaxAge and LSRefreshTime timers, and LS sequence numbers.
- The length of the LS Sequence Number field in an LSA header or an LSA is 32 bits. Beginning with the leftmost bit set, the first number is 0x80000001 and the last number is 0x7FFFFFFF. The sequence number is used to detect old or redundant LSA records. The larger the number, the more recent the LSA. An OSPF router would generate a new LSA upon a link-state change. One can notice which parts of the network are changing the most by looking at the LSDB – the parts that are described by LSAs whose have large LS sequence numbers and small LS ages.
- An OSPF router refreshes its self-originated LSAs that it is responsible for every 30 minutes (the LSRefreshTime interval) to ensure all routers have an accurate link-state database, just in case an LSA has been lost from or corrupted in one of the databases of other routers. Every time a LSA record is flooded or refreshed, its sequence number is incremented by 1, and its aging timer is reset. An LSA will never remain in the database for more than 1 hour (MaxAge) without being refreshed; it is considered invalid (eg: the originating router has failed) and will be removed from the LSDB. This LSA validation method conserves more bandwidth than DV routing protocols, which send entire routing tables at short periodical intervals.
- Since an invalid LSA may take as long as 1 hour for it to be removed from LSDBs and the LSA would certainly contain out-of-date (stale) information, OSPF requires the routers at both ends of a point-to-point link advertise the link before considering it during the route computation process.
- It is theoretically possible for an LSA to exist in the database for very long periods of time, refreshed every 30 minutes, until a point where the sequence number rolls over to the beginning. When this occurs, the existing LSA must first be deleted by prematurely aged out (MaxAge is set to 1 hour) to allow other routers to accept the new LSA with sequence number of 0x80000001. **Note:** A router is only allowed to update its self-originated LSAs at most once every 5 seconds. In the absence of errors, the 32-bit sequence number will require 680 years to roll over! 🤖
- The OSPF **premature aging** process allows the deletion of an LSA from a routing domain by setting its LS Age to MaxAge and floods its without waiting for its aging timer to reach MaxAge, as sometimes a router may want to delete an LSA instead of updating its contents. In order to avoid possible thrashing in which a router originates an LSA while another router deletes it, a router is allowed to prematurely age (delete) only its self-originated LSAs.
- The LS Age field of an LSA is also examined for other OSPF functions as below:

Constant	Value	Description
MinLSArrival	1 second	Minimum rate at which a router will accept an LSA update.
MinLSInterval	5 seconds	Minimum rate (rate limit) at which a router can update an LSA.
CheckAge	5 minutes	The rate at which a router verifies the checksum of an LSA contained in its LSDB.
MaxAgeDiff	15 minutes	2 LSA instances are considered separate when they differ by more than 15 minutes. The LSA with the smaller LS Age is considered more recent and accepted.
LSRefreshTime	30 minutes	The rate at which a router must refresh its self-originated LSAs.
MaxAge	1 hour	The maximum age that an LSA is considered valid and being retained in the LSDB.

- An OSPF router can easily identify its self-originated LSAs as the Advertising Router field in those LSAs is set to its own Router ID. An OSPF router is allowed to update and delete only its self-originated LSAs that it is responsible for. Knowing which router has originated a particular LSA allows a calculating router to decide whether the LSA should be used in route computation, and how it should be used.

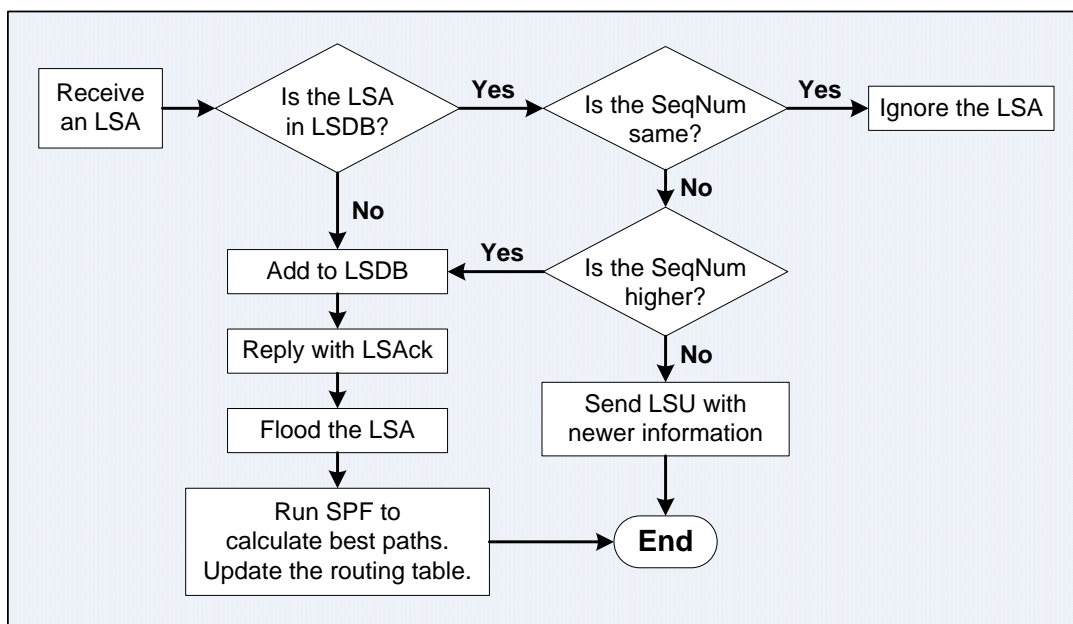


Figure 7-7: LSA Operations upon Receiving an LSA

- The **show ip ospf database EXEC** command displays the link-state database on a router.

```

RT1#sh ip ospf database

                OSPF Router with ID (10.10.10.1) (Process ID 100)

                Router Link States (Area 0)

Link ID        ADV Router    Age          Seq#           Checksum Link count
10.10.10.1    10.10.10.1    21          0x80000002    0x000419 3
10.10.10.2    10.10.10.2    21          0x80000002    0x0039DE 3

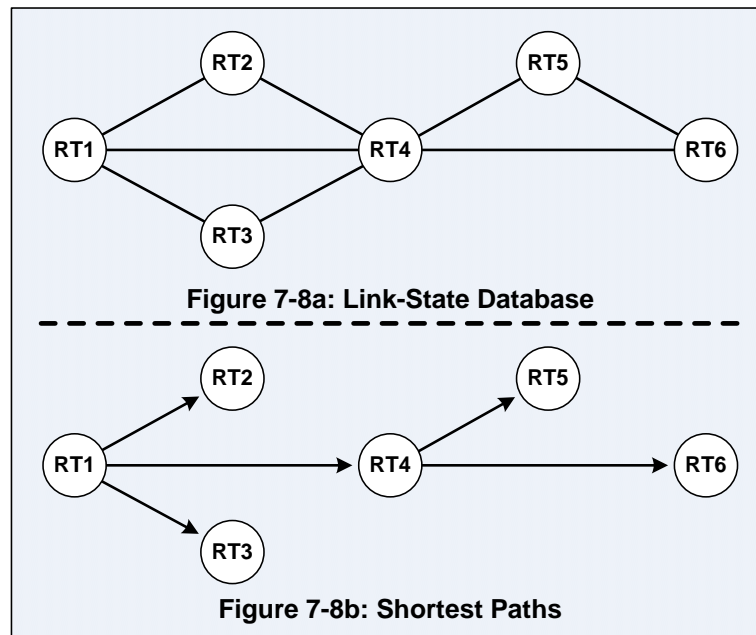
                Net Link States (Area 0)

Link ID        ADV Router    Age          Seq#           Checksum
192.168.1.2    10.10.10.2    22          0x80000001    0x004E20
  
```

<b>Link ID</b>	Indicates the Router ID of the source or originating router of a Router LSA, which is announced by every OSPF router for the status of its own interfaces.
<b>ADV Router</b>	Indicates the Router ID of the source or originating OSPF router of an LSA.
<b>Age</b>	Indicates the length of time in seconds since the LSA was last updated. The maximum age is 3600 seconds (1 hour).
<b>Seq#</b>	Indicates the sequence number of an LSA.
<b>Checksum</b>	Indicates the checksum of an LSA to ensure reliable receipt of the LSA.
<b>Link count</b>	Used only in Router LSAs to indicate the total number of directly connected links, which include all point-to-point, transit, and stub links. Point-to-point serial links are count as 2 (as connections to another router and a stub network); all other links, including Ethernet, are count as 1.

## Shortest Path First (SPF) Route Calculation

- LS routing protocols use a mathematical algorithm developed by Edsger Dijkstra for calculating the best paths throughout a complex internetwork. By assigning a cost or metric to every link in the network and placing a specific node at the root of a SPF tree and summing the costs toward every destination, the branches of the SPF tree can be calculated to determine the best path to every destination. The best paths are offered to the forwarding database – the routing table.
- By default, an OSPF interface cost is calculated based on its configured bandwidth. The higher the bandwidth, the lower the cost. An OSPF cost can also be manually defined for each interface, but this will override the default cost value calculated based on the configured bandwidth.  
**Note:** OSPF metrics can be **asymmetric** – different metrics could be assigned to both sides of a link.



**Figure 7-8:** Dijkstra's SPF Calculation

- Figure 7-8 shows an example of Dijkstra's SPF calculation and below are the steps in the route computation process. Assume that all links are Fast Ethernet with an OSPF cost of 1.
  - RT6 advertises its LSAs to RT4 and RT5. RT5 advertises RT6 and its own LSAs to RT4. RT4 advertises RT5, RT6, and its own LSAs to its neighbors – RT1, RT2, and RT3. RT2 and RT3 advertise all LSAs from RT4, RT5, RT6 and their own LSAs to RT1.
  - The LSA advertisement or flooding process follows the split-horizon rule, in which a router is not allowed to advertise an LSA back to the originating router of the LSA. Ex: RT5 is not allowed to advertise RT6's LSAs back to RT6.
  - RT1 has 3 neighboring routers – RT2, RT3, and RT4, in which its will receives the LSAs of all other routers in the network. RT1 can eventually figure out the links between all routers and the network topology as shown in Figure 7-8a based on the received LSAs.
  - RT1 calculates the best path to each destination by summing the costs to each destination.
  - Figure 7-8b shows the best paths (SPF tree) to each destination from the perspective of RT1. The best paths or routes will be offered to the forwarding database – the routing table.
- The SPF route calculation is considered the simplest part of OSPF. However, it can become complicated in the presence of broadcast and NBMA networks, OSPF hierarchical routing, and when some OSPF extensions (eg: NSSA) are being used.

## OSPF Network Types

- OSPF was originally designed for computer networks interconnected with point-to-point links. OSPF has evolved to support broadcast (eg: Ethernet) and **non-broadcast multi-access (NBMA)** (eg: ATM, Frame Relay) networks. The OSPF operation is different for each network type; therefore OSPF must be properly configured to function correctly over certain network types. **Note:** There are 5 OSPF operation modes available for NBMA networks.
- A point-to-point network often being used to connect 2 routers via a serial leased-line with HDLC or PPP as the data link layer encapsulation. Both routers dynamically discover each other via Hello packets destined to the 224.0.0.5 multicast IP address, exchange database information, and establish full adjacency between them. No DR and BDR election is performed; as there are only 2 routers reside in a point-to-point network, in which there is no requirement for DR/BDR.
- Usually the source IP address of an OSPF packet is set to the IP address of the outgoing interface. OSPF supports the use of IP unnumbered interfaces, in which the source IP address is set to the IP address of another interface.
- A number of  $n$  routers reside on a broadcast multi-access network could have been modeled as  $\frac{n(n-1)}{2}$  point-to-point connections (one for each pair of routers) and followed by performing the point-to-point LSA flooding process over each connection. Although this approach is simple, it would consume unnecessary bandwidth resources over the network. Instead, the network is modeled as a star network with  $n$  point-to-point connections. A special router called the **Designated Router (DR)** is elected on the network, in which all other routers on the network only required to flood or receive LSAs to/from the DR in order to conserve bandwidth resources. The DR manages the LSA flooding process on a broadcast multi-access network.
- The main benefit of having DR and BDR for broadcast multi-access networks (eg: Ethernet) is **reducing routing update traffic**. The DR and BDR act as the central point of contact for link-state database exchange between routers on a broadcast multi-access network, in which each router establishes full adjacency and sends its link-state information to the DR and BDR only (using the AllDRouters IP multicast address), instead of all routers on the network. The DR will then flood the link-state information received from a router to all other routers on the network. If the BDR does not see the LSU from the DR within the LSA retransmission interval (typically 5 seconds), it will step in and flood the LSA back into the broadcast network (using the AllSPFRouters IP multicast address) in order to keep the reliable database synchronization running smoothly over the broadcast network, even before noticing the failure of the DR.

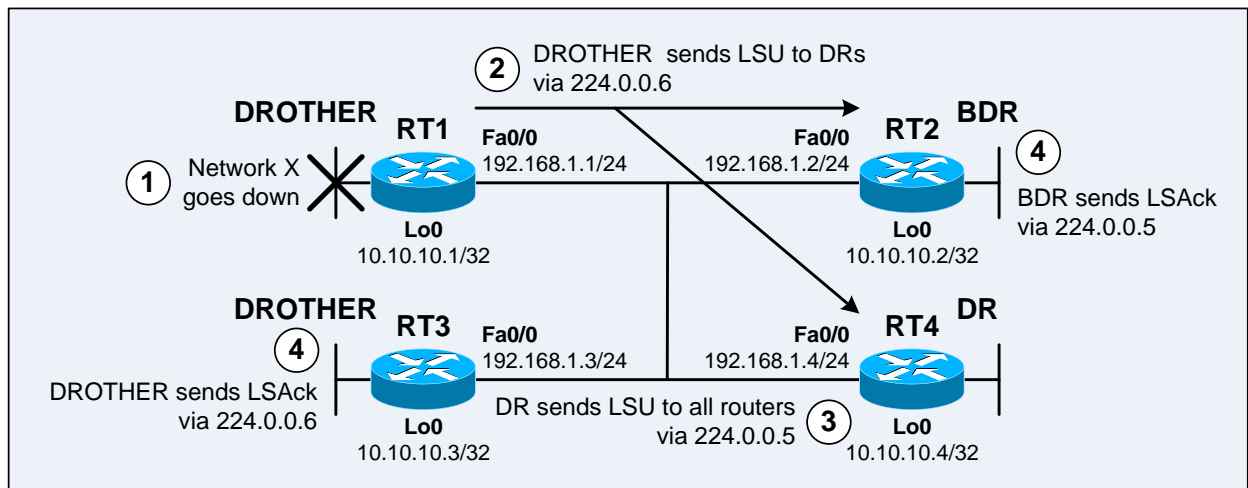


Figure 7-9: OSPF LSA Flooding on Broadcast Network

- The DR performs the LSA flooding and LSDB synchronization tasks; whereas the BDR does not perform any of the DR functions when the DR is operating, it just receives all the information. The BDR acts as the hot standby and performs the mentioned tasks in case the DR fails. If the DR fails, the BDR automatically becomes the DR, and another new BDR will be elected.
- The formula for calculating the number of adjacencies on a LAN with the use of DR and BDR is  $2(n - 1) - 1$ , where  $n$  is the number of routers on the LAN.  
Ex: There are 5 OSPF adjacencies between 4 routers on a LAN; 2 between DR and DROTHERs, 2 between BDR and DROTHERs, and 1 between DR and BDR.
- The DR and BDR represents all routers on a broadcast network and will pass their LSDBs to a new router that joins the network; as it is inefficient to allow all routers on the network to pass the same information to the new router. **Note:** There are chances where a new router that joins a broadcast network may first form an adjacency with the BDR instead of the DR. The BDR would exchange link-state information with the new router but it won't flood any LSU sent by the new router to all other routers on the network. The flooding of LSUs from the new router will be performed by the DR when the new router form an adjacency and exchange LSDBs with the DR.
- The DR election occurs when 2 neighboring OSPF routers have entered into the 2-WAY state. If the Hello packets indicate a DR of 0.0.0.0, it means that the DR has not been elected yet; the routers would wait for a period known as the OSPF **wait time** before attempt to elect the DR. The purpose is to allocate enough time for all routers reside on a broadcast network to complete their initialization process upon a power failure before participate in the DR election. Otherwise, the 1st router that became active would always be elected as the DR. The OSPF Wait timer is set to the same value as the OSPF Dead timer.
- If the received Hello packets already specified a Router ID for the DR, it means that the DR has already been elected and a new router that joins the broadcast network does not have to wait to elect a DR and will accept the DR specified in the received Hello packet as the current DR.
- During the DR election process, the routers with the highest and second-highest OSPF priority interfaces are elected as the DR and BDR respectively. The default OSPF interface priority is 1. In case of a tie in the OSPF interface priority, the routers with the highest and second-highest Router IDs are elected as the DR and BDR respectively. A router with an OSPF interface priority of 0 is not intended to be elected as a DR or BDR. This technique is often used to prevent certain routers to consume additional resources for becoming the DR or BDR. A router that is neither a DR nor BDR is designated as a DROTHER. The **ip ospf priority {priority}** interface subcommand changes the OSPF priority for a particular interface.  
**Note:** The BDR is always elected first. It is then being promoted to become the DR, and a new BDR is then elected. This order is important because the process must be repeatable when the DR is lost – the BDR is promoted, and a new BDR is elected.
- When a router with a higher priority or Router ID joins a broadcast network, it does not preempt the existing DR and BDR. The DR or BDR changes only when one of them is out of service. If the DR is out of service, the BDR becomes the DR, and a new BDR will be elected; if the BDR is out of service, a new BDR will be elected. The newly elected BDR would perform the time-consuming database synchronization process with all DROTHERs on the network.  
**Note:** Changing the priority on a DROTHER does not preempt the existing DR and BDR and does not affect the normal operation of the DR and BDR. However, changing the priority to 0 on a DR or BDR takes effect immediately in which the DR or BDR will become a DROTHER, the BDR becomes the DR, and a new BDR will be elected.



- Different sets of DR and BDR will be elected for each multi-access broadcast network. A router can be a DR on a network and a DROTHER on another network.
- The operation of OSPF over NBMA networks is almost same as the operation of OSPF over broadcast networks. The flooding mechanism with the DR; and the representation of networks with network-LSAs in the LSDB are identical for both broadcast and NBMA networks. The only difference is the **discovery of neighboring routers**. On broadcast networks, an OSPF router can dynamically discover its neighbors with multicast OSPF Hello packets; whereas on NBMA networks, OSPF neighbors must be configured manually and unicast OSPF Hello packets will be used instead. Frame Relay, X.25, and ATM are examples of NBMA networks.
- The non-broadcast nature of NBMA creates reachability issues for the operation of OSPF. NBMA networks support more than 2 routers on a network, but have no broadcast capability. In order to implement broadcasting or multicasting, a router replicates the packets to be broadcast or multicast and sends them individually on every PVC to all destinations; however, this process is CPU and bandwidth intensive. If an NBMA network topology is not fully meshed, a broadcast or multicast sent by a router would not reach all other routers.
- The hub-and-spoke or star topology is the most common NBMA topology in which remotes sites connect to a central site that provides an application service. The hub-and-spoke partial mesh physical topology does not provide the multi-access capability in which OSPF relies on. The election of DR and BDR is an issue in NBMA networks as the DR and BDR require full physical connectivity with all routers in the NBMA networks.
- By default, OSPF unable to automatically establish adjacencies with neighboring routers over NBMA interfaces. Several OSPF configuration options are available for NBMA networks depending upon the type of network topology.
- RFC 2328 – OSPFv2 defines the following official OSPF operation modes for NBMA networks:

<b>Non-broadcast</b>	Simulates the OSPF operation on broadcast networks. OSPF neighbors must be configured manually and will elect a DR and BDR. Typically used for full-mesh topologies.
<b>Point-to-multipoint</b>	Treats an NBMA network as a collection of point-to-point links. The routers automatically identify their neighboring routers but do not elect a DR and BDR. Typically used for partial-mesh topologies.

**Note:** The choice between the above operation modes determine how the Hello and LSA flooding mechanisms work over an NBMA network. The main advantage of point-to-multipoint mode is requires less manual configuration; while the main advantage of non-broadcast mode is less overhead traffic compared to point-to-multipoint mode.

**Note:** Cisco offers 3 more OSPF operations modes for NBMA networks – **non-broadcast point-to-multipoint, broadcast, and point-to-point**.

- The DR resides on a **broadcast** or **non-broadcast** NBMA network would generate Type-2 Network-LSA for the NBMA network.



- The **ip ospf network {broadcast | non-broadcast | point-to-multipoint [non-broadcast] | point-to-point}** interface subcommand select the OSPF operation mode for an interface.

Parameter	Description
<b>broadcast</b> (Cisco-specific)	For full-mesh or partial-mesh NBMA topology. One IP subnet. 10-sec/40-sec Hello/Dead timers. Makes the WAN to be appeared as a LAN. Uses multicast Hello packets to automatically discover the neighbors. Performs DR/BDR election (after the OSPF Wait timer elapsed). <b>Note:</b> Manually determine the DR using OSPF interface priority and ensure the DR has full connectivity with all other routers.
<b>non-broadcast</b> (RFC-compliant)	For full-mesh or partial-mesh NBMA topology. One IP subnet. 30-sec/120-sec Hello/Dead timer. Neighbors are manually configured. Performs DR/BDR election (after the OSPF Wait timer elapsed). <b>Note:</b> Manually determine the DR using OSPF interface priority and ensure the DR has full connectivity with all other routers.
<b>point-to-multipoint</b> (RFC-compliant)	For full-mesh or partial-mesh NBMA topology. One IP subnet. 30-sec/120-sec Hello/Dead timer. Uses multicast Hello packets to automatically discover the neighbors. Does not perform DR/BDR election. Neighboring routers enter into the EXSTART state as soon as they entered into the 2-WAY state. Treats a NBMA network as a collection of point-to-point links rather than a multi-access network.
<b>point-to-multipoint non-broadcast</b> (Cisco-specific)	For full-mesh or partial-mesh NBMA topology. One IP subnet. 30-sec/120-sec Hello/Dead timer. Neighbors are manually configured. Does not perform DR/BDR election. Neighboring routers enter into the EXSTART state as soon as they entered into the 2-WAY state. Uses when the network has many dynamic connections, in which SVCs are being used instead of PVCs.
<b>point-to-point</b> (Cisco-specific)	For point-to-point subinterfaces partial-mesh NBMA topology. Different IP subnet for each subinterface. Supports IP Unnumbered. 10-sec/40-sec Hello/Dead timer. Uses multicast Hello packets to automatically discover the neighbors. Does not perform DR/BDR election. Neighboring routers enter into the EXSTART state as soon as they entered into the 2-WAY state. Used when only 2 routers need to form an adjacency over a single link.

The default OSPF operation mode on a main Frame Relay interface is **non-broadcast**.

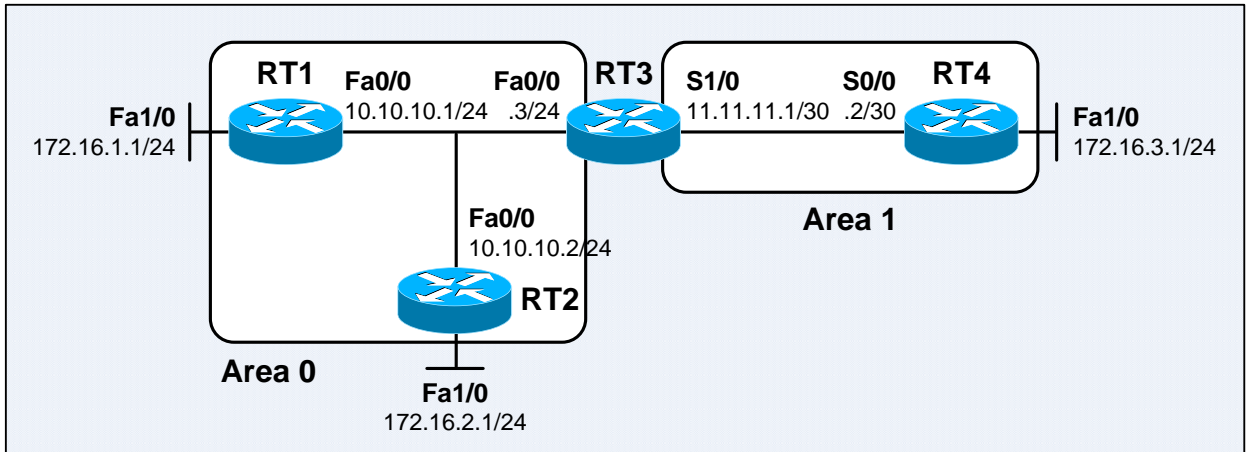
The default OSPF operation mode on a Frame Relay multipoint subinterface is **non-broadcast**.

The default OSPF operation mode on a Frame Relay point-to-point subinterface is **point-to-point**.

- Not all link layer technologies are able to be mapped directly to the point-to-point leased lines, broadcast networks, and NBMA networks as discussed earlier. Ex: The OSPF Demand Circuit Extension which suppress the Hello and LSA refresh functions has been developed to support OSPF over demand circuits (eg: dial-up telephone lines). With the OSPF Demand Circuit option configured on an interface, OSPF establish an adjacency and perform database synchronization, the adjacency remains active even after Layer 2 of the demand circuit goes down.
- **OSPF Multicast Extension (MOSPF)** is an enhancement to the OSPF unicast routing protocol that supports IP multicast routing within an administrative domain.

## OSPF in a Single Area Lab

- The **router ospf** *{process-id}* global configuration command enables an OSPF process and enters into the OSPF configuration mode. It requires a parameter called the **OSPF Process ID**. When a router runs multiple OSPF processes, the process IDs are used to distinguish the processes and databases. Unlike the autonomous system number in EIGRP router configuration, the OSPF process ID is **locally significant** and does not need to match the process IDs on other routers in order for the OSPF routers to establish adjacency and exchange routing information.
- Running multiple OSPF processes on a router is not recommended, as it creates multiple independent database instances that add extra overhead to the router.  
**Note:** Running multiple OSPF processes on a router is not the same as running multiarea OSPF.
- The **network** *{ip-addr}* *{wildcard-mask}* **area** *{area-id}* router subcommand identifies the interfaces that belong to an OSPF process, the interfaces to transmit and receive routing updates, the OSPF area that the interfaces reside on, and the networks to be included in the routing updates.
- The combination of *ip-address* **0.0.0.0** exactly matches an interface address; while the combination of **0.0.0.0 255.255.255.255** matches all interfaces on a router.  
**Note:** Configuring the combination of **0.0.0.0 0.0.0.0** will be automatically changed to **0.0.0.0 255.255.255.255**.
- The *area-id* parameter can be specified as either a decimal value in the range of 0 to 4294967295, or in dotted-decimal notation similar to an IP address – A.B.C.D, which allows the association of OSPF areas with IP subnets, in which a subnet address is specified as the *area-id*.  
**Note:** The OSPF Area ID field is 32-bit in length, which is the same as an IP address. Therefore its value ranges from 0 to  $2^{32} - 1$ , which is 4294967295. An OSPF Area ID of 10.10.10.0 is equivalent to 168430080 (0x0A0A0A00).
- Beginning with Cisco IOS Release 12.3(11)T, OSPF can be enabled directly on an interface with the **ip ospf** *{process-id}* **area** *{area-id}* [**secondaries none**] interface subcommand. This command simplifies the configuration of unnumbered interfaces. The **secondaries none** keyword prevents secondary IP addresses on an interface from being advertised.
- Loopback interface IP addresses are advertised as /32 host routes in Router-LSAs, as IP packets (eg: ICMP, SNMP) may still be destined to them even they are unavailable for regular traffic flow. A loopback interface may be looped back either in hardware or software.
- The **log-adjacency-changes** [**detail**] router subcommand configures a router to send a Syslog message upon an OSPF neighbor adjacency state change. The **log-adjacency-changes** router subcommand is enabled by default, but only OSPF neighbor up and down events are reported. The **detail** keyword configures a router to send a Syslog message upon each OSPF state change, not just when a neighbor goes up or down.
- The **ip ospf hello-interval** *{sec}* and **ip ospf dead-interval** *{sec}* interface subcommands change the OSPF Hello and Dead interval timers for a particular interface respectively.  
**Note:** Both timer values must be the same for all routers reside on a network to form adjacencies.



**Figure 8-1:** Sample OSPF Multiarea Network

- The **show ip ospf** EXEC command displays the OSPF Router ID (RID), OSPF timer values, the number of times the SPF algorithm has been executed, authentication, and LSA information.

```

RT3#sh ip ospf
Routing Process "ospf 100" with ID 11.11.11.1
  Supports only single TOS(TOS0) routes
  Supports opaque LSA
  Supports Link-local Signaling (LLS)
It is an area border router
  Initial SPF schedule delay 5000 msec
  Minimum hold time between two consecutive SPF's 10000 msec
  Maximum wait time between two consecutive SPF's 10000 msec
  Minimum LSA interval 5 secs. Minimum LSA arrival 1 sec
  LSA group pacing timer 240 secs
  Interface flood pacing timer 33 msec
  Retransmission pacing timer 66 msec
  Number of external LSA 0. Checksum Sum 0x000000
  Number of opaque AS LSA 0. Checksum Sum 0x000000
  Number of DCbitless external and opaque AS LSA 0
  Number of DoNotAge external and opaque AS LSA 0
  Number of areas in this router is 2. 2 normal 0 stub 0 nssa
  External flood list length 0
    Area BACKBONE (0)
      Number of interfaces in this area is 1
      Area has no authentication
      SPF algorithm last executed 00:08:26.588 ago
      SPF algorithm executed 4 times
      Area ranges are
      Number of LSA 6. Checksum Sum 0x042E4D
      Number of opaque link LSA 0. Checksum Sum 0x000000
      Number of DCbitless LSA 0
      Number of indication LSA 0
      Number of DoNotAge LSA 0
      Flood list length 0
    Area 1
      Number of interfaces in this area is 1
      Area has no authentication
      SPF algorithm last executed 00:08:46.616 ago
      SPF algorithm executed 4 times
      Area ranges are
      Number of LSA 5. Checksum Sum 0x036393
      Number of opaque link LSA 0. Checksum Sum 0x000000
      Number of DCbitless LSA 0
      Number of indication LSA 0
      Number of DoNotAge LSA 0
      Flood list length 0
  
```

- The OSPF Router ID set using the **router-id** {rid-ip-addr} router subcommand overrides the use of the IP address of a physical or loopback interface as the Router ID. It is recommended to set the Router ID of an OSPF router using the **router-id** router subcommand to ensure that the Router ID does not change when there is any change upon the interface IP address.
- Router IDs must be unique. The OSPF database uses the Router IDs to uniquely identify and describe every router in a network, and every router would keep a complete topology database of all routers and links in the network. Therefore, Router IDs must be unique throughout an OSPF routing domain in order for OSPF to operate correctly.
- **Note:** Each OSPF process running on a router requires a unique Router ID.  
**Ex:** When there is a single loopback interface and 2 OSPF processes running on a router, the 2nd OSPF process must select the IP address of a physical interface as its Router ID. Therefore, the IP address of a loopback interface will not always be used as the Router ID!
- The **show ip ospf interface** [intf-type intf-num | **brief**] EXEC command displays the OSPF areas associates with the interfaces, network type, cost, interface state, priority, timer values, the numbers of neighbors and adjacent neighbors along with the corresponding adjacency states.

```

RT3#sh ip ospf interface
FastEthernet0/0 is up, line protocol is up
  Internet Address 10.10.10.3/24, Area 0
  Process ID 100, Router ID 11.11.11.1, Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State DROTHER, Priority 1
  Designated Router (ID) 172.16.2.1, Interface address 10.10.10.2
  Backup Designated router (ID) 172.16.1.1, Interface address 10.10.10.1
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:05
  Index 1/1, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 0, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 2, Adjacent neighbor count is 2
    Adjacent with neighbor 172.16.1.1 (Backup Designated Router)
    Adjacent with neighbor 172.16.2.1 (Designated Router)
  Suppress hello for 0 neighbor(s)
Serial1/0 is up, line protocol is up
  Internet Address 11.11.11.1/30, Area 1
  Process ID 100, Router ID 11.11.11.1, Network Type POINT_TO_POINT, Cost: 64
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:07
  Index 1/2, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 2
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 172.16.3.1
  Suppress hello for 0 neighbor(s)
RT3#
RT3#sh ip ospf interface brief
Interface      PID   Area      IP Address/Mask    Cost  State Nbrs F/C
Fa0/0         100   0         10.10.10.3/24      1     DROTH 2/2
Se1/0         100   1         11.11.11.1/30     64    P2P   1/1
RT3#

```

- The **show ip ospf neighbor** [intf-type intf-num] [neighbor-id] [detail] EXEC command displays the information of OSPF neighbors on a per-interface basis.

```
RT3#sh ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
172.16.1.1	1	FULL/BDR	00:00:39	10.10.10.1	FastEthernet0/0
172.16.2.1	1	FULL/DR	00:00:32	10.10.10.2	FastEthernet0/0
172.16.3.1	0	FULL/ -	00:00:31	11.11.11.2	Serial1/0

```
RT3#
```

```
RT3#sh ip ospf neighbor detail
```

```
Neighbor 172.16.1.1, interface address 10.10.10.1
```

```
In the area 0 via interface FastEthernet0/0
```

```
Neighbor priority is 1, State is FULL, 6 state changes
```

```
DR is 10.10.10.2 BDR is 10.10.10.1
```

```
Options is 0x52
```

```
LLS Options is 0x1 (LR)
```

```
Dead timer due in 00:00:39
```

```
Neighbor is up for 00:09:10
```

```
Index 2/3, retransmission queue length 0, number of retransmission 1
```

```
First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
```

```
Last retransmission scan length is 1, maximum is 1
```

```
Last retransmission scan time is 0 msec, maximum is 0 msec
```

```
Neighbor 172.16.2.1, interface address 10.10.10.2
```

```
In the area 0 via interface FastEthernet0/0
```

```
Neighbor priority is 1, State is FULL, 6 state changes
```

```
DR is 10.10.10.2 BDR is 10.10.10.1
```

```
Options is 0x52
```

```
LLS Options is 0x1 (LR)
```

```
Dead timer due in 00:00:32
```

```
Neighbor is up for 00:09:07
```

```
Index 1/2, retransmission queue length 0, number of retransmission 1
```

```
First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
```

```
Last retransmission scan length is 1, maximum is 1
```

```
Last retransmission scan time is 0 msec, maximum is 0 msec
```

```
Neighbor 172.16.3.1, interface address 11.11.11.2
```

```
In the area 1 via interface Serial1/0
```

```
Neighbor priority is 0, State is FULL, 6 state changes
```

```
DR is 0.0.0.0 BDR is 0.0.0.0
```

```
Options is 0x52
```

```
LLS Options is 0x1 (LR)
```

```
Dead timer due in 00:00:31
```

```
Neighbor is up for 00:08:58
```

```
Index 1/1, retransmission queue length 0, number of retransmission 1
```

```
First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
```

```
Last retransmission scan length is 1, maximum is 1
```

```
Last retransmission scan time is 0 msec, maximum is 0 msec
```

```
RT3#
```

- The **show ip ospf database** EXEC command consists of many arguments that display the overview, summary, and detail information of the OSPF LSDB on a router.

```

RT3#sh ip ospf database

                OSPF Router with ID (11.11.11.1) (Process ID 100)

                Router Link States (Area 0)

Link ID        ADV Router    Age           Seq#           Checksum Link count
11.11.11.1    11.11.11.1    516          0x80000003    0x00DCCD 1
172.16.1.1    172.16.1.1    528          0x80000002    0x00E6BF 2
172.16.2.1    172.16.2.1    527          0x80000002    0x00F7AA 2

                Net Link States (Area 0)

Link ID        ADV Router    Age           Seq#           Checksum
10.10.10.2    172.16.2.1    516          0x80000002    0x00B4FF

                Summary Net Link States (Area 0)

Link ID        ADV Router    Age           Seq#           Checksum
11.11.11.0    11.11.11.1    546          0x80000001    0x005D5F
172.16.3.0    11.11.11.1    526          0x80000001    0x0060B9

                Router Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum Link count
11.11.11.1    11.11.11.1    536          0x80000003    0x00E07B 2
172.16.3.1    172.16.3.1    535          0x80000002    0x007B72 3

                Summary Net Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum
10.10.10.0    11.11.11.1    501          0x80000003    0x0017E2
172.16.1.0    11.11.11.1    506          0x80000001    0x00FD5D
172.16.2.0    11.11.11.1    506          0x80000001    0x00F267
RT3#

```

## OSPF Authentication

- OSPF neighbor authentication (also known as **neighbor router authentication** or **route authentication**) configures OSPF routers to only participate in routing domain based on predefined passwords for preventing an OSPF router from receiving unauthorized or fraudulent routing updates from unknown sources.
- OSPF uses **null** authentication method by default, which means that no authentication and routing updates exchanged between neighbors are not authenticated. OSPF supports **simple password** (plain text) and **Message Digest 5 (MD5) Cryptographic Checksum** authentication methods. The OSPF MD5 authentication includes a non-decreasing cryptographic sequence number in every OSPF packet to protect against replay attacks. The message digest is appended to the end of an OSPF packet, and is not considered part of the packet itself. **Note:** An OSPF router with the **ip ospf authentication null** interface subcommand configured on an interface would establish adjacency and exchange routing information with another OSPF router without the configuration (through the interface).

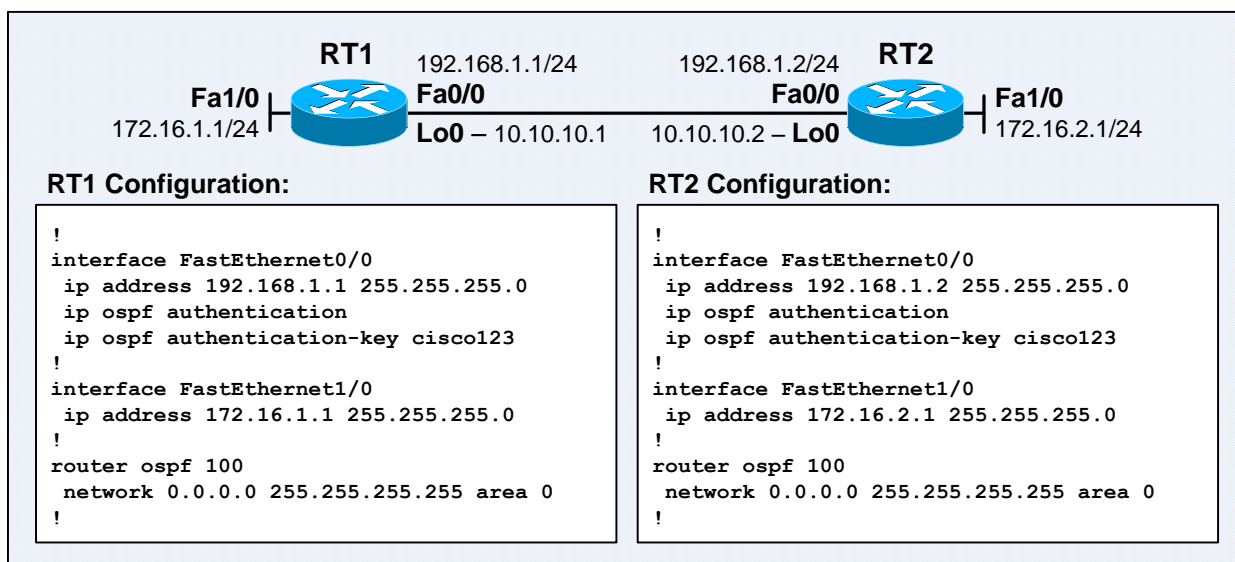


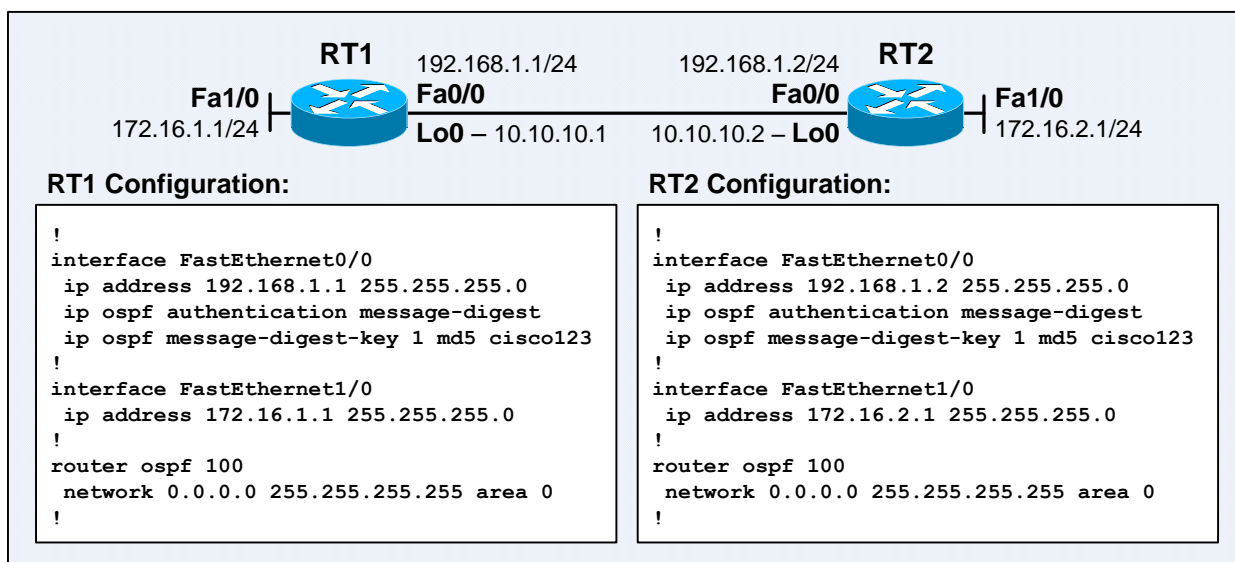
Figure 8-2: OSPF Simple Password Authentication Configuration

- The **ip ospf authentication-key** {*passwd*} interface subcommand specifies a password to be used when performing OSPF simple password authentication with neighboring routers. A router would only accept a password as a continuous string up to 8 characters. It will display the "%OSPF: Warning: The password/key will be truncated to 8 characters" warning message when the password longer than 8 characters, and uses the first 8 characters. The password is inserted directly into the Authentication Data field of the OSPF header of all OSPF packets originated from the router. All neighboring routers reside on the same network must have the same password to establish adjacencies and exchange routing information. Different passwords can be assigned to different networks as this command works on a per-interface basis. **Note:** The **service password-encryption** global configuration command would affect the appearance of the plain text password in the startup and running configuration files.
- The **ip ospf authentication** [**message-digest** | **null**] interface subcommand specifies the authentication method for an interface. Implement this command without any parameter to enable simple password authentication. This command overrides the authentication method specified using the **area** {*area-id*} **authentication** [**message-digest**] router subcommand.
- The **debug ip ospf adj** privileged command displays OSPF adjacency-related events and is useful for troubleshooting OSPF authentication. **Note:** This command does not display any OSPF authentication information but does display OSPF authentication failures if there are any. Below shows the debug messages for authentication type and authentication key mismatches.

```

XX:XX:XX: OSPF: Rcv pkt from 192.168.1.2, FastEthernet0/0 : Mismatch
Authentication type. Input packet specified type 2, we use type 1
=====
XX:XX:XX: OSPF: Rcv pkt from 192.168.1.2, FastEthernet0/0 : Mismatch
Authentication Key - Clear Text
=====
XX:XX:XX: OSPF: Rcv pkt from 192.168.1.2, FastEthernet0/0 : Mismatch
Authentication Key - Message Digest Key 1
=====
XX:XX:XX: OSPF: Rcv pkt from 192.168.1.2, FastEthernet0/0 : Mismatch
Authentication Key - No message digest key 2 on interface

```



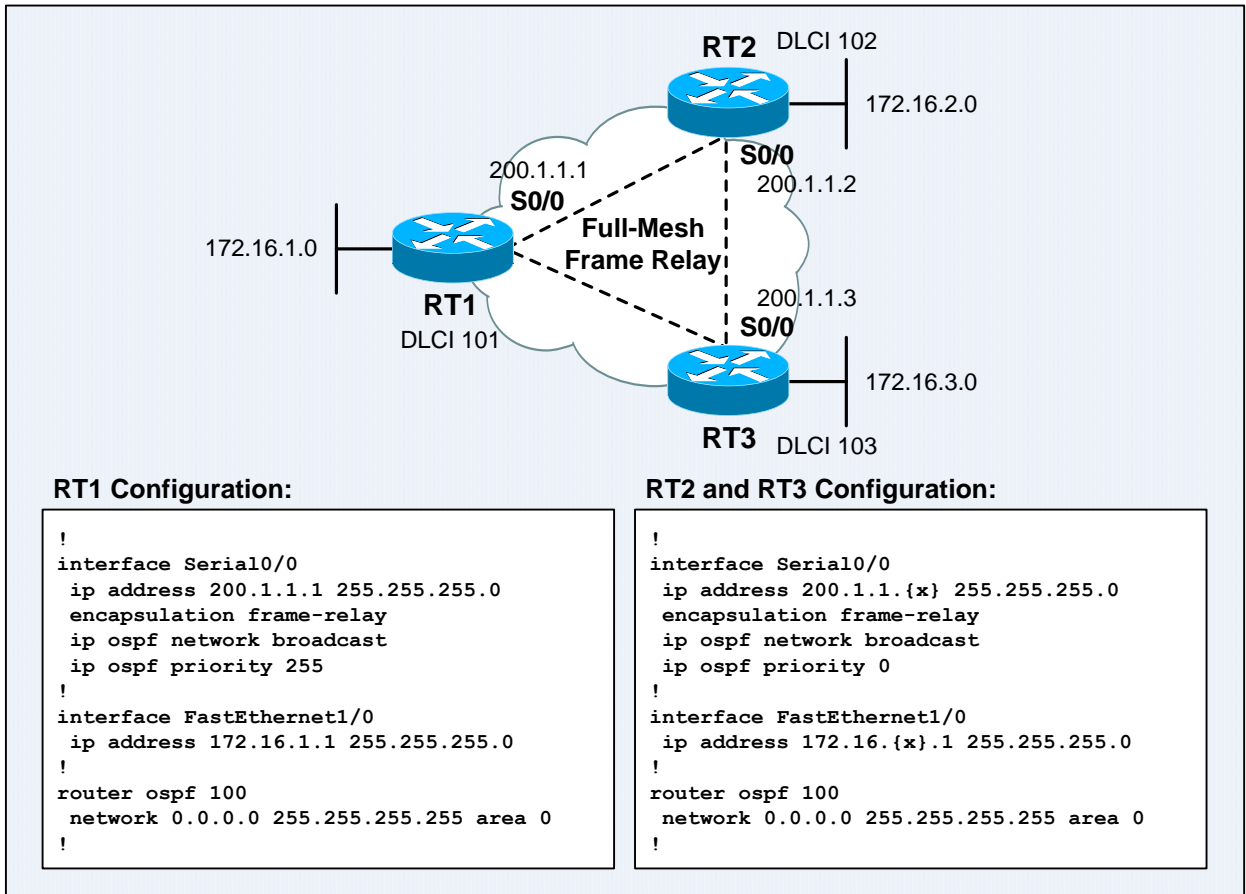
**Figure 8-3: OSPF MD5 Cryptographic Checksum Authentication Configuration**

- The **ip ospf message-digest-key {key-id} md5 {key}** interface subcommand specifies a password to be used when performing OSPF MD5 authentication with neighboring routers. The *key-id* is an identifier in the range from 1 to 255; while the key is a continuous string up to 16 characters. The *key-id* and MD5 key are used to generate and verify a message digest (or hash) that is to be appended to the end of all OSPF packets. All neighboring routers reside on the same network must have the same set of *key-id* and MD5 key to establish adjacencies and exchange routing information. Different MD5 keys can be assigned to different networks as this command works on a per-interface basis.  
**Note:** The **service password-encryption** global configuration command would affect the appearance of the MD5 key in the startup and running configuration files.
- The cryptographic sequence number increases at a rate of 1 per second. The Cisco implementation of OSPF uses the **UNIX epoch** as the reference point, which defines the number of seconds elapsed since **1st Jan 1970 (UTC)**. [www.epochconverter.com](http://www.epochconverter.com) can be used to find out the exact time that an MD5-authenticated OSPF packet that has been generated by a Cisco router. The 32-bit cryptographic sequence number field will rollover on 6:28:15 7th Feb 2106 (UTC).
- The *key-id* allows uninterrupted transitions between MD5 keys and therefore allows the changing of OSPF password without interrupting the OSPF adjacencies between routers. When an interface is configured with a new MD5 key, it would send multiple copies of the same OSPF packet with different MD5 keys. The router would stop sending duplicate packets when it received OSPF packets from neighbors that were authenticated using the new MD5 key and concluded that all its neighbors have adopted the new MD5 key when. Once all neighbors have been updated with the new MD5 key, the old MD5 key should be removed to prevent the router from communicating with hostile systems that knows the old MD5 key.
- The **show ip ospf interface [intf-type intf-num] EXEC** command displays the OSPF authentication method configured for an interface.

```
Router#sh ip ospf int fa0/0 | in authentication
Simple password authentication enabled
Router#sh ip ospf int fa1/0 | in authentication
Message digest authentication enabled
```



## OSPF Configuration over NBMA Environment



**Figure 8-4:** OSPF Broadcast Mode over Full-Mesh Frame Relay Network with One IP Subnet

- Below shows the OSPF neighbors and IP routing table on RT1:

```

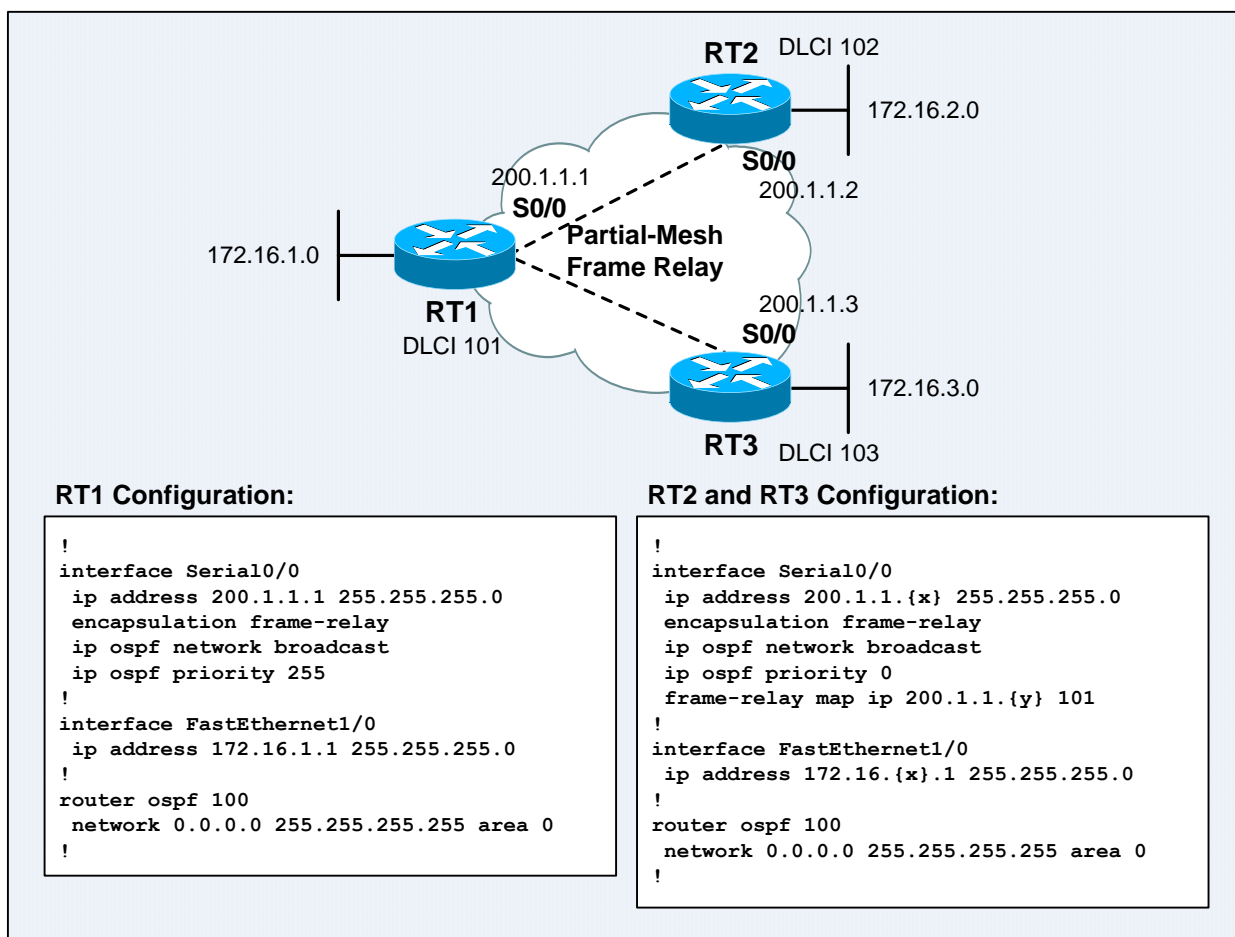
RT1#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address        Interface
200.1.1.2        0     FULL/DROTHER    00:00:39   200.1.1.2     Serial0/0
200.1.1.3        0     FULL/DROTHER    00:00:33   200.1.1.3     Serial0/0
RT1#
RT1#sh ip route

Gateway of last resort is not set

C    200.1.1.0/24 is directly connected, Serial0/0
    172.16.0.0/24 is subnetted, 3 subnets
C    172.16.1.0 is directly connected, FastEthernet1/0
O    172.16.2.0 [110/65] via 200.1.1.2, 00:01:00, Serial0/0
O    172.16.3.0 [110/65] via 200.1.1.3, 00:01:00, Serial0/0
RT1#

```



**Figure 8-5: OSPF Broadcast Mode over Partial-Mesh Frame Relay Network with One IP Subnet**

- The **frame-relay map ip {ip-addr} {dlci}** interface subcommand is required on spoke routers as there are no PVCs between spoke routers. The **broadcast** keyword is optional for this setup.
- Below shows the OSPF neighbors and IP routing table on RT2:

```

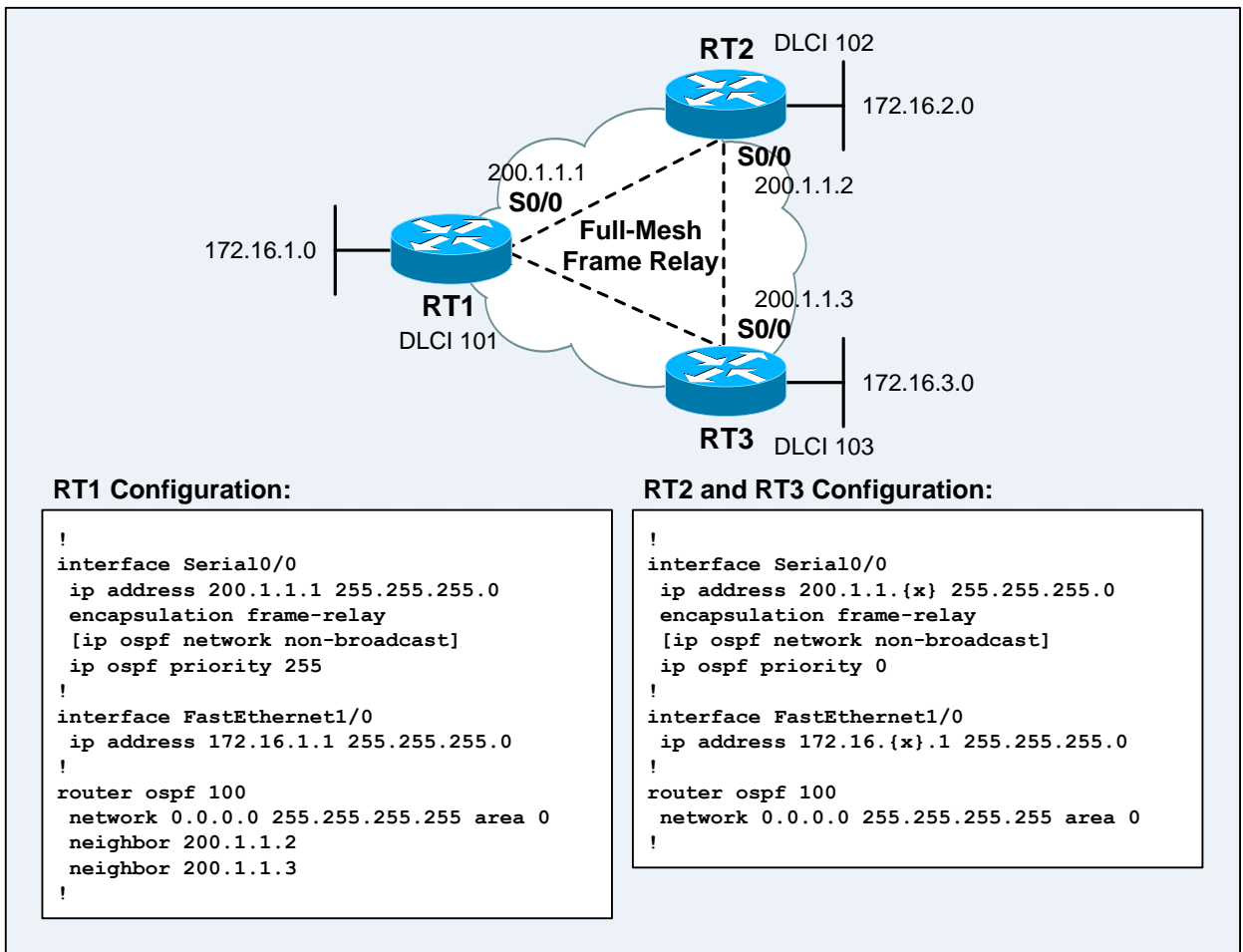
RT2#sh frame-relay map
Serial0/0 (up): ip 200.1.1.1 dlci 101(0x65,0x1850), dynamic,
                broadcast,, status defined, active
Serial0/0 (up): ip 200.1.1.3 dlci 101(0x65,0x1850), static,
                CISCO, status defined, active
RT2#
RT2#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address        Interface
200.1.1.1        255   FULL/DR         00:00:39   200.1.1.1     Serial0/0
RT2#
RT2#sh ip route

Gateway of last resort is not set

C    200.1.1.0/24 is directly connected, Serial0/0
     172.16.0.0/24 is subnetted, 3 subnets
O    172.16.1.0 [110/65] via 200.1.1.1, 00:00:17, Serial0/0
C    172.16.2.0 is directly connected, FastEthernet1/0
O    172.16.3.0 [110/65] via 200.1.1.3, 00:00:17, Serial0/0
RT2#

```



**Figure 8-6:** OSPF Non-Broadcast Mode over Full-Mesh Frame Relay Network with One IP Subnet

- Below shows the OSPF neighbors on RT1, RT2, and RT3:

```

RT1#sh ip ospf neighbor

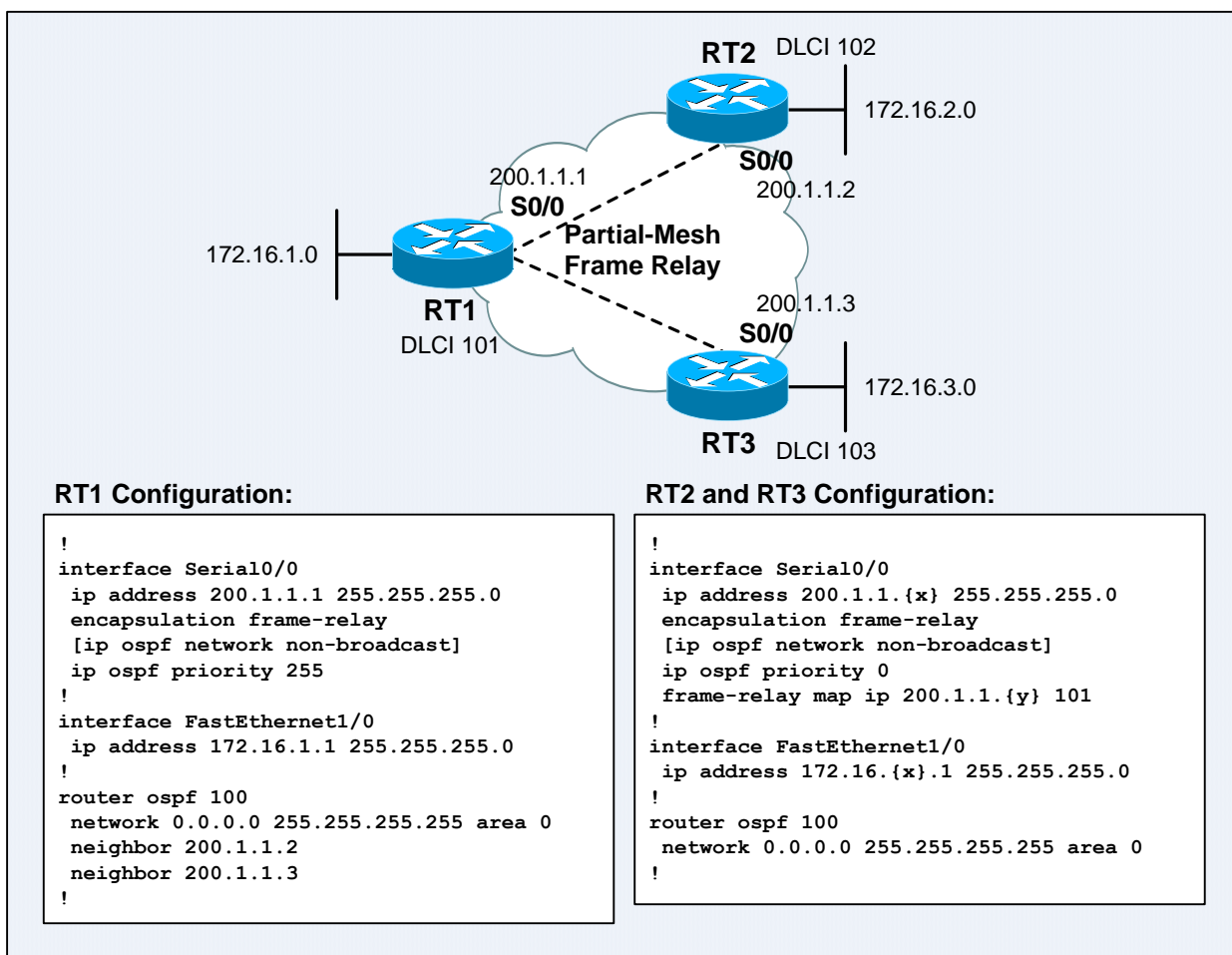
Neighbor ID      Pri   State           Dead Time   Address      Interface
N/A              0     ATTEMPT/DROTHER 00:01:42   200.1.1.2   Serial0/0
N/A              0     ATTEMPT/DROTHER 00:01:42   200.1.1.3   Serial0/0
RT1#
00:07:04: %OSPF-5-ADJCHG: Process 100, Nbr 200.1.1.2 on Serial0/0 from
LOADING to FULL, Loading Done
00:07:04: %OSPF-5-ADJCHG: Process 100, Nbr 200.1.1.3 on Serial0/0 from
LOADING to FULL, Loading Done
RT1#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address      Interface
200.1.1.2       0     FULL/DROTHER    00:01:45   200.1.1.2   Serial0/0
200.1.1.3       0     FULL/DROTHER    00:01:45   200.1.1.3   Serial0/0
RT1#
=====
RT2#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address      Interface
200.1.1.1       255   FULL/DR         00:01:55   200.1.1.1   Serial0/0
RT2#
=====
RT3#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address      Interface
200.1.1.1       255   FULL/DR         00:01:47   200.1.1.1   Serial0/0
RT3#

```



**Figure 8-7:** OSPF Non-Broadcast Mode over Partial-Mesh Frame Relay Network with One IP Subnet

- The **frame-relay map ip {ip-addr} {dlci}** interface subcommand is required on spoke routers as there are no PVCs between spoke routers. The **broadcast** keyword is optional for this setup.
- Below shows the OSPF neighbors on RT1, RT2, and RT3:

```

RT1#sh ip ospf neighbor

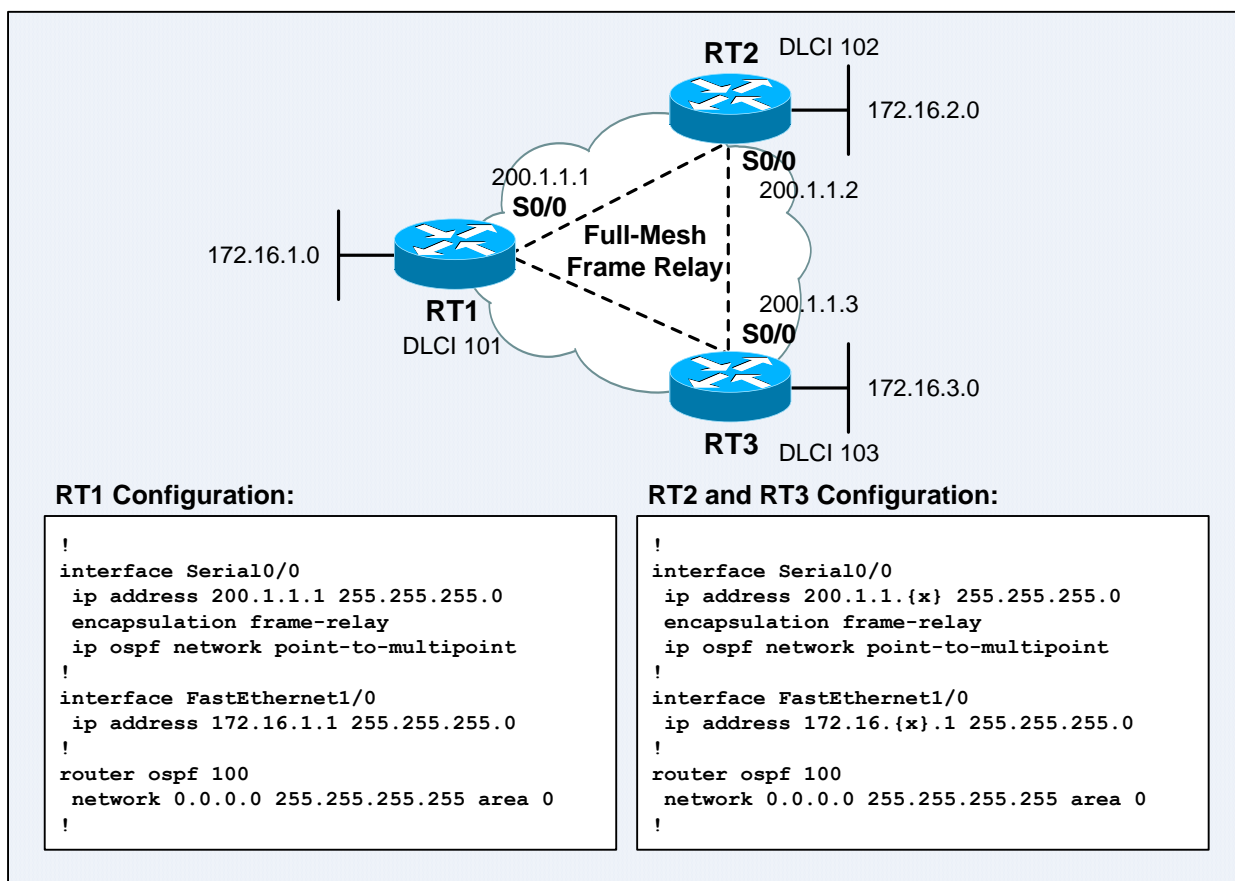
Neighbor ID      Pri   State           Dead Time   Address      Interface
200.1.1.2        0     FULL/DROTHER    00:01:55   200.1.1.2   Serial0/0
200.1.1.3        0     FULL/DROTHER    00:01:55   200.1.1.3   Serial0/0
RT1#
=====
RT2#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address      Interface
200.1.1.1        255   FULL/DR         00:01:58   200.1.1.1   Serial0/0
RT2#
=====
RT3#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address      Interface
200.1.1.1        255   FULL/DR         00:01:42   200.1.1.1   Serial0/0
RT3#

```

- The **neighbor** {*ip-addr*} [**cost** *num*] [**database-filter all out**] [**poll-interval** *sec*] [**priority** *num*] router subcommand specifies a neighboring OSPF router that connects to a NBMA network. The **priority** option specifies the router priority for a neighboring router. The default priority value is 0, which designates that the neighboring routers as a DROTHER. **Note:** The priority configured with the **ip ospf priority** interface subcommand on spoke router would override the priority configured with the **neighbor** router subcommand on the hub router.
- When a neighboring router has become inactive or dead (Hello packets have not been received for RouterDeadInterval), it may still be necessary to send Hello packets to the dead neighbor. These Hello packets will be sent at the reduced rate – PollInterval, which should be much larger than HelloInterval. The default PollInterval value is 120 seconds (2 minutes).
- The interval between Hello packets is determined by the state of the neighbor. If the neighbor is in the DOWN state, Hello packets are sent every PollInterval; otherwise, the Hello packets are sent every HelloInterval.
- When an OSPF router first starts up, it sends only Hello packets to other routers with non-zero priority – routers that are eligible to become DR and BDR. After the DR election, DR and BDR will then start sending Hello packets to all other neighboring routers in order to form adjacencies.
- The **cost** option assigns a cost to the link to a neighboring router to reflect the link bandwidth. The **database-filter all out** keyword filters outgoing LSAs to a neighboring router.
- The **poll-interval** and **priority** options are allowed only for **non-broadcast** interface type; while the **cost** and **database-filter** options are allowed only for **point-to-multipoint** and **point-to-multipoint non-broadcast** interface types.
- In **non-broadcast** mode, the **neighbor** statements are required only on the DR and BDR. In a hub-and-spoke partial-mesh topology, the **neighbor** statements are required on the hub router, which must also be configured to become a DR by being assigned a higher priority; while the **neighbor** statements are not required on the spoke routers. In a full-mesh topology, the **neighbor** statements are required on all routers unless the DR and BDR are manually configured using the **ip ospf priority** interface subcommand.
- In **point-to-multipoint** mode, the DR and BDR are not elected. Multicast Hello packets are used to automatically discover the neighbors, and manual neighbor configuration is not required. The Network-LSAs are not exchanged between neighboring routers; instead, routers exchange additional LSAs to describe the underlying network topology (be it full-mesh or partial-mesh).
- The **point-to-multipoint** mode reduces the number of VCs required for complete connectivity, as it does not require a full-mesh topology. It allows routing between routers that are not directly connected via a VC but are connected through a (hub) router that has VCs connected to them.
- The **point-to-multipoint** mode treats a NBMA network as a collection of point-to-point links rather than a multi-access network, which is not suitable for networks with dynamic connections, in which SVCs are being used instead of PVCs. The Cisco-proprietary **point-to-multipoint non-broadcast** extension mode should be used instead.
- **Note:** Cisco routers only support Frame Relay SVC in Data Terminal Equipment (DTE) mode. The Cisco Frame Relay switching feature does not support SVC. Due to the limitation of hardware availability and limited deployment of SVCs in current networking industry, the configuration for **point-to-multipoint non-broadcast** mode will not be discussed.



**Figure 8-8:** OSPF Point-to-Multipoint Mode over Full-Mesh Frame Relay Network with One IP Subnet

- Below shows the OSPF neighbors, IP routing table, and OSPF link-state database on RT1:

```

RT1#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address        Interface
200.1.1.3        0     FULL/ -         00:01:48    200.1.1.3     Serial0/0
200.1.1.2        0     FULL/ -         00:01:51    200.1.1.2     Serial0/0
RT1#
RT1#sh ip route

Gateway of last resort is not set

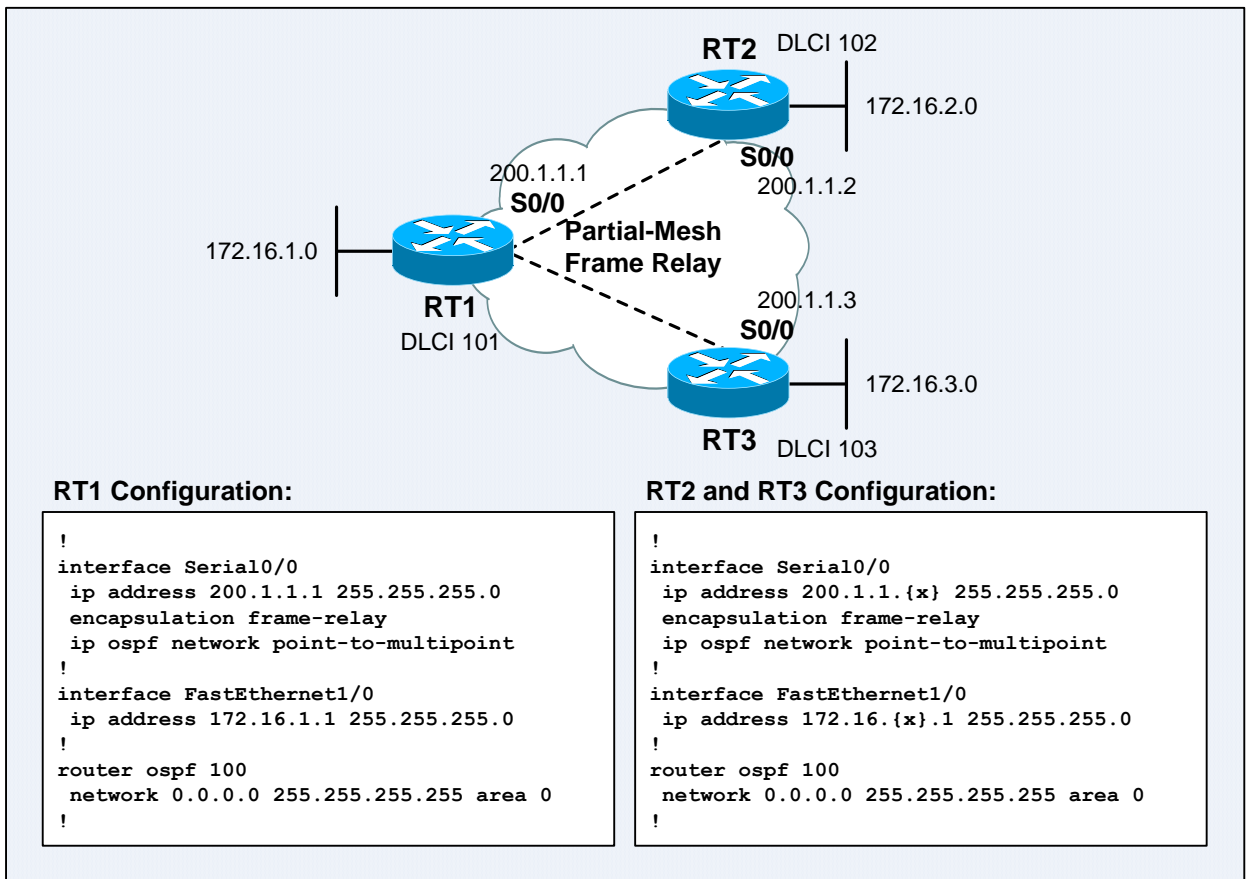
    200.1.1.0/24 is variably subnetted, 3 subnets, 2 masks
C       200.1.1.0/24 is directly connected, Serial0/0
O       200.1.1.2/32 [110/64] via 200.1.1.2, 00:00:05, Serial0/0
O       200.1.1.3/32 [110/64] via 200.1.1.3, 00:00:05, Serial0/0
    172.16.0.0/24 is subnetted, 3 subnets
C       172.16.1.0 is directly connected, FastEthernet1/0
O       172.16.2.0 [110/65] via 200.1.1.2, 00:00:05, Serial0/0
O       172.16.3.0 [110/65] via 200.1.1.3, 00:00:05, Serial0/0
RT1#
RT1#sh ip ospf database

                OSPF Router with ID (200.1.1.1) (Process ID 100)

                Router Link States (Area 0)

Link ID        ADV Router    Age         Seq#          Checksum Link count
200.1.1.1      200.1.1.1    32         0x80000003   0x00AB84 4
200.1.1.2      200.1.1.2    32         0x80000002   0x00F239 4
200.1.1.3      200.1.1.3    27         0x80000003   0x0042E3 4
RT1#

```



**Figure 8-9: OSPF Point-to-Multipoint Mode over Partial-Mesh Frame Relay Network with One IP Subnet**

- **Note:** The **frame-relay map ip {ip-addr} {dcli}** interface subcommand is NOT required on spoke routers even there are no PVCs between spoke routers.
- Below shows the OSPF neighbors on RT1, RT2, and RT3:

```

RT1#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address      Interface
200.1.1.3        0    FULL/ -         00:01:45   200.1.1.3   Serial0/0
200.1.1.2        0    FULL/ -         00:01:45   200.1.1.2   Serial0/0
RT1#
=====

RT2#sh ip ospf neighbor

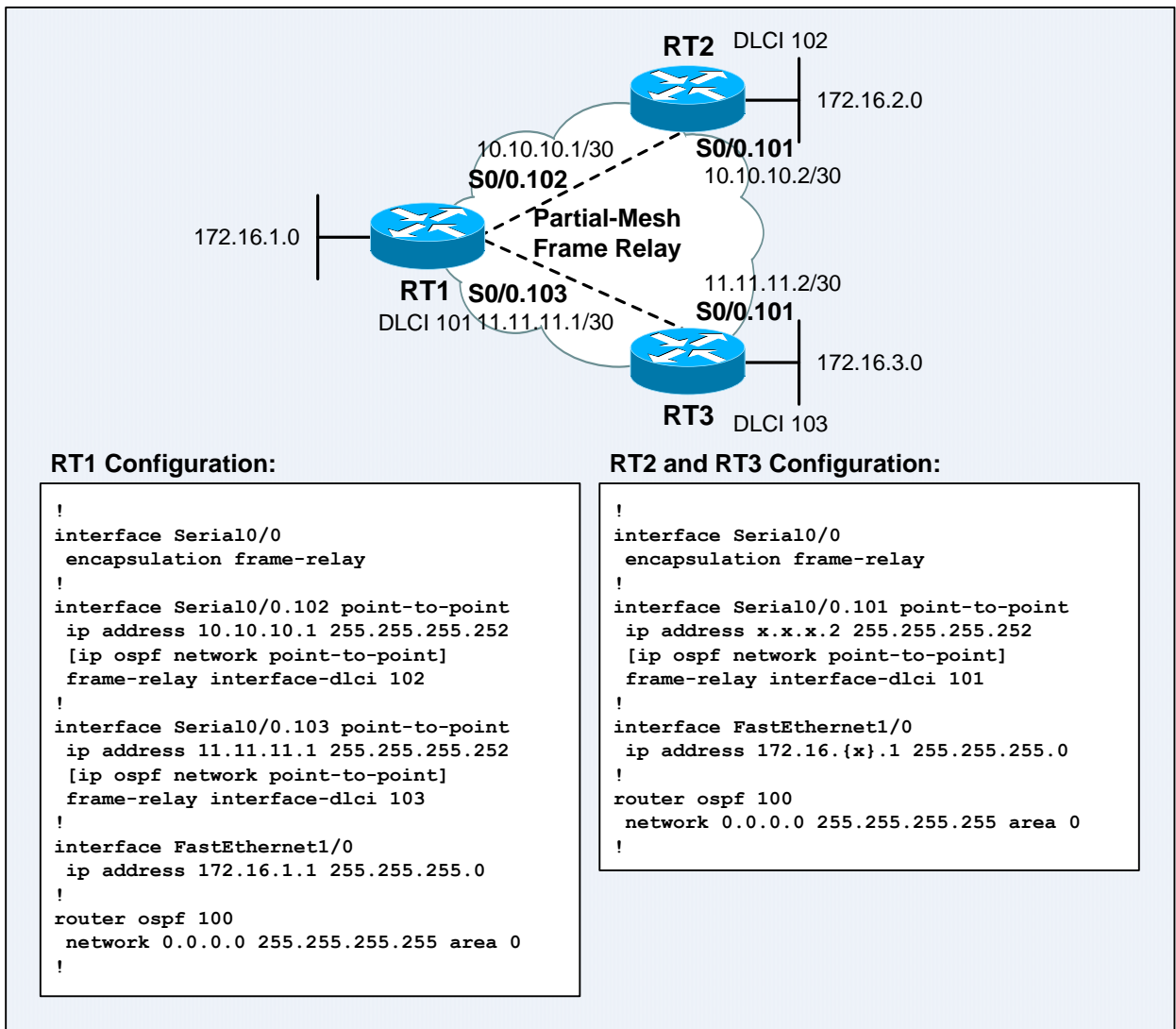
Neighbor ID      Pri   State           Dead Time   Address      Interface
200.1.1.1        0    FULL/ -         00:01:40   200.1.1.1   Serial0/0
RT2#
=====

RT3#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address      Interface
200.1.1.1        0    FULL/ -         00:01:58   200.1.1.1   Serial0/0
RT3#

```

- **Note:** The network setup above is operational when Serial0/0 on RT2 and RT3 is configured with **ip ospf network point-to-point**. However, the **ip ospf hello-interval 10** interface subcommand must be configured on RT1 Serial0/0 or else the adjacencies will not be established due to mismatched Hello parameters – 10 seconds for P2P and 40 seconds for NBMA.



**Figure 8-10: OSPF Point-to-Point Mode over Partial-Mesh Frame Relay Network with One IP Subnet Per VC**

- Below shows the OSPF neighbors on RT1, RT2, and RT3:

```

RT1#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address        Interface
172.16.3.1       0     FULL/-          00:00:36   11.11.11.2    Serial0/0.103
172.16.2.1       0     FULL/-          00:00:34   10.10.10.2    Serial0/0.102
RT1#
=====
RT2#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address        Interface
172.16.1.1       0     FULL/-          00:00:33   10.10.10.1    Serial0/0.101
RT2#
=====
RT3#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address        Interface
172.16.1.1       0     FULL/-          00:00:34   11.11.11.1    Serial0/0.101
RT3#

```



## OSPF over Unnumbered Link

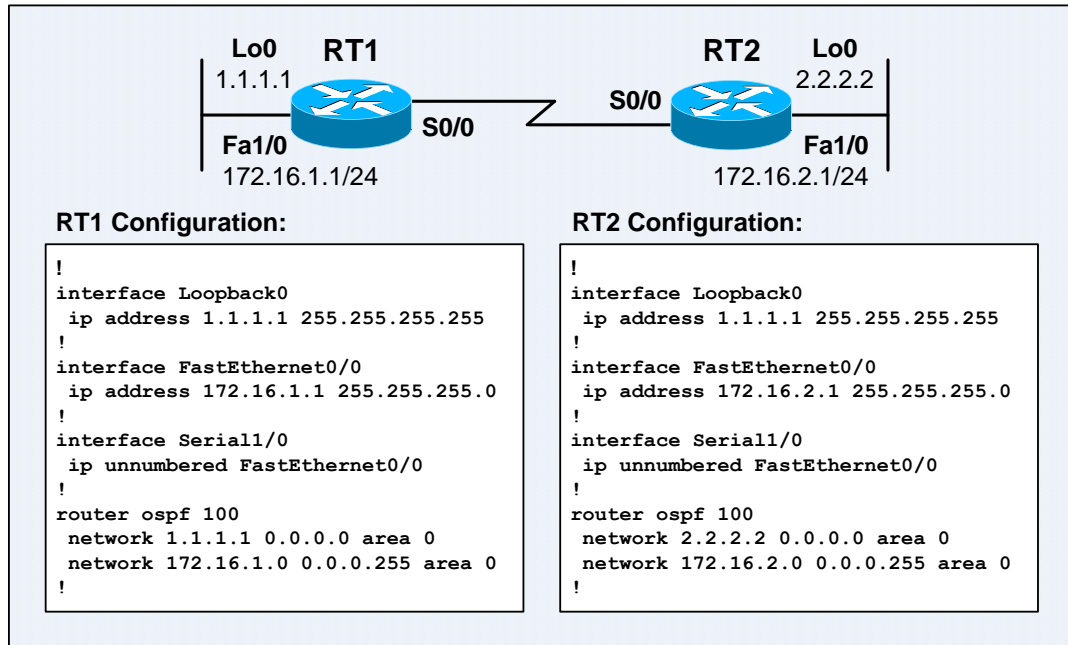


Figure 8-11: OSPF over Unnumbered Link

- The only time that OSPF forms adjacency between neighbors that do not reside on the same subnet is when the neighbors are connected through an unnumbered point-to-point link. When enabling OSPF on an unnumbered interface, use the address-wildcard-mask pair of the primary physical interface which the unnumbered interface relies upon.  
**Note:** OSPF does not work if one side is unnumbered while the other side is numbered on a link, as this would create a discrepancy in the OSPF database and prevent routes from being installed into the routing table.
- Below shows the OSPF neighbors, IP routing table, and OSPF database on RT1:

```

RT1#sh ip ospf neighbor
Neighbor ID      Pri   State           Dead Time   Address        Interface
2.2.2.2          0    FULL/-         00:00:33   172.16.2.1    Serial0/0
RT1#
RT1#sh ip route
Gateway of last resort is not set

  1.0.0.0/32 is subnetted, 1 subnets
C       1.1.1.1 is directly connected, Loopback0
  2.0.0.0/32 is subnetted, 1 subnets
O       2.2.2.2 [110/65] via 172.16.2.1, 00:00:34, Serial0/0
 172.16.0.0/24 is subnetted, 2 subnets
C       172.16.1.0 is directly connected, FastEthernet1/0
O       172.16.2.0 [110/65] via 172.16.2.1, 00:00:34, Serial0/0
RT1#
RT1#sh ip ospf database

                OSPF Router with ID (1.1.1.1) (Process ID 100)

                Router Link States (Area 0)

Link ID        ADV Router    Age         Seq#          Checksum Link count
1.1.1.1        1.1.1.1      58         0x80000002   0x00C835 3
2.2.2.2        2.2.2.2      61         0x80000001   0x00B73E 3
RT1#

```

## Interconnecting Multiple OSPF Areas

- In large networks which the web of router links is often very complex due to the large number of potential paths to all subnets, the SPF calculations that considering all the possible paths would be very complex and consume significant time and router resources.
- OSPF is able to reduce the effort of SPF calculations by partitioning a network into areas. With smaller number of routers and LSAs that flood within an area, the link-state database would become smaller, in which the SPF calculations are relatively easier and faster. Additionally, the flooding of information upon a router or link failure only affects the routers within an area; routers outside the area will not receive the info hence do not need to perform SPF calculation. Hierarchical routing is a often being used when building large networks.
- OSPF implements a 2-level hierarchy routing scheme which comprises of the following area types:

<b>Backbone or Transit area</b>	Provides fast and efficient forwarding of IP packets between standard areas. Backbone area distributes routing information between standards areas. End systems are not found within the backbone or transit area.
<b>Standard or Regular area</b>	Connects end systems and resources. Standard areas (non-backbone areas) do not allow traffic from other areas to pass through it. All traffic from other areas must transmit across a transit area. It includes a number of subtypes, namely <b>stub area</b> , <b>totally stubby area</b> , and <b>not-so-stubby (NSSA) area</b> .

Furthermore, OSPF allows an autonomous system to be split into additional layers or levels by incorporating 2 levels of external routing (E1 and E2 external routes) into the OSPF routing domain.

- LS routing protocols require all routers to maintain a copy of the LSDB; the more OSPF routers, the larger the LSDB. It would be nice to have all information of all routers, but this approach has scalability issues for larger networks. With the concept of **area**, routers inside an area only required to maintain detailed information about the routers and links within the area and summary information about routers and links in other areas. By implementing a hierarchical structure using areas, an OSPF autonomous system or routing domain can scale very large. **Note:** Cisco recommends that generally there should be no more than 50 routers per area.
- The Internet is split into autonomous systems with the boundaries of network segments. For OSPF, the routers are the area boundaries. This means that a network segment would belong to only one area, which eases the aggregation of network addresses across area boundaries.
- The **abstraction** process summarizes a list of network addresses within an area when advertising the information of the area to other areas. The information is being passed from area to area, as like routes that are being advertised from router to router with the Distance Vector algorithm. Although the DV algorithm is used between areas, OSPF was designed to avoid the common pitfalls of DV algorithm by requiring that all areas must be directly connected to a special area known as the backbone areas, which results in a hub-and-spoke organization that DV algorithm is able to handle without any problem. The DV algorithm between areas will be discussed later.
- The restriction that requires all areas to be directly connected to the backbone area could be compromised through the use of **virtual links**, which are logical extensions of the backbone area. Virtual links tunnel routing information through areas which create a logical hub-and-spoke area topology regardless of the physical area topology.

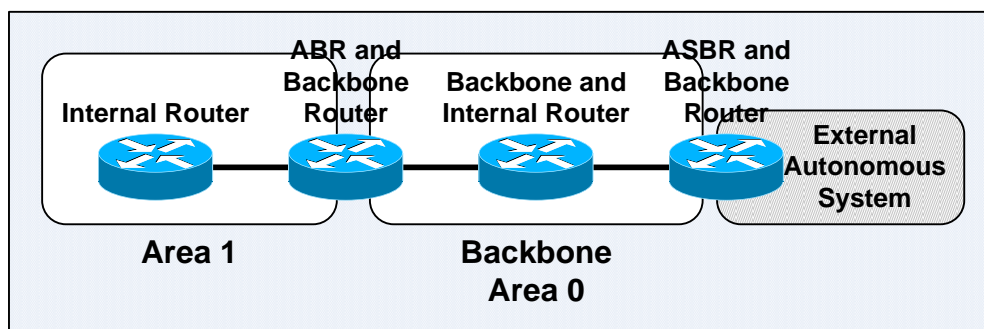


Figure 9-1: OSPF Router Types

- Below describes the various types of OSPF routers:

<b>Internal Router</b>	An OSPF router in which all its interfaces reside within a single area. All routers within the same area maintain the same LSDB for the area.
<b>Backbone Router</b>	An OSPF router in which all its interfaces reside within the backbone area or has at least one interface resides in the backbone area.
<b>Area Border Router (ABR)</b>	An OSPF router that has interfaces reside in the backbone area and non-backbone areas. An ABR maintains separate (complete) LSDBs for all areas that it resides, route packets between the areas, and separates the LSA flooding zones – LSA flooding would stop at the area boundary. An ABR is the primary point of routing information exchange for an area, distributes the routing information of a standard area into the backbone; the backbone routers propagate the info to other ABRs and eventually internal routers of other areas. An ABR can be configured to summarize / filter the routing information in its LSDB before sending it to other areas. In a multiarea OSPF network, an area can have multiple ABRs.
<b>Autonomous System Boundary Router (ASBR)</b>	An OSPF router that has at least one interface connects to an external internetwork – another autonomous system or non-OSPF network. ASBRs can perform <b>route redistribution</b> to import / export non-OSPF network information between the OSPF routing domain and external internetwork.

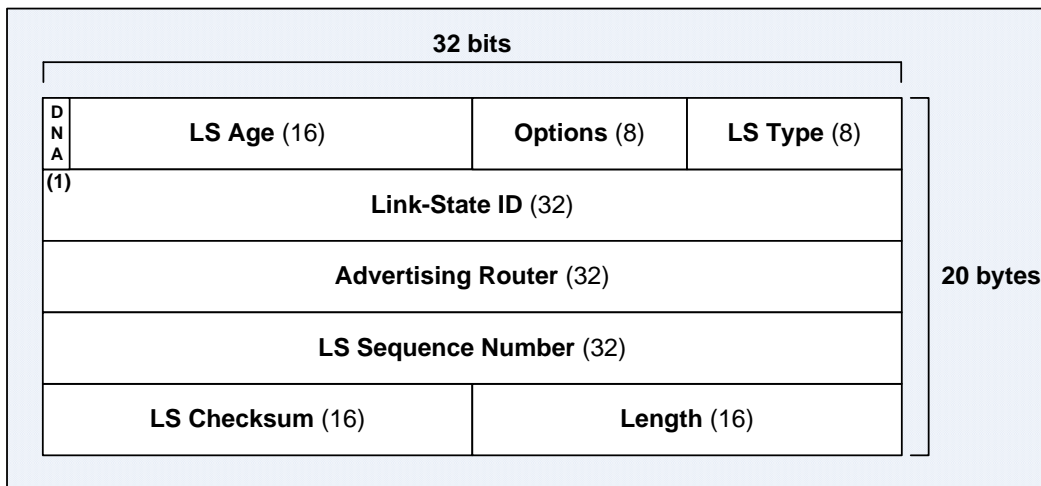
**Note:** A router can be more than one router type. Ex: A router which connects to area 0, area 1, and a non-OSPF network is a backbone router, an ABR, and an ASBR.

- Below are some OSPF design guidelines based on real-world studies and implementations:

	Min	Mean	Max
<b>Routers in a domain</b>	20	510	1000
<b>Routers in an area</b>	20	160	350
<b>Areas in a domain</b>	1	23	60

- An area should have no more than 50 routers.
- A router should have no more than 60 OSPF neighbors.
- A router should not be in more than 3 areas. The ideal design is to have an ABR connects to 2 areas only – the backbone and a standard area.
- A router can be a DR or BDR for more than 1 network segment.
- Do not run more than 1 OSPF process on an ABR.

## OSPF LSA Types



**Figure 9-2: OSPF LSA Header Format**

- All types of LSA have a 20-byte common LSA header as illustrated in the figure above. The **Link-State Advertisement (LSA)** is the building block of the OSPF link-state database (LSDB). Its primary function is to advertise a router's routing topology to other routers. Since OSPF is designed for scalability, some types of LSAs are not flooded out on all interfaces, instead only on those appropriate interfaces. As a result, detailed information can be localized to an area, while summary information is flooded to other areas of the network.
- Below describes the various types of **Link-State Advertisements (LSAs)**:

LSA Type	Description
1	<b>Router-LSA.</b> Originated from every router to other routers in the same area about the information of its directly connected links to other routers and networks. Type-1 LSAs are flooded within the originating area only. The Link-State ID of a Type-1 LSA is the Router ID of the originating router. The Link-State ID and Advertising Router fields in the LSA header are identical for Router-LSAs.
2	<b>Network-LSA.</b> Originated from the DR on a broadcast or multi-access network (eg: Ethernet) to list all the routers that attached to the <b>transit network</b> . Type-2 LSAs are flooded within the originating area only. The Link-State ID of a Type-2 LSA is the physical interface IP address of the DR that connects to the network and advertises the LSA.
3	<b>Network-Summary-LSA.</b> Originated from an ABR to advertise the subnets in an area (omitting the topological data) to neighboring routers outside the area. ABRs can be configured to summarize and filter the routing information in its LSDBs before sending it to other areas. Route summarization and filtering on ABRs provide greater scalability for OSPF. The Link-State ID of a Type-3 LSA is the IP address of the subnet that is being advertised.
4	<b>ASBR-Summary-LSA.</b> Originated from an ABR that resides within the same area as the ASBR to identify the ASBR and describes the route to the ASBR. Type-5 LSAs are flooded to all areas but the detailed information to reach the ASBR may not be available throughout all areas. This problem is solved by having an ABR to describe the route to the ASBR that originated the Type-5 LSA. The Link-State ID of a Type-4 LSA is the Router ID of the ASBR that is being advertised.

5	<b>AS-External-LSA.</b> Originated from an ASBR and contains information imported into an OSPF routing domain from other routing processes. Type-5 LSAs are flooded throughout all areas except stubby areas, totally stubby areas, and NSSAs. When a stub router receives a Type-5 LSA from another router within the stub area, the LSA will be rejected. The Link-State ID of a Type-5 LSA is the IP address of the external subnet that is being advertised.
6	<b>Group-Membership-LSA.</b> Defined for Multicast extensions to OSPF (MOSPF), a multicast routing protocol that does not receive wide acceptance. <b>Note:</b> Cisco IOS does not support MOSPF, mainly due to the reasons that it uses a dense-mode multicast forwarding scheme and is protocol dependent. A Cisco router would generate a %OSPF-4-BADLSATYPE error message upon receiving a Type-6 LSA. The <b>ignore lsa mospf</b> router subcommand configures an OSPF router to ignore Type-6 LSAs and therefore prevents the router from generating the error message.
7	<b>NSSA-External-LSA.</b> ASBRs in a Not-So-Stubby-Area (NSSA) do not receive Type-5 LSAs from ABRs within the NSSA, but are allowed to advertise external routing information upon route redistribution. They advertise Type-7 LSAs into the NSSAs about the external routes, which the NSSA ABRs then translate to Type-5 LSAs and flood normally throughout all areas except stubby areas, totally stubby areas, and NSSAs. Type-7 LSAs are flooded within an NSSA only.
8	<b>External-Attributes-LSA.</b> Type-8 OSPFv2 LSAs are being used as an alternative to Internal BGP (IBGP) for OSPF and BGP internetworking to carry BGP path attributes across an OSPF routing domain; while Type-8 OSPFv3 link-local scope LSAs advertise information about link-local IPv6 addresses on a link or segment. <b>Note:</b> Cisco IOS does not support Type-8 LSAs.
9, 10, 11	<b>Opaque LSAs</b> (RFC 2370 – The OSPF Opaque LSA Option) are designated to allow the future upgrades or extensions to OSPF for application-specific purposes. An Opaque LSA consists of a standard LSA header followed by information field, which may be used directly by OSPF or other applications. The standard OSPF LSA flooding mechanism distributes Opaque LSAs to all or limited portion of an OSPF routing domain – Opaque LSAs have different flooding scopes. <b>Type-9 Link-Local-Opaque-LSA.</b> Type-9 LSAs are flooded within a local link. <b>Type-10 Area-Local-Opaque-LSA.</b> Type-10 LSAs are typically used for Traffic Engineering extensions to OSPF (RFC 3630) to advertise extra link information besides the metric, namely maximum bandwidth, maximum reservable bandwidth, unreserved bandwidth, and administrative group. Type-10 LSAs are flooded within an area by any OSPF router – both TE capable and non-TE capable routers. <b>Type-11 Autonomous-System-Opaque-LSA.</b> Type-11 LSAs are flooded throughout all areas except stub areas – equivalent to Type-5 LSAs. <b>Note:</b> Type-9 through Type-11 Opaque LSAs that are only defined in OSPFv3. Additionally, Opaque LSAs are not used for route calculation but are used for MPLS Traffic Engineering.

- The LS Age field is often less than 1800. LSA checksum is performed upon an LSA excluding the LS Age field of the LSA. The Length field indicates the length of an LSA including the 20-byte LSA header.
- Type-1 and Type-2 LSAs are found in all areas, and are never flooded across an area boundary. Whether the other types of LSAs are flooded within an area depends on the area types, eg: standard area, backbone area, stub area, totally stubby area, and not-so-stubby area (NSSA).
- There is no Type-4 and Type-5 LSAs when there is no ASBR in an OSPF routing domain.

## Type-1 Router-LSA

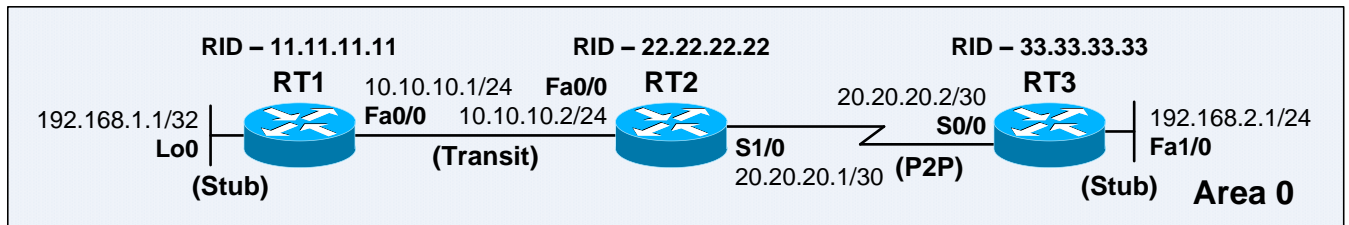


Figure 9-3: Network Setup for Type-1 Router-LSA and Type-2 Network-LSA

- Type-1 Router-LSAs are originated from every router to other routers in the same area about the status information of its directly connected links (interfaces) to other routers and networks. Type-1 LSAs are flooded within the originating area only and never crosses an area boundary. The Link-State ID of a Type-1 Router-LSA is the Router ID of the originating router. The Link-State ID and Advertising Router fields in the LSA header are same for Router-LSAs.

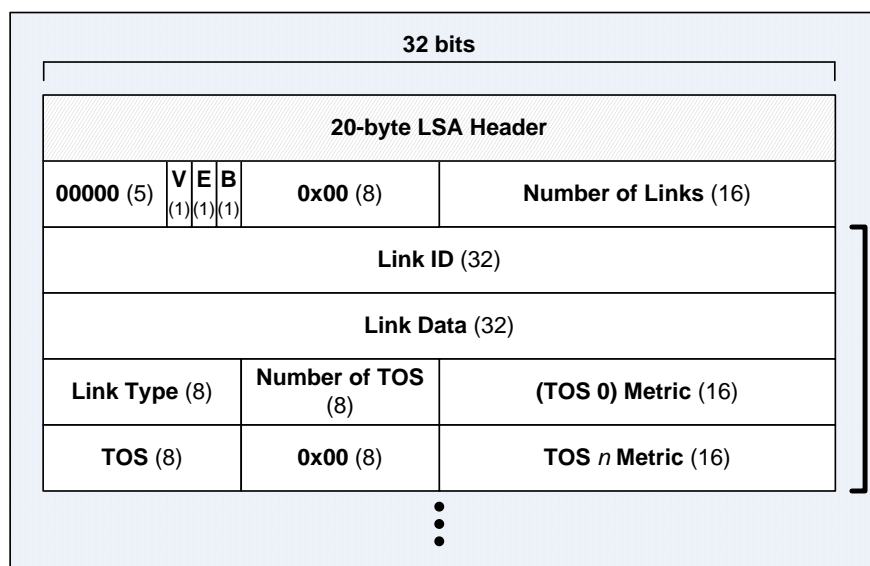


Figure 9-4: OSPF Type-1 Router-LSA Format

- Below lists the fields in the OSPF Type-1 Router-LSA:

Field	Description
<b>V bit</b>	Indicates whether the advertising router is an endpoint of a virtual link.
<b>E bit</b>	Indicates whether the advertising router is an Autonomous System Border Router (ASBR).
<b>B bit</b>	Indicates whether the advertising router is an Area Border Router (ABR).
<b>Number of Links</b>	Indicates the number of links of the advertising router. A Router-LSA would contain the information of all router links of the advertising router.
<b>Link Type</b>	Indicates the 4 types of router links. <b>Note:</b> There are 2 types of point-to-point links – numbered and unnumbered.
<b>Link ID and Link Data</b>	The Link ID and Link Data values vary according to the Link Type. The Link ID identifies the object to which a link connects to; while the Link Data provides extra information for a link. <b>Note:</b> The LSA header contains the Link-State ID while an LSA entry contains the Link ID allowing for easier differentiation.
<b>Metric</b>	Indicates the OSPF cost of a router link.
<b>ToS, ToS Metric</b>	Represent the Type of Service and are normally set to 0.

- The Link ID, Link Data, Link Type, Number of TOS, Metric, TOS, and TOS metric fields appear one or more times in a Router-LSA corresponding to the Number of Links field.  
**Note:** Cisco supports only TOS = 0. The ToS and ToS Metric fields appear corresponding to the Number of TOS field. Ex: If Number of TOS is 2, there will be 2 32-bit words containing 2 instances of these fields. If Number of TOS is 0, there will be no instances of these fields.

- Below lists the Link ID and Link Data values for the various Type-1 Router-LSA link types:

Type	Description	Link ID	Link Data
1	Point-to-point numbered connection to another router	The Router ID of neighboring router	Physical interface IP address of the router to the network
1	Point-to-point unnumbered connection to another router	The Router ID of neighboring router	The MIB-II ifIndex value of the unnumbered interface <sup>[1]</sup>
2	Connection to a transit network	Physical interface IP address of the DR	Physical interface IP address of the router to the network
3	Connection to a stub network	IP network number or subnet number	Subnet mask
4	Virtual link	The Router ID of virtual link neighbor	Physical interface IP address of the router to reach the virtual link neighbor

[1] – This value usually starts with 0.

- When the connected object is another router, the Link ID is same as the Link-State ID in the Router-LSA header of the neighboring router. The Link ID is being used to find the Router-LSA of the neighboring router during the route computation process.
- A transit interface is a broadcast or NBMA interface with at least one OSPF neighbor. A transit network is described by a Type-2 Network-LSA. A stub interface could be a loopback, broadcast, point-to-point, or multipoint interface on which there is no OSPF neighbor. An OSPF router advertises a multi-access interface as a stub link as long as it has no OSPF neighbors reachable through the interface. When the adjacency to the first OSPF neighbor reaches the FULL state, the OSPF router replaces the stub link as a transit link and advertises its Router-LSA with an incremented sequence number.
- An OSPF router generates a single Type-1 Router-LSA for each area that it belongs to.  
**Note:** A Router-LSA OSPF packet is fragmented when there are more than 119 LSA entries in a single unauthenticated OSPF packet over an Ethernet link with 1500-byte MTU.

20-byte	IP header
24-byte	OSPF header
4-byte	Number of LSAs field
20-byte	Router-LSA header
4-byte	Flags and Number of Links field
1428 bytes	119 LSA entries; 12-byte per entry. 119 x 12-byte = 1428 bytes
<b>Total = 1500 bytes</b>	

- Below shows the interfaces on RT1, RT2, and RT3:

```

RT1#sh ip ospf interface brief
Interface      PID   Area          IP Address/Mask   Cost   State Nbrs F/C
Lo0            100   0             192.168.1.1/32   1      LOOP  0/0
Fa0/0         100   0             10.10.10.1/24    1      BDR   1/1
RT1#
-----
RT2#sh ip ospf interface brief
Interface      PID   Area          IP Address/Mask   Cost   State Nbrs F/C
Se1/0         100   0             20.20.20.1/30    64     P2P   1/1
Fa0/0         100   0             10.10.10.2/24    1      DR    1/1
RT2#
-----
RT3#sh ip ospf interface brief
Interface      PID   Area          IP Address/Mask   Cost   State Nbrs F/C
Fa1/0         100   0             192.168.2.1/24   1      DR    0/0
Se0/0         100   0             20.20.20.2/30    64     P2P   1/1
RT3#

```

- Below shows the Type-1 Router-LSA originated from RT2:

```

RT1#sh ip ospf database router 22.22.22.22

          OSPF Router with ID (11.11.11.11) (Process ID 100)

          Router Link States (Area 0)

LS age: 78
Options: (No TOS-capability, DC)
LS Type: Router Links
Link State ID: 22.22.22.22
Advertising Router: 22.22.22.22
LS Seq Number: 80000002
Checksum: 0x8A1D
Length: 60
Number of Links: 3

  Link connected to: another Router (point-to-point)
    (Link ID) Neighboring Router ID: 33.33.33.33
    (Link Data) Router Interface address: 20.20.20.1
    Number of TOS metrics: 0
      TOS 0 Metrics: 64

  Link connected to: a Stub Network
    (Link ID) Network/subnet number: 20.20.20.0
    (Link Data) Network Mask: 255.255.255.252
    Number of TOS metrics: 0
      TOS 0 Metrics: 64

  Link connected to: a Transit Network
    (Link ID) Designated Router address: 10.10.10.2
    (Link Data) Router Interface address: 10.10.10.2
    Number of TOS metrics: 0
      TOS 0 Metrics: 1

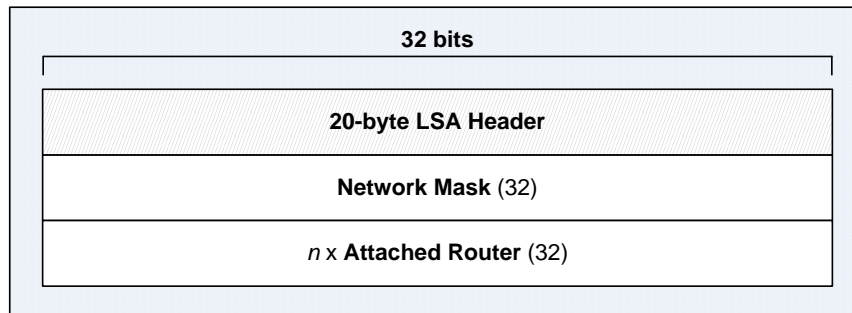
RT1#

```



## Type-2 Network-LSA

- Type-2 Network-LSAs are originated from the DR on a broadcast or multi-access network (eg: Ethernet) to list all the routers that attached to the **transit network**. Type-2 LSAs are flooded within the originating area only. The Link-State ID of a Type-2 Network-LSA is the physical interface IP address of the DR that connects to the network and advertises the LSA.



**Figure 9-5:** OSPF Type-2 Network-LSA Format

- Below lists the fields in the OSPF Type-2 Network-LSA:

Field	Description
<b>Network Mask</b>	Indicates the subnet mask on the transit network.
<b>Attached Router</b>	Lists the Router IDs of all routers on the multi-access network that have established FULL adjacency with the DR and the DR itself.

- Below shows the Type-2 Network-LSA originated from RT2:

```

RT1#sh ip ospf database network

      OSPF Router with ID (11.11.11.11) (Process ID 100)

      Net Link States (Area 0)

Routing Bit Set on this LSA
LS age: 91
Options: (No TOS-capability, DC)
LS Type: Network Links
Link State ID: 10.10.10.2 (address of Designated Router)
Advertising Router: 22.22.22.22
LS Seq Number: 80000001
Checksum: 0xA29A
Length: 32
Network Mask: /24
    Attached Router: 22.22.22.22
    Attached Router: 11.11.11.11

RT1#
    
```

- **Note:** Type-2 Network-LSAs are not necessary be generated for LAN interfaces. Type-2 LSAs are generated only when necessary, which there is more than one router attached to the LAN.

## Type-3 Network-Summary-LSA

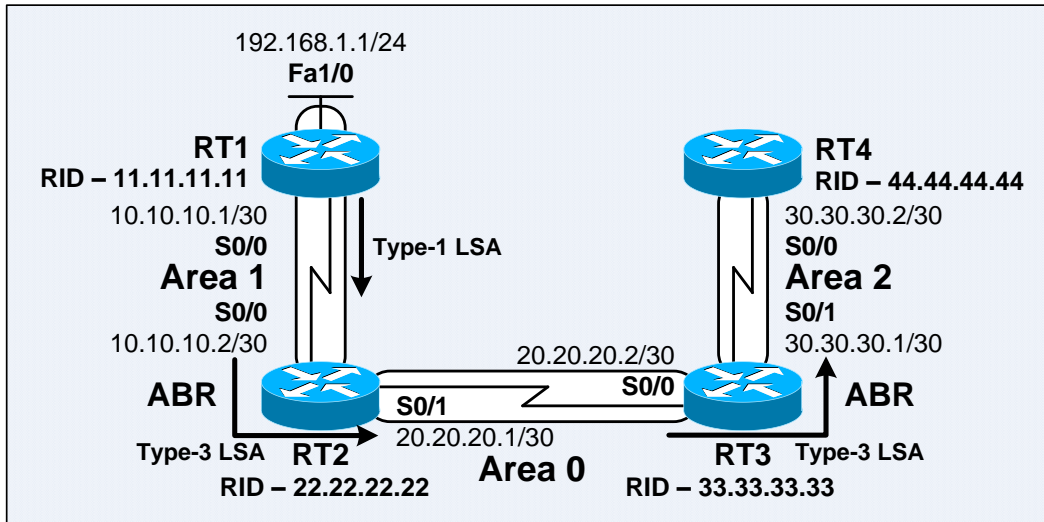


Figure 9-6: Network Setup for Type-3 Network-Summary-LSA

- Type-3 Network-Summary-LSAs are originated from an ABR to advertise the subnets in an area (omitting the information of Type-1 and Type-2 LSAs) to neighboring routers outside the area. Type-1 and Type-2 LSAs stay within an area, when an ABR receives a Type-1 or Type-2 LSA, it generates a Type-3 LSA for the network learnt via the Type-1 or Type-2 LSA to other areas.
- The Type-3 Network-Summary-LSA and Type-4 ASBR-Summary LSA have the same format, with the differences in the contents of the LS Type and Link-State ID fields in the LSA header. For Type-3 LSAs, the Link-State ID is the IP address of the subnet that is being advertised; while for Type-4 LSAs, the Link-State ID is the Router ID of the ASBR that is being advertised.

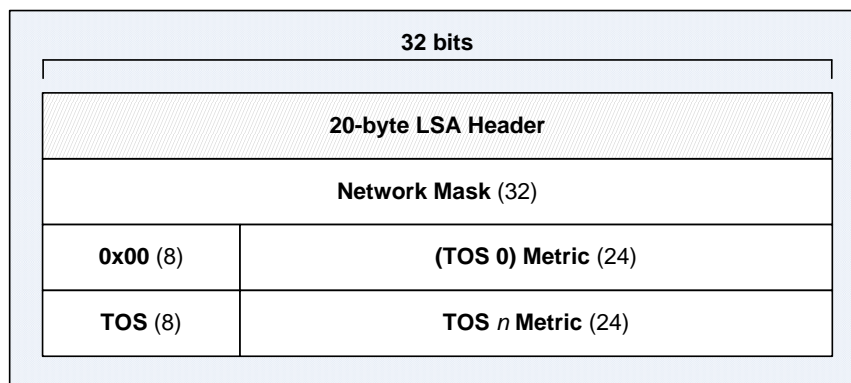


Figure 9-7: OSPF Summary-LSA

(Type-3 Network-Summary-LSA and Type-4 ASBR-Summary-LSA) Format

- Below lists the fields in the OSPF Summary-LSA:

Field	Description
<b>Network Mask</b>	For Type-3 LSAs, it indicates the subnet mask of the network that is being advertised. For Type-4 LSAs, this field has no meaning and is set to 0.0.0.0. <b>Note:</b> If a Type-3 LSA is advertising a default route, both the Link-State ID and Network Mask fields will be 0.0.0.0.
<b>Metric</b>	Indicates the cost of the route to the destination. <b>Note:</b> Cisco supports only TOS 0 therefore there are no TOS and TOS Metric fields in the Cisco Summary-LSAs.

- Type-3 Network-Summary-LSAs are not being summarized and therefore do not contain summary routes by default. All subnets in an area will be advertised by default.
- OSPF does not perform autosummarization as like RIPv2 and EIGRP. Route summarization in OSPF requires manual configuration. Route summarization and filtering at the ABRs should always be considered due to the nature that a Type-3 LSA will be advertised into the backbone area for every network in the originating area, which can cause significant flooding problems. Route summarization and filtering on ABRs provide greater scalability for OSPF.
- **Note:** Type-3 and Type-4 Summary-LSAs do not carry topological information; they carry only IP prefixes or subnets. Summary-LSAs are originated by an ABR with the following rules:
  - From a non-backbone area to the backbone area, Summary-LSAs are generated for **connected routes** and **intra-area routes**. Only intra-area routes are advertised into the backbone area to avoid routing loops. If there are any inter-area routes coming from a non-backbone area it means that the backbone is discontinuous, and OSPF does not allow discontinuous backbone.
  - From the backbone area to a non-backbone area, Summary-LSAs are generated for **connected routes**, **intra-area routes**, and **inter-area routes**.
- The Advertising Route field in the LSA header of a Summary-LSA contains the Router ID of the ABR that generates the Summary-LSA.
- Below shows the routing table and Type-3 Network-Summary-LSAs received on RT4:

```

RT4#sh ip route

Gateway of last resort is not set

    20.0.0.0/30 is subnetted, 1 subnets
O IA   20.20.20.0 [110/128] via 30.30.30.1, 00:00:21, Serial0/0
    10.0.0.0/30 is subnetted, 1 subnets
O IA   10.10.10.0 [110/192] via 30.30.30.1, 00:00:18, Serial0/0
O IA   192.168.1.0/24 [110/193] via 30.30.30.1, 00:00:18, Serial0/0
    30.0.0.0/30 is subnetted, 1 subnets
C      30.30.30.0 is directly connected, Serial0/0
RT4#
RT4#sh ip ospf database

                OSPF Router with ID (44.44.44.44) (Process ID 100)

                Router Link States (Area 2)

Link ID        ADV Router    Age           Seq#           Checksum Link count
33.33.33.33    33.33.33.33   34           0x80000001    0x005CDA  2
44.44.44.44    44.44.44.44   31           0x80000002    0x00A663  2

                Summary Net Link States (Area 2)

Link ID        ADV Router    Age           Seq#           Checksum
10.10.10.0     33.33.33.33   19           0x80000001    0x0031EB
20.20.20.0     33.33.33.33   29           0x80000001    0x0045F9
192.168.1.0    33.33.33.33   19           0x80000001    0x00F9D2
RT4#

```

RT4#sh ip ospf database summary

OSPF Router with ID (44.44.44.44) (Process ID 100)

Summary Net Link States (Area 2)

Routing Bit Set on this LSA  
LS age: 19  
Options: (No TOS-capability, DC, Upward)  
LS Type: Summary Links(Network)  
**Link State ID: 10.10.10.0** (summary Network Number)  
**Advertising Router: 33.33.33.33**  
LS Seq Number: 80000001  
Checksum: 0x31EB  
Length: 28  
Network Mask: /30  
TOS: 0 Metric: 128

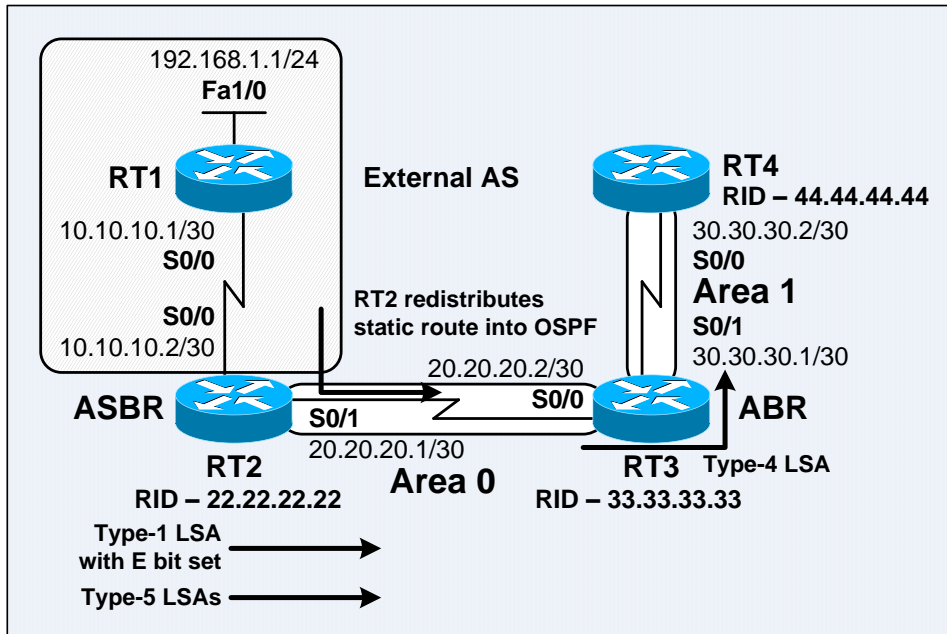
Routing Bit Set on this LSA  
LS age: 29  
Options: (No TOS-capability, DC, Upward)  
LS Type: Summary Links(Network)  
**Link State ID: 20.20.20.0** (summary Network Number)  
**Advertising Router: 33.33.33.33**  
LS Seq Number: 80000001  
Checksum: 0x45F9  
Length: 28  
Network Mask: /30  
TOS: 0 Metric: 64

Routing Bit Set on this LSA  
LS age: 19  
Options: (No TOS-capability, DC, Upward)  
LS Type: Summary Links(Network)  
**Link State ID: 192.168.1.0** (summary Network Number)  
**Advertising Router: 33.33.33.33**  
LS Seq Number: 80000001  
Checksum: 0xF9D2  
Length: 28  
Network Mask: /24  
TOS: 0 Metric: 129

RT4#

- OSPF routers must perform SPF calculations upon the intra-area topology changes (Type-1 and Type-2 LSAs), as these changes impact the selection of best paths towards destination within an area. Changes upon Type-3 LSAs do not trigger an SPF calculation, as the changes do not affect the topology between an OSPF router and the ABR.

## Type-4 ASBR-Summary-LSA and Type-5 AS-External-LSA

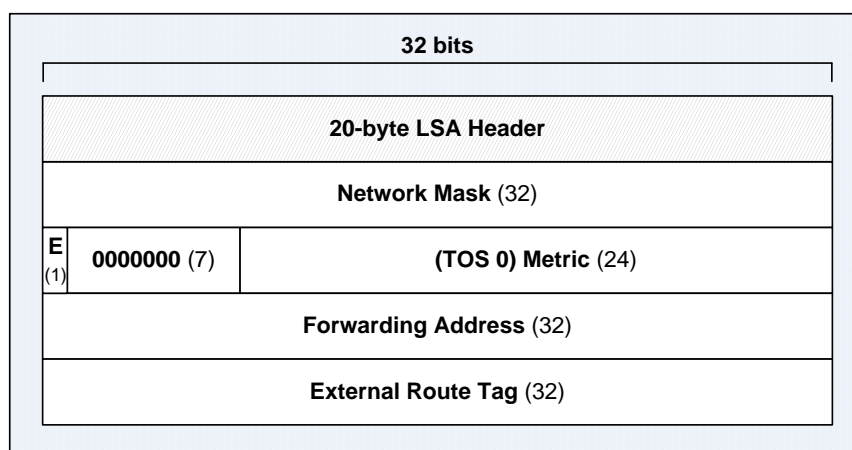


**Figure 9-8:** Network Setup for Type-4 ASBR-Summary-LSA and Type-5 AS-External-LSA

- Type-4 ASBR-Summary-LSAs are originated from an ABR that resides within the same area as the ASBR to identify the ASBR and describes the route to the ASBR. Type-5 LSAs are flooded to all areas but the detailed info to reach the ASBR may not be available throughout all areas. This problem is solved by having an ABR to describe the route to the ASBR that originated the Type-5 LSA. The Link-State ID of a Type-4 LSA is the Router ID of the ASBR.
- Type-5 AS-External-LSAs are originated from an ASBR and contains information imported into an OSPF routing domain from other routing processes. Type-5 LSAs are flooded throughout all areas except stub areas, totally stubby areas, and NSSAs. When a stub router receives a Type-5 LSA from another router within the stub area, the LSA will be rejected. The Link-State ID of a Type-5 LSA is the IP address of the external subnet that is being advertised.

### MISC Note:

- Internal routes are always preferred over external routes.
- There is no Type-4 and Type-5 LSAs when there is no ASBR in an OSPF routing domain.
- All OSPF routers in a routing domain maintain the same Type-5 LSA in their LSDBs.
- All types of LSAs have *area flooding scope* (not flooded across area borders); expect Type-5 LSAs which have *AS flooding scope* (flooded throughout all areas).



**Figure 9-9:** OSPF Type-5 AS-External-LSA Format

- Below lists the fields in the OSPF AS-External-LSA:

Field	Description
<b>Network Mask</b>	Indicates the subnet mask of the external network that is being advertised. <b>Note:</b> If the Type-5 LSA is advertising a default route, the Link-State ID and the Network Mask are both 0.0.0.0.
<b>E (External Metric) bit</b>	Indicates the type of external metric for the advertised external route. If the E bit is set to 1, the metric type is E2; else the metric type is E1. E2 is the default route type for routes learnt via route redistribution.
<b>Metric</b>	Indicates the cost of the external route as set by the ASBR.
<b>Forwarding Address</b>	Indicates the destination IP address to which the packets for the advertised external network should be forwarded. If the Forwarding Address is 0.0.0.0, the packets are forwarded to the originating ASBR. The Forwarding Address will be non-zero in the following situations to avoid suboptimal routing: <ul style="list-style-type: none"> <li>- OSPF is enabled on the ASBR's next-hop interface</li> <li>- The ASBR's next-hop interface is non-passive to OSPF</li> <li>- The ASBR's next-hop interface network type is not point-to-point or point-to-multipoint</li> <li>- The ASBR's next-hop interface address falls into the OSPF network range</li> </ul> <b>Note:</b> The route to the non-zero Forwarding Address must be known via an intra-area or inter-area route; or else the external route will not be installed into the routing table!
<b>External Route Tag</b>	Indicates an arbitrary tag that may be applied upon the external route. It is not being used by OSPF itself, but is instead provides external route management using route maps.

- Below lists the criteria for an OSPF router to install an AS-External-LSA into its routing table:
  - The router must be able to reach the ASBR through an intra-area or inter-area route, which means that there must be a Type-1 Router-LSA or Type-4 ASBR-Summary-LSA generated for the ASBR.
  - The Forwarding Address must be known via an intra-area or inter-area route.
- A default seed metric of 1 will be used for redistributed BGP routes; while a default seed metric of 20 will be used for routes redistributed from other routing protocols, including static routes and directly connected interfaces. The **seed metric** is a starting metric for redistributed routes.
- Below shows the routing table, as well as Type-4 ASBR-Summary-LSA and Type-5 AS-External-LSAs received on RT4:

```

RT4#sh ip route

Gateway of last resort is not set

    20.0.0.0/30 is subnetted, 1 subnets
O IA   20.20.20.0 [110/128] via 30.30.30.1, 00:00:45, Serial0/0
    10.0.0.0/30 is subnetted, 1 subnets
O E2   10.10.10.0 [110/20] via 30.30.30.1, 00:00:35, Serial0/0
O E2   192.168.1.0/24 [110/20] via 30.30.30.1, 00:00:35, Serial0/0
    30.0.0.0/30 is subnetted, 1 subnets
C       30.30.30.0 is directly connected, Serial0/0
RT4#

```

RT4#**sh ip ospf database asbr-summary**

OSPF Router with ID (44.44.44.44) (Process ID 100)

Summary ASB Link States (Area 1)

Routing Bit Set on this LSA

LS age: 45

Options: (No TOS-capability, DC, Upward)

LS Type: Summary Links(AS Boundary Router)

**Link State ID: 22.22.22.22 (AS Boundary Router address)**

**Advertising Router: 33.33.33.33**

LS Seq Number: 80000001

Checksum: 0x24FA

Length: 28

Network Mask: /0

TOS: 0 Metric: 64

RT4#

RT4#**sh ip ospf database external**

OSPF Router with ID (44.44.44.44) (Process ID 100)

Type-5 AS External Link States

Routing Bit Set on this LSA

LS age: 57

Options: (No TOS-capability, DC)

LS Type: AS External Link

**Link State ID: 10.10.10.0 (External Network Number )**

**Advertising Router: 22.22.22.22**

LS Seq Number: 80000001

Checksum: 0xD557

Length: 36

Network Mask: /30

Metric Type: 2 (Larger than any link state path)

TOS: 0

Metric: 20

Forward Address: 0.0.0.0

External Route Tag: 0

Routing Bit Set on this LSA

LS age: 57

Options: (No TOS-capability, DC)

LS Type: AS External Link

**Link State ID: 192.168.1.0 (External Network Number )**

**Advertising Router: 22.22.22.22**

LS Seq Number: 80000001

Checksum: 0x9449

Length: 36

Network Mask: /24

Metric Type: 2 (Larger than any link state path)

TOS: 0

Metric: 20

Forward Address: 0.0.0.0

External Route Tag: 0

RT4#

**Note:** Type-4 LSAs are not translated directly from link-state database into routing table.

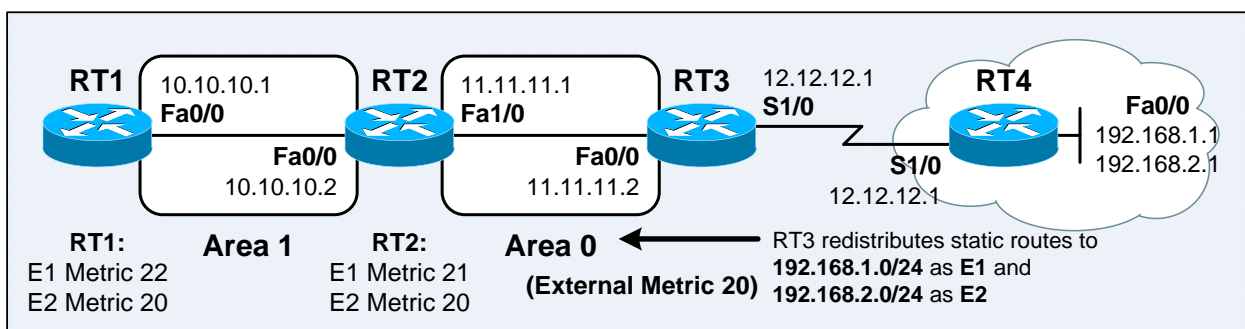
**Note:** The metric value in a Type-4 LSA indicates the cost from the ABR to reach the ASBR.

This value is used by routers within an area to determine the nearest ABR for an external route.

- Below describes the routing table designators for OSPF:

Code	Description
O	<b>OSPF intra-area route</b> that indicates a network within the area which the router resides. Learnt via a Router-LSA or Network-LSA.
O IA	<b>OSPF inter-area route</b> that indicates a network outside the area which the router resides but within the OSPF routing domain. Learnt via a Summary-LSA.
O E1 / O E2	<b>OSPF Type-1 / Type-2 external route</b> that indicates a network outside the autonomous system which the router resides. Learnt via an AS-External-LSA.

- The difference between **External Type-1 (E1)** and **External Type-2 (E2)** external routes is that the metric of an E1 route is similar to the OSPF metric in which the external metric is being accumulated with the internal metric of every link across the forwarding path towards the ASBR; while the metric of an E2 route remains the same as the external metric throughout an OSPF routing domain. E2 is the default route type for routes that learnt via route redistribution.
- The recommended best practices are implement E1 routes when multiple ASBRs are advertising an external route into the same routing domain in order to avoid suboptimal routing; implement E2 routes when there is only 1 ASBR is advertising an external route into the routing domain.



**Figure 9-10: Network Setup for OSPF E1 and E2 Routes**

- Below shows the routing tables on RT1 and RT2 for the OSPF E1 and E2 routes advertised by RT3:

```

RT1#sh ip route

Gateway of last resort is not set

    10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0
    11.0.0.0/24 is subnetted, 1 subnets
O IA    11.11.11.0 [110/2] via 10.10.10.2, 00:00:39, FastEthernet0/0
O E1   192.168.1.0/24 [110/22] via 10.10.10.2, 00:00:24, FastEthernet0/0
O E2   192.168.2.0/24 [110/20] via 10.10.10.2, 00:00:24, FastEthernet0/0
RT1#

-----

RT2#sh ip route

Gateway of last resort is not set

    10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0
    11.0.0.0/24 is subnetted, 1 subnets
C       11.11.11.0 is directly connected, FastEthernet1/0
O E1   192.168.1.0/24 [110/21] via 11.11.11.2, 00:00:37, FastEthernet1/0
O E2   192.168.2.0/24 [110/20] via 11.11.11.2, 00:00:37, FastEthernet1/0
RT2#

```



- External Type-1 (E1) routes are always preferred over External Type-2 (E2) routes. When there are multiple External Type-2 (E2) routes for the same external network in the OSPF LSDB, the LSA with the lowest advertised E2 metric will be selected and installed into the routing table. The **forward metric** (cost to reach an ASBR) will be considered when multiple E2 routes have the same metric value. The **show ip ospf border-routers** command displays the cost to an ASBR. **Note:** The internal metric or cost towards the ASBR is not considered when comparing E2 routes.

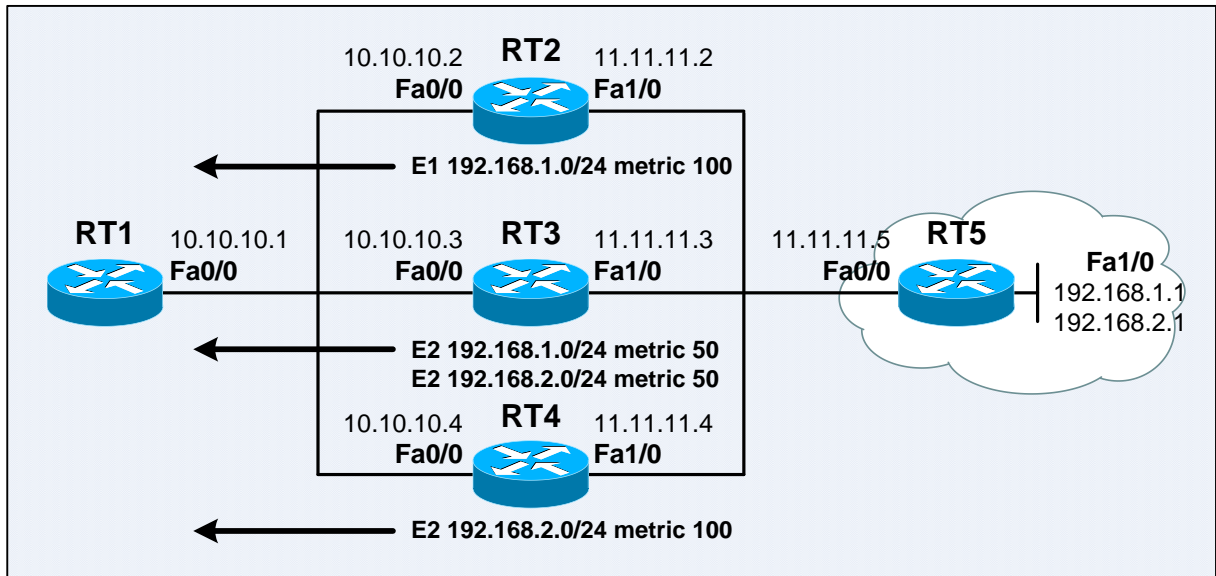


Figure 9-11: Network Setup for OSPF E1 and E2 Routes Selection

- Below shows the routing table and OSPF LSDB on RT1 for the OSPF external routes:

```

RT1#sh ip route

Gateway of last resort is not set

    10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0
O E1 192.168.1.0/24 [110/101] via 10.10.10.2, 00:00:30, FastEthernet0/0
O E2 192.168.2.0/24 [110/50] via 10.10.10.3, 00:00:30, FastEthernet0/0
RT1#sh ip ospf database

        OSPF Router with ID (10.10.10.1) (Process ID 100)

--- output omitted ---

        Net Link States (Area 0)

Link ID        ADV Router    Age           Seq#           Checksum
10.10.10.4     11.11.11.4   59            0x80000001    0x00B4AA

        Type-5 AS External Link States

Link ID        ADV Router    Age           Seq#           Checksum      Tag
192.168.1.0  11.11.11.2   96            0x80000001    0x00B58D    0
192.168.1.0  11.11.11.3   96            0x80000001    0x003DB6    0
192.168.2.0  11.11.11.3   96            0x80000001    0x0032C0    0
192.168.2.0  11.11.11.4   100           0x80000001    0x00229D    0
RT1#

```

## OSPF Route Summarization and Filtering

- Route summarization consolidates multiple routes into a single summary route. Proper route summarization planning and implementation directly affects the amount of bandwidth, CPU, and memory resources consumed by routers.
- Without route summarization, every inter-area LSA (Type-3, Type-4, and Type-5) is propagated by ABRs throughout an OSPF domain, consuming additional bandwidth and router resources. Upon the flooding of an inter-area LSA, all affected OSPF routers must process the LSA and perform route computation to locate the lowest metric path to the ABR for the destination network (link up) or flush an MaxAge LSA (link down).
- With route summarization, only the summary routes are being propagated by ABRs. Route summarization reduces unnecessary LSA flooding, reduces route re-computation upon topology changes, and increases network stability.
- Below describes the 2 types of route summarization:

<b>Inter-area route summarization</b>	Occurs on ABRs and applies upon routes propagated between areas. Does not apply to external routes injected into OSPF via redistribution. To achieve effective inter-area route summarization, network numbers within areas should be assigned contiguously for the addresses to be summarized into a minimal number of summary addresses as possible.
<b>External route summarization</b>	Applies upon external routes injected into OSPF via redistribution. It is important to ensure the continuity of external address ranges that are being summarized. Summarizing overlapping address ranges from different ASBRs can cause packets to be forwarded through the wrong path. External addresses should be assigned contiguously as well.

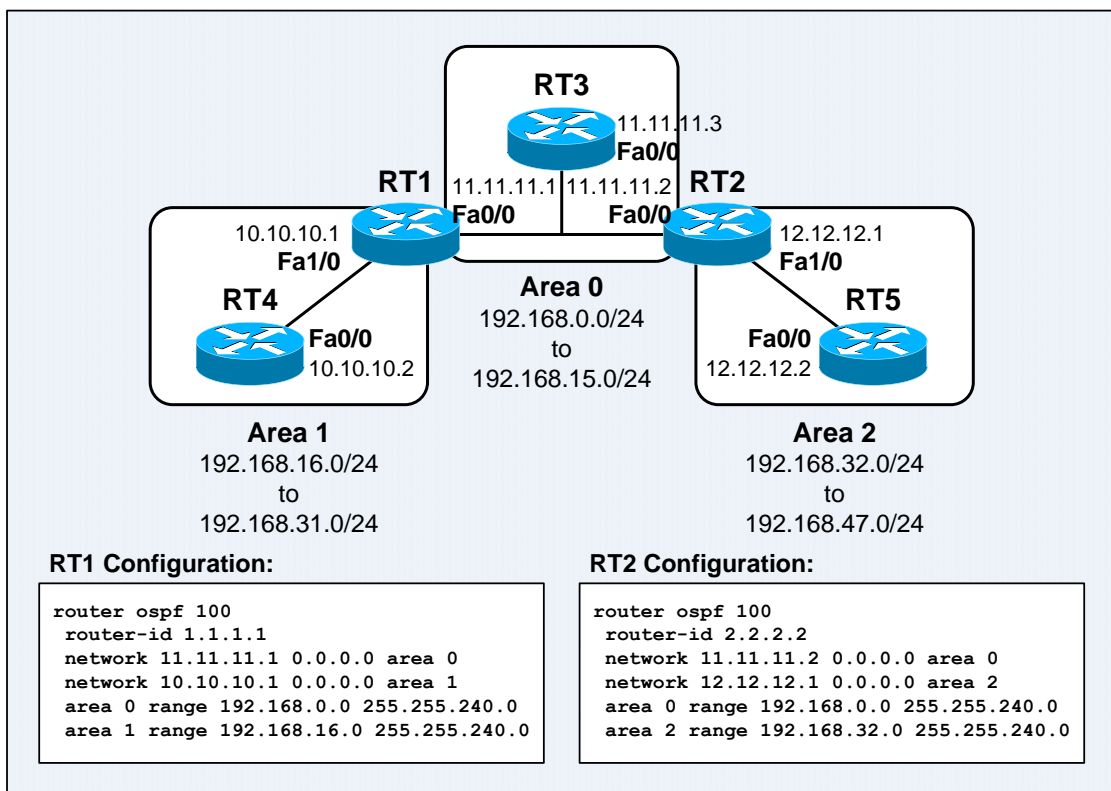
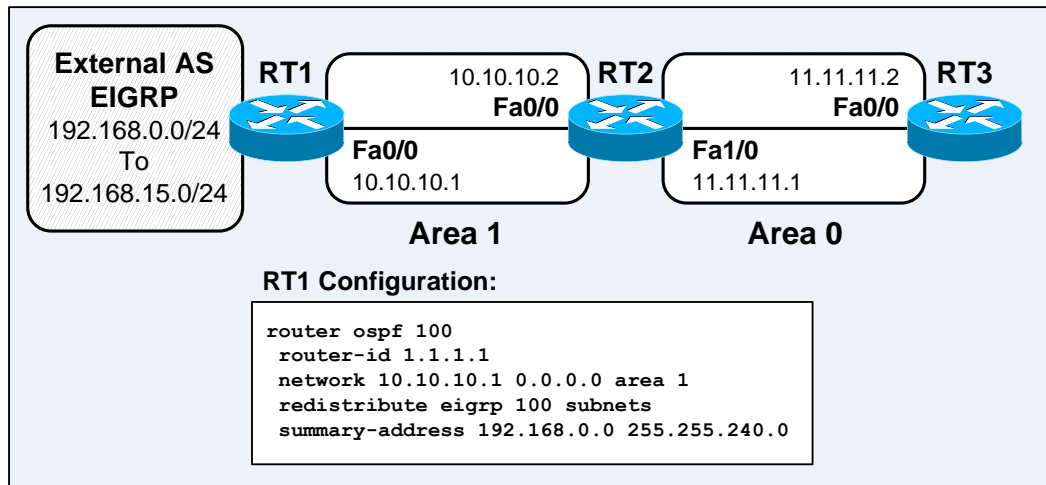


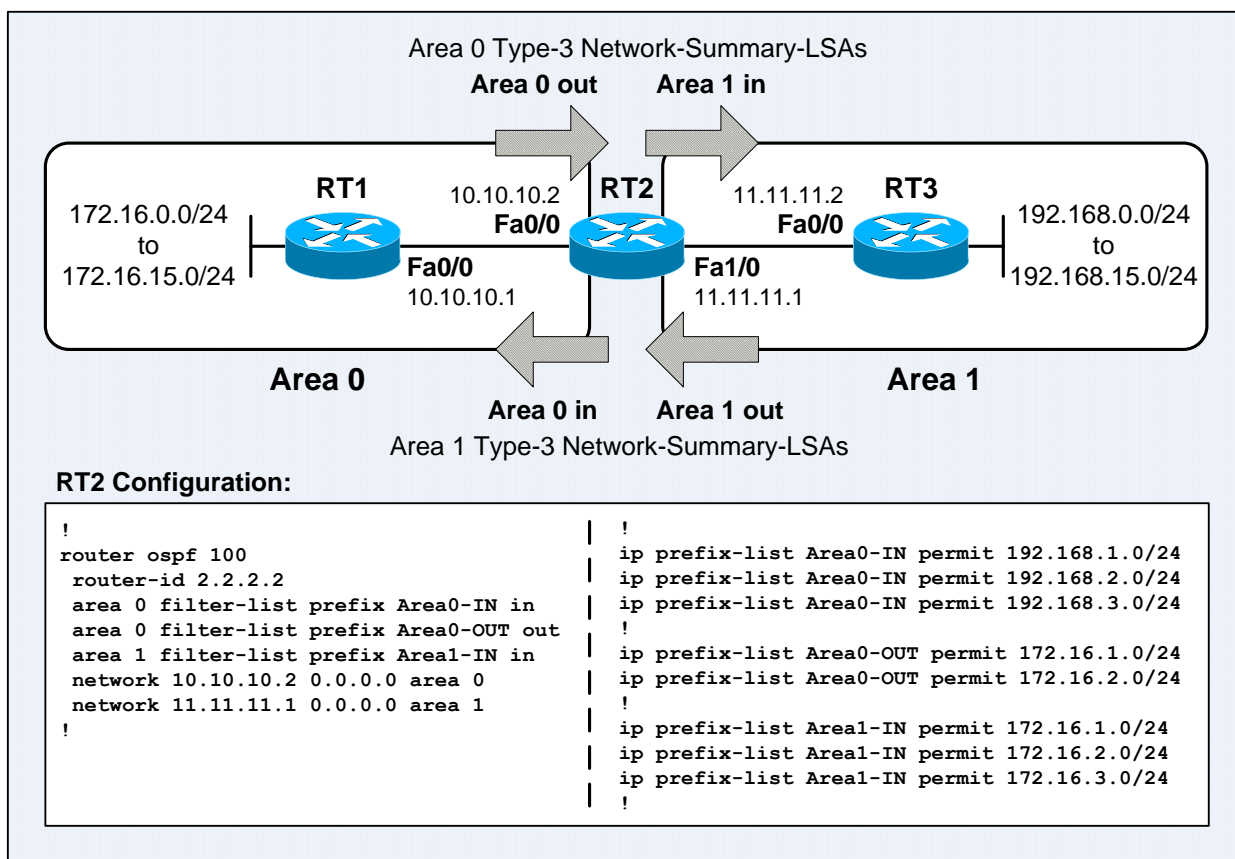
Figure 9-12: OSPF Route Summarization on an ABR

- **Note:** Always reconsider when summarizing networks in the backbone area to other areas. When there is more than 1 ABR between an area and the backbone area, propagating a Type-3 Network-Summary-LSA with the explicit network information into an area ensures that the shortest path to the destination outside the area will be selected and prevents suboptimal routing.



**Figure 9-13: OSPF Route Summarization on an ASBR**

- The **summary-address**  $\{\{address\ mask\} | \{prefix\ mask\}\}$  [**not-advertise**] [**tag tag**] OSPF router subcommand configures a summary route for external routes on an OSPF ASBR. The **not-advertise** keyword suppresses the routes that match the prefix/mask pair.
- Link-state routing protocols do not advertise routes, but advertise topology information instead. OSPF is a link-state routing protocol that relies upon the information in the topology database for route computation. All routers within an area must have the same and consistent topology information to make accurate routing decisions; the routers **independently calculate** the shortest paths to destination networks based on the topology information using the SPF algorithm.
- OSPF route filtering within an area can be achieved using distribute lists (in conjunction with access lists, prefix lists, and route maps), or modifying the administrative distance.
  - Note:** OSPF routers do not advertise routes; instead, they advertise LSAs. Any filtering applied upon OSPF LSU packets would need to filter the transmission of LSAs. Performing OSPF route filtering within an area does not affect routes as they enter the OSPF topology database, but the IP routing table instead; and on only the router on which the route filtering is configured. Additionally, it does not affect the propagation of LSAs from the router that performs route filtering – route filtering is local significant.
  - Note:** Route filtering on link-state routing protocols often result in different routing tables (but same topology database) on routers, which would then introduce routing loops or routing black holes to the network.
- The **OSPF ABR Type-3 LSA Filtering** feature extends the ability of OSPF ABRs to filter routes that are being propagated between OSPF areas. It allows only routes matched with specified prefixes to be sent from one area to another area and restricts all other routes. It can be applied out from an area, into an area, or both into and out of an area at the same time.
- The main benefit of the OSPF ABR Type-3 LSA Filtering feature is provides improved control of route distribution between OSPF areas by filtering Type-3 LSAs originated from ABRs.



**Figure 9-14: OSPF ABR Type-3 LSA Filtering**

- Below shows the routing table on RT1 and RT3 with route filtering configured on RT2:

```

RT2#sh ip ospf
--- output omitted ---
Area BACKBONE (0)
--- output omitted ---
Area-filter Area0-IN in
Area-filter Area0-OUT out
--- output omitted ---
Area 1
--- output omitted ---
Area-filter Area1-IN in
--- output omitted ---

RT2#
=====
RT1#sh ip route

--- output omitted ---
O IA 192.168.1.0/24 [110/3] via 10.10.10.2, 00:00:32, FastEthernet0/0
O IA 192.168.2.0/24 [110/3] via 10.10.10.2, 00:00:32, FastEthernet0/0
O IA 192.168.3.0/24 [110/3] via 10.10.10.2, 00:00:32, FastEthernet0/0
RT1#
=====

RT3#sh ip route

--- output omitted ---
O IA 172.16.1.0 [110/3] via 11.11.11.1, 00:00:40, FastEthernet0/0
O IA 172.16.2.0 [110/3] via 11.11.11.1, 00:00:40, FastEthernet0/0
RT3#

```

- **Note:** RT3 does not see 172.16.3.0/24 in its routing table. This is because the route was not advertised from Area 0 (Area0-OUT) even it is allowed to be advertised into Area 1 (Area1-IN).
- The **area area-id filter-list prefix prefix-name {in | out}** OSPF router subcommand filters IP prefixes or subnets as advertised as Type-3 Network-Summary-LSAs between OSPF areas. The *area-id* is the identifier of the area for which route filtering is configured; it can be specified as either a decimal value or an IP address. The **prefix** keyword indicates that a prefix list is used. Prefixes that do not match any entry in the prefix list are implicitly denied.
- The **in** direction applies upon the Type-3 LSAs originated by the ABR based on the information from other areas into the specified area; while the **out** direction applies upon the Type-3 LSAs originated by the ABR based on the information from the specified area to other areas. Eventually, the **area filter-list** command filters Type-3 LSAs from going in or out of an area. Filtering Type-3 LSAs between areas prevents the routes from entering the OSPF topology database and eventually prevents them from entering the routing table.
- Type-3 LSAs that were originated as a result of the **area range** OSPF router subcommand are treated as ordinary Type-3 LSAs. When route summarization (**area range**) and route filtering (**area filter-list**) are implemented together on an OSPF router to propagate a summary route into an area (**in** direction), the summary route must be matched with an entry in the prefix list for it to be propagated into the area. While propagating a summarized route from an area (**out** direction), at least one prefix in the summarized area range must be matched with an entry in the prefix list for it to be propagated to other areas – if no subordinate routes exist, the ABR does not advertise the summary.
- When the **area range** OSPF router subcommand is configured with the **not-advertise** keyword, both the summary route and component routes will not be advertised as Type-3 LSAs. As a result, it has the same effect as the **area filter-list** OSPF router subcommand with the **out** keyword, which filters the LSAs from originating from a specific area to other areas.
- Component routes are also being referred to as **subordinate routes** – the routes whose address ranges are inside the range of addresses defined by the summary route.

## Advanced OSPF – OSPF Stub Areas and OSPF Virtual Links

### OSPF Stub Areas

- The characteristics of an OSPF area determine the types of routing information it will receive. The main purpose behind all types of stub areas is to inject a default route into an area so that the external (and summary) LSAs are not flooded into the area. Replacing multiple external routes with a single default route reduces the size of LSDBs and routing tables in the OSPF routers within the stub area, reduces the memory requirements for OSPF routers within the stub area, and the OSPF routers run fewer SPF calculations as they will receive lesser routing updates.
- Below describes the possible OSPF area types and their characteristics:

<b>Standard area</b>	Accepts Type-1 and Type-2 link info updates, Type-3 and Type-4 summary routes, and Type-5 external routes.
<b>Backbone area or transit area</b>	The central entity to which all other areas connect. All other areas connect to this area to exchange routing information. The OSPF backbone area has all the properties of an OSPF standard area.
<b>Stubby area</b>	Doesn't accept external routes (Type-4 and Type-5 LSAs) from other autonomous systems (via route redistribution). The OSPF routers within a stubby area use the default route as indicated as 0.0.0.0 to route packets to networks outside the autonomous system. Stubby areas cannot contain ASBRs; however, the ABR may be an ASBR.
<b>Totally Stubby area</b>	A Cisco-proprietary feature that further minimizes routing information than stubby area by filtering Type-4 and Type-5 LSAs, and Type-3 Net-Summary-LSAs from other areas within the autonomous system.
<b>Not-So-Stubby-Area (NSSA)</b>	NSSA is defined in RFC 3101 as an addendum to the OSPF RFCs. The OSPF NSSA offers the benefits that are similar to the stubby area. However, <b>NSSA allows ASBRs</b> (by introducing the Type-7 NSSA-External-LSA), which against the rules of OSPF stub area.
<b>Not-So-Stubby-Totally-Stubby Area</b>	Combines the characteristics of totally stubby area and NSSA – does not accept Type-3, Type-4, and Type-5 LSAs; allows ASBRs.

- An OSPF area is qualified as a stubby or totally stubby area if it has the following characteristics:
  - i) There is a single exit point (ABR) from the area; or if there are multiple exit points, multiple ABRs inject default route into the stub area and suboptimal routing is acceptable – routing to other areas or autonomous systems can take a suboptimal path to reach the destination network by exiting the stub area via an exit point that is farther from the destination than other exit points.
  - ii) The area is not needed as a transit area for virtual links.
  - iii) The area is not the backbone area.
  - iv) No ASBR is inside the stubby or totally stubby area.
  - v) All OSPF routers (internal routers and ABRs) within the stub area must be configured as stub routers for them to become neighbors and exchange routing information. The Hello packets exchanged between the routers will have the stub flag set – the E bit (ExternalRoutingCapability) of the Options field is set to 0.
- **Note:** The ABR of a stub area generates a default route into the stub area regardless of whether it has a default route or received any external route from other ASBR resides in the standard area. A stub router selects the closest ABR as the gateway to reach networks outside the stub area.

## OSPF Stubby Area

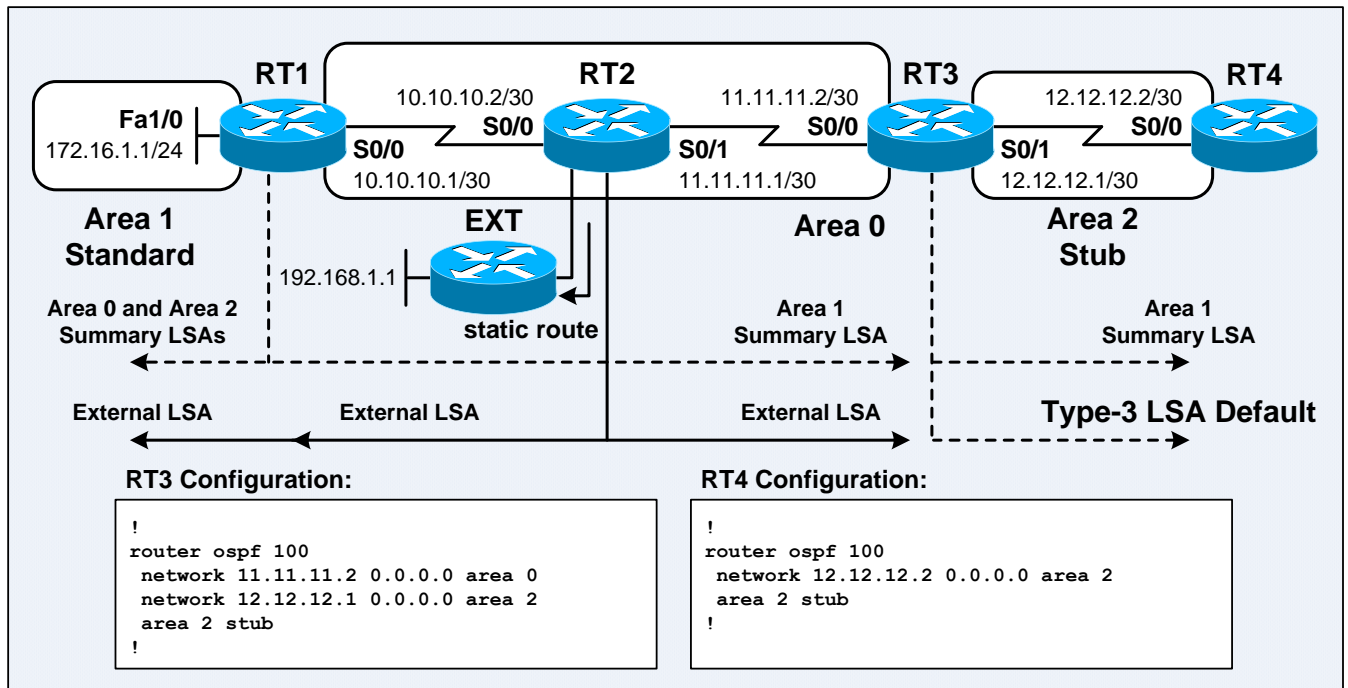


Figure 10-1: OSPF Stubby Area

- Below shows the routing table and OSPF LSDB on RT4:

```
RT4#sh ip route
Gateway of last resort is 12.12.12.1 to network 0.0.0.0

 172.16.0.0/24 is subnetted, 1 subnets
O IA   172.16.1.0 [110/193] via 12.12.12.1, 00:00:24, Serial0/0
 10.0.0.0/30 is subnetted, 1 subnets
O IA   10.10.10.0 [110/192] via 12.12.12.1, 00:00:24, Serial0/0
 11.0.0.0/30 is subnetted, 1 subnets
O IA   11.11.11.0 [110/128] via 12.12.12.1, 00:00:30, Serial0/0
 12.0.0.0/30 is subnetted, 1 subnets
C      12.12.12.0 is directly connected, Serial0/0
O*IA  0.0.0.0/0 [110/65] via 12.12.12.1, 00:00:30, Serial0/0
RT4#
RT4#sh ip ospf database

        OSPF Router with ID (12.12.12.2) (Process ID 100)

--- output omitted ---

                Summary Net Link States (Area 2)

Link ID        ADV Router    Age         Seq#         Checksum
0.0.0.0        3.3.3.3      40         0x80000001  0x0057DA
10.10.10.0     3.3.3.3      26         0x80000001  0x00D6C0
11.11.11.0     3.3.3.3      36         0x80000001  0x0030A4
172.16.1.0     3.3.3.3      26         0x80000001  0x00CB28
RT4#
```

- The ABR of a stubby or totally stubby area advertises a default route with a cost of 1 by default. The **area {area-id} default-cost {cost}** router subcommand changes the cost of the default route.

## OSPF Totally Stubby Area

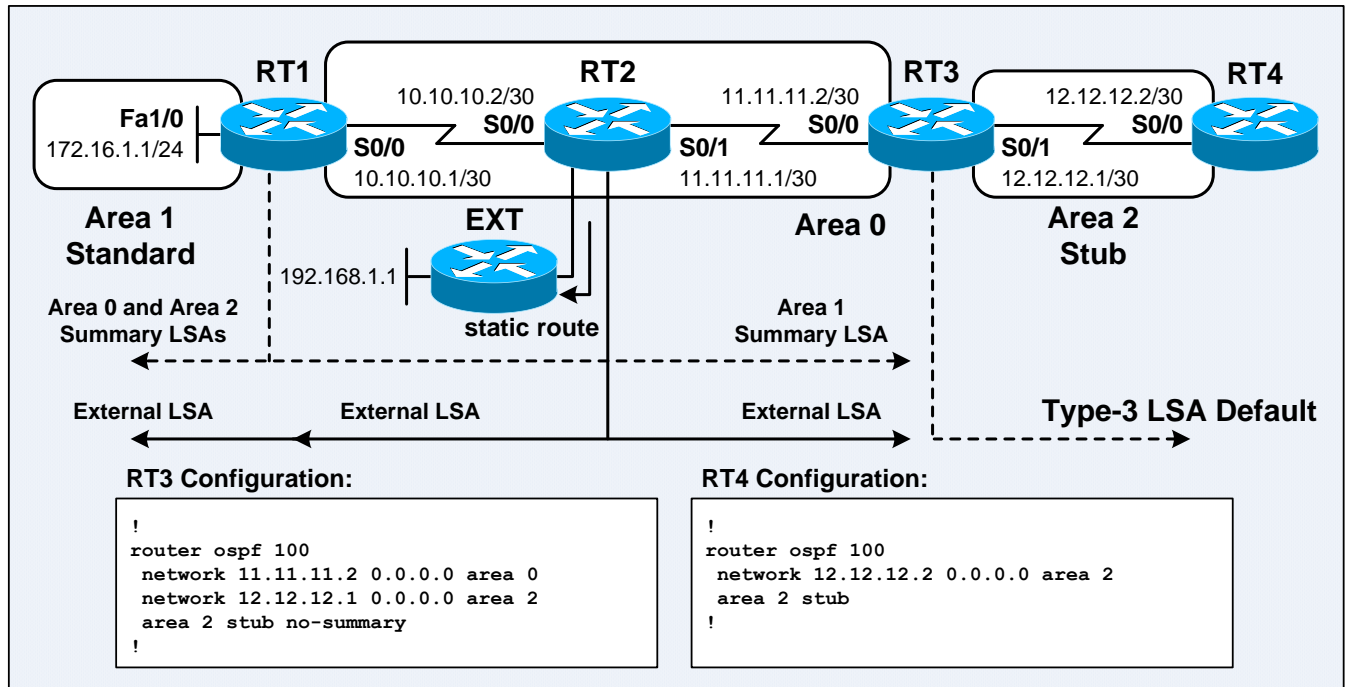


Figure 10-2: OSPF Totally Stubby Area

- Below shows the routing table and OSPF LSDB on RT4, and the excerpt of **show ip ospf** on RT3:

```
RT4#sh ip route
Gateway of last resort is 12.12.12.1 to network 0.0.0.0

 12.0.0.0/30 is subnetted, 1 subnets
 C    12.12.12.0 is directly connected, Serial0/0
 O*IA 0.0.0.0/0 [110/65] via 12.12.12.1, 00:00:32, Serial0/0
RT4#
RT4#sh ip ospf database

          OSPF Router with ID (12.12.12.2) (Process ID 100)

--- output omitted ---

          Summary Net Link States (Area 2)

Link ID        ADV Router    Age           Seq#           Checksum
0.0.0.0        3.3.3.3      42           0x80000001    0x0057DA
RT4#
=====
RT3#sh ip ospf
--- output omitted ---
Area 2
  Number of interfaces in this area is 1
  It is a stub area, no summary LSA in this area
  generates stub default route with cost 1
--- output omitted ---
RT3#
```

- RT4 is configured with the **area {area-id} stub** command without the **no-summary** keyword. The **no-summary** keyword only required to be configured on the ABR of a totally stubby area to stop the ABR from propagating summary LSAs into the totally stubby area.



## OSPF Not-So-Stubby-Area (NSSA)

- OSPF Not-So-Stubby-Area (NSSA) is defined in **RFC 3101 – The OSPF Not-So-Stubby Area (NSSA) Option** as an addendum to the official OSPF Request for Comments (RFC). It is a non-proprietary extension upon the OSPF stub area feature that allows ASBRs in a stub area and therefore allows the injection of external routes into the stub area in a limited fashion. The NSSA retains the main feature of stub areas – the NSSA ABR generates a default route into the NSSA instead of external routes originated from other ASBRs.
- An ASBR within an NSSA originates Type-7 NSSA-External-LSA upon route redistribution. All fields of the NSSA-External-LSA and AS-External-LSA are identical except the Forward Address field. AS-External-LSAs are flooded throughout an OSPF routing domain; while NSSA-External-LSAs are flooded only within the NSSA in which it was originated. The **show ip ospf database nssa-external** EXEC command displays NSSA-External-LSAs.
- When an NSSA ASBR generates a Type-7 NSSA-External-LSA, the **NSSA ABR with the highest Router ID** among the NSSA ABRs (when there are multiple NSSA ABRs) translates the Type-7 LSA to a Type-5 LSA and propagates it throughout the OSPF routing domain.
- Type-7 NSSA-External-LSAs are denoted as O N2 or O N1 in the IP routing table. The metrics of N1 and N2 are calculated as like E1 Type-1 and E2 Type-2 external routes respectively.
- The **area area-id nssa [default-information-originate [metric metric] [metric-type metric-type]] [no-redistribution] [no-summary]** OSPF router subcommand is used instead of the **area area-id stub** OSPF router subcommand to configure an NSSA OSPF router. All OSPF routers (internal routers and ABRs) within an NSSA must be configured as NSSA routers for them to become neighbors and exchange routing information. The Hello packets exchanged between the routers will have the NSSA flag (the N bit in the Options field) set to 1.
- The **default-information-originate** keyword generates a Type-7 default route into an NSSA. This keyword takes effect only on an NSSA ASBR or NSSA ABR. An NSSA ASBR can generate a default route only when it has a default route in its routing table; while an NSSA ABR can generate a default route with or without a default route in its routing table.
- The **no-summary** keyword of the **area {area-id} nssa** OSPF router subcommand prevents summary routes from being advertised into an NSSA – it becomes a Totally Stubby NSSA or Not-So-Stubby-Totally-Stubby area. 🤖 This keyword takes effect only on an NSSA ABR. A single default route would replace both external and summary routes into an NSSA. **Note:** When **no-summary** is configured, the **default-information-originate** keyword is not required for the ABR on a totally stubby NSSA to generate a default route into the area.
- The **no-redistribution** keyword is used in situations where there is no need to redistribute external routes into an NSSA as Type-7 LSAs, eg: when an NSSA ASBR is also an NSSA ABR.
- An ABR in an NSSA never originate Type-4 ASBR-Summary-LSA for the ASBR in the NSSA, as Type-7 NSSA-External-LSA never flooded beyond the border of the NSSA.
- The totally stubby and No-So-Stubby-Totally-Stubby configurations are Cisco-proprietary features. ~~Type 3 Network Summary LSAs (from other areas) are not flooded into both types of stub areas.~~
- Routers in stub area can only establish adjacencies with routers in stub or totally stubby area; while routers in NSSA can only establish adjacencies with the routers in NSSA or totally NSSA.

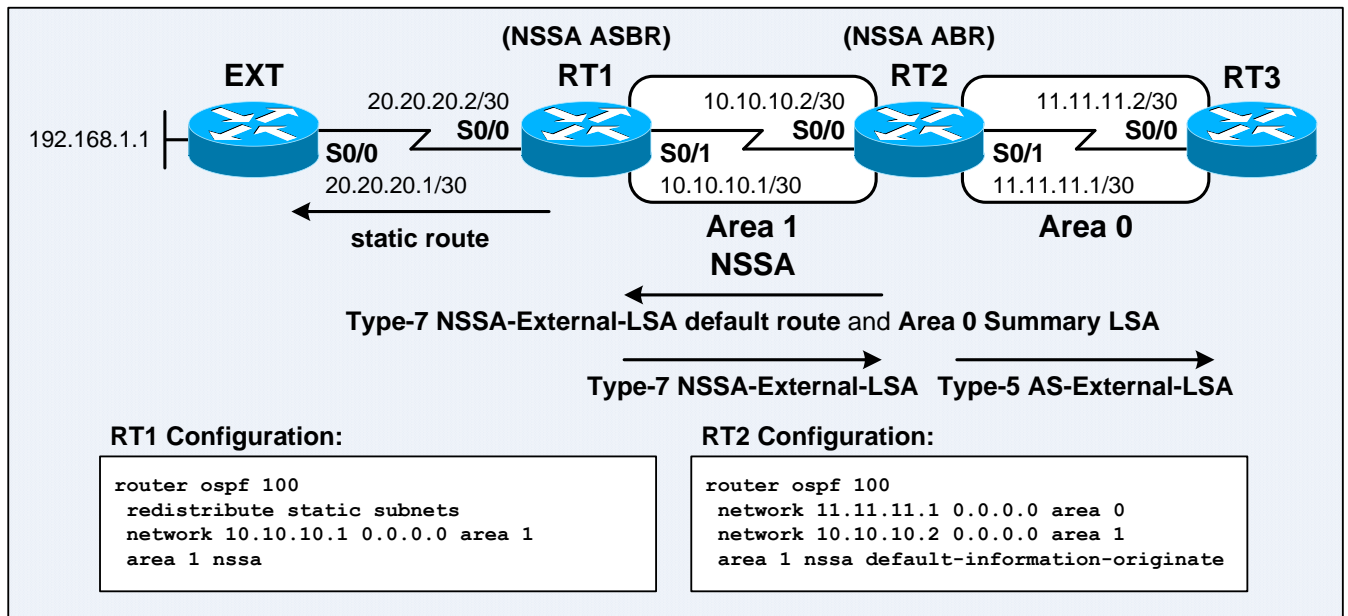


Figure 10-3: OSPF Not-So-Stubby-Area (NSSA)

- Below shows the routing table and OSPF LSDB on RT1:

```

RT1#sh ip route

Gateway of last resort is 10.10.10.2 to network 0.0.0.0

   20.0.0.0/30 is subnetted, 1 subnets
C       20.20.20.0 is directly connected, Serial0/0
   10.0.0.0/30 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, Serial0/1
   11.0.0.0/30 is subnetted, 1 subnets
O IA   11.11.11.0 [110/128] via 10.10.10.2, 00:00:56, Serial0/1
S     192.168.1.0/24 [1/0] via 20.20.20.1
O*N2  0.0.0.0/0 [110/1] via 10.10.10.2, 00:00:57, Serial0/1
RT1#
RT1#sh ip ospf database

          OSPF Router with ID (1.1.1.1) (Process ID 100)

          Router Link States (Area 1)

Link ID      ADV Router    Age         Seq#         Checksum Link count
1.1.1.1      1.1.1.1      67         0x80000002  0x006BE5  2
2.2.2.2      2.2.2.2      69         0x80000001  0x00103C  2

          Summary Net Link States (Area 1)

Link ID      ADV Router    Age         Seq#         Checksum
11.11.11.0   2.2.2.2      64         0x80000001  0x00D5FA

          Type-7 AS External Link States (Area 1)

Link ID      ADV Router    Age         Seq#         Checksum Tag
0.0.0.0      2.2.2.2      69         0x80000001  0x00D0D8  0
192.168.1.0  1.1.1.1      68         0x80000001  0x00EF19  0
RT1#

```

- Below shows the routing table and OSPF LSDB on RT2 and RT3:

```

RT2#sh ip route

Gateway of last resort is not set

    10.0.0.0/30 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, Serial0/0
    11.0.0.0/30 is subnetted, 1 subnets
C       11.11.11.0 is directly connected, Serial0/1
O N2 192.168.1.0/24 [110/20] via 10.10.10.1, 00:00:55, Serial0/0
RT2#
RT2#sh ip ospf
--- output omitted ---
    Area 1
        It is a NSSA area
        Perform type-7/type-5 LSA translation
        generates NSSA default route with cost 1
--- output omitted ---
RT2#sh ip ospf database
--- output omitted ---

                Type-7 AS External Link States (Area 1)

LinkID        ADV Router    Age           Seq#          Checksum Tag
0.0.0.0        2.2.2.2       70           0x80000001   0x00D0D8 0
192.168.1.0    1.1.1.1       71           0x80000001   0x00EF19 0

                Type-5 AS External Link States

Link ID        ADV Router    Age           Seq#          Checksum Tag
192.168.1.0    2.2.2.2       64           0x80000001   0x0066A8 0
RT2#
=====
RT3#sh ip route

Gateway of last resort is not set

    10.0.0.0/30 is subnetted, 1 subnets
O IA   10.10.10.0 [110/128] via 11.11.11.1, 00:00:57, Serial0/0
    11.0.0.0/30 is subnetted, 1 subnets
C       11.11.11.0 is directly connected, Serial0/0
O E2 192.168.1.0/24 [110/20] via 11.11.11.1, 00:00:57, Serial0/0
RT3#sh ip ospf database

                OSPF Router with ID (3.3.3.3) (Process ID 100)

                Router Link States (Area 0)

Link ID        ADV Router    Age           Seq#          Checksum Link count
2.2.2.2        2.2.2.2       73           0x80000001   0x000144 2
3.3.3.3        3.3.3.3       67           0x80000002   0x0095AC 2

                Summary Net Link States (Area 0)

Link ID        ADV Router    Age           Seq#          Checksum
10.10.10.0     2.2.2.2       68           0x80000001   0x005485

                Type-5 AS External Link States

Link ID        ADV Router    Age           Seq#          Checksum Tag
192.168.1.0    2.2.2.2       66           0x80000001   0x0066A8 0
RT3#

```

## OSPF Totally Stubby NSSA

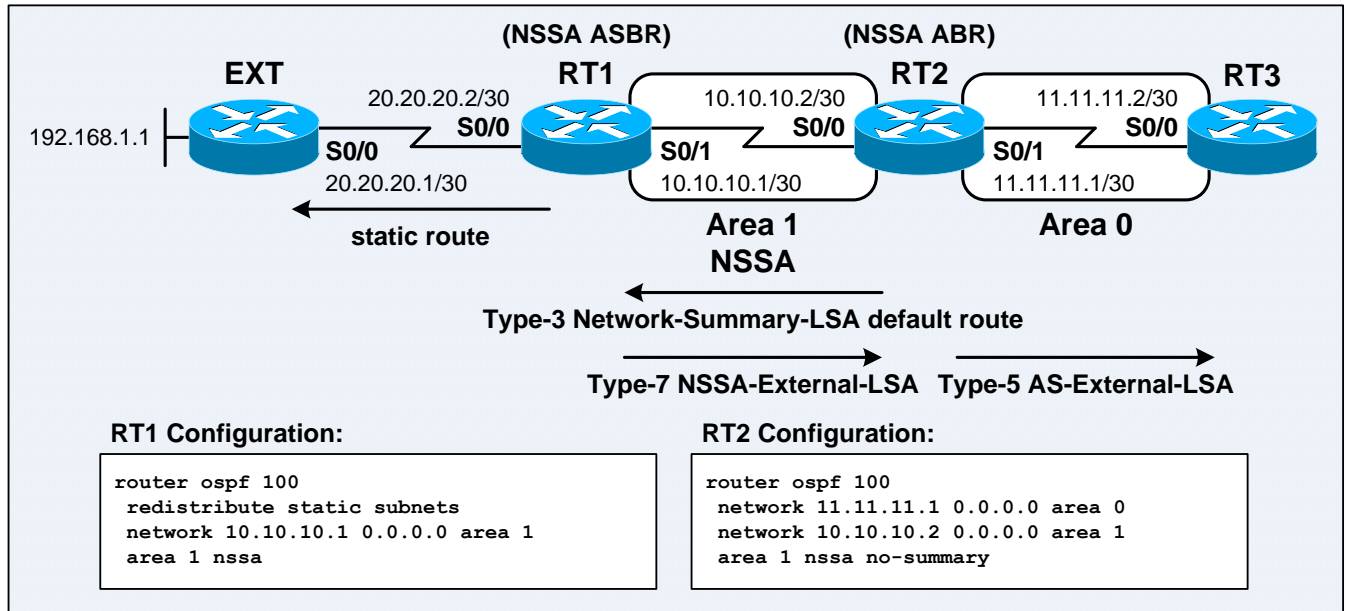


Figure 10-4: OSPF Totally Stubby NSSA

- Below shows the routing table and OSPF LSDB on RT1:

```

RT1#sh ip route
Gateway of last resort is 10.10.10.2 to network 0.0.0.0

 20.0.0.0/30 is subnetted, 1 subnets
 C       20.20.20.0 is directly connected, Serial0/0
 10.0.0.0/30 is subnetted, 1 subnets
 C       10.10.10.0 is directly connected, Serial0/1
 S       192.168.1.0/24 [1/0] via 20.20.20.1
 O*IA 0.0.0.0/0 [110/65] via 10.10.10.2, 00:00:10, Serial0/1
RT1#
RT1#sh ip ospf database

        OSPF Router with ID (1.1.1.1) (Process ID 100)

        Router Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum Link count
1.1.1.1        1.1.1.1      24           0x80000002    0x006BE5  2
2.2.2.2        2.2.2.2      28           0x80000001    0x00103C  2

        Summary Net Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum
0.0.0.0        2.2.2.2      28           0x80000001    0x00FC31

        Type-7 AS External Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum Tag
192.168.1.0    1.1.1.1      26           0x80000001    0x00EF19  0
RT1#
  
```

Area Type	ABRs flood Type-5 External-LSAs into the area?	ABRs flood Type-3 Summary-LSAs into the area?	Allows redistribution of External-LSAs into the stubby area?
<b>Stub</b>	No	Yes	No
<b>Totally Stubby</b>	No	No	No
<b>NSSA</b>	No	Yes	Yes
<b>Totally NSSA</b>	No	No	Yes

Figure 10-5: OSPF Stubby Area Types

## OSPF Default Route

- OSPF generates and advertises default route (0.0.0.0/0) varies upon the OSPF area types that the default route is being advertised into – a standard area, stubby area, totally-stubby area, NSSA, or totally-stubby NSSA.
- By default, OSPF routers in standard areas do not generate default routes into OSPF domains. The **default-information originate [always] [metric metric] [metric-type metric-type] [route-map map]** OSPF router subcommand configures an OSPF router to become an ASBR and generates an External Type-2 (E2) default route with 0.0.0.0 as the Link-State ID and Network Mask, with 1 as the default metric value. It is recommended to advertise OSPF default routes as E2 routes, in which the internal cost or metric towards the ASBR will not be considered when comparing E2 routes. This is important when implementing a primary-backup default routing setup, as the internal cost or metric towards the ASBR will not affect the route selection. **Note:** The Cisco IOS Command Reference for the **default-information originate** command mentions that the default metric value is 10, while testing shows that it is actually 1.
- If the ASBR has a default route in its routing table, the default route can be redistributed into the OSPF routing domain with the **default-information originate** OSPF router subcommand. If the ASBR does not have a default route in its routing table, the **always** keyword can be added to the **default-information originate** command to redistribute a default route into the OSPF routing domain regardless of whether the router has a default route in its routing table.
- Another benefit of the **always** keyword is that it can add stability to the OSPF routing domain, in which an OSPF router will always advertise a default route regardless of the stability of the default route that could be learnt through other routing domains.
- Whether to use the **always** keyword is depends upon the design of the network.  
Ex: When an OSPF router has a non-OSPF external default route (static or dynamically-learnt) towards the Internet, it should advertise the default route only when it has the default route itself. Otherwise it could attract and blackhole the traffic from other routers.
- **Note:** Configure a static default route with the **ip route 0.0.0.0 0.0.0.0 {gateway}** global configuration command and try to redistribute the static route into an OSPF domain with the **redistribute static subnets** OSPF router command is unable to generate a default route!  
**Note:** The default route selected using the **ip default-network** global configuration command is not propagated by OSPF.

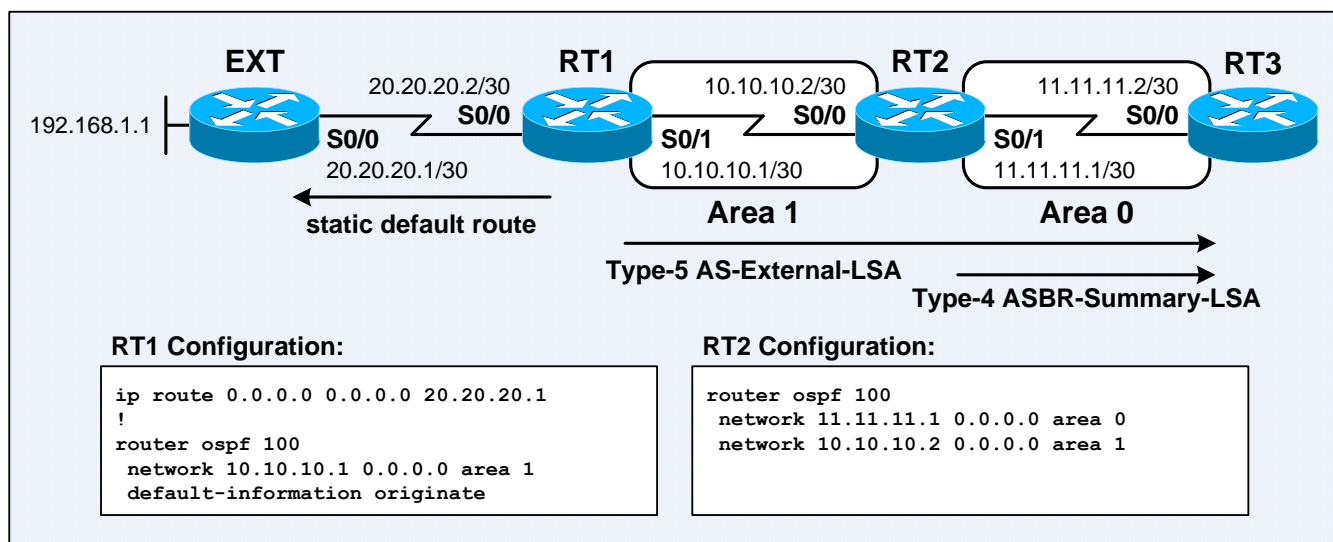


Figure 10-6: OSPF Default Route on Standard OSPF Area

- Below shows the routing table on RT3:

```
RT3#sh ip route

Gateway of last resort is 11.11.11.1 to network 0.0.0.0

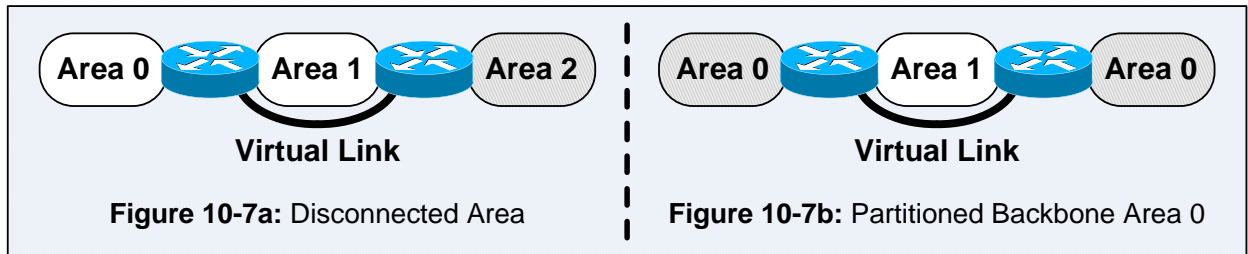
    10.0.0.0/30 is subnetted, 1 subnets
O IA   10.10.10.0 [110/128] via 11.11.11.1, 00:00:01, Serial0/0
    11.0.0.0/30 is subnetted, 1 subnets
C      11.11.11.0 is directly connected, Serial0/0
O*E2  0.0.0.0/0 [110/1] via 11.11.11.1, 00:00:01, Serial0/0
RT3#
```

- In stubby and totally stubby areas, the ABR of the stub area generates a Type-3 Network-Summary-LSA with Link-State ID and Network Mask of 0.0.0.0 into the stub area. The ABR does not need to have a default route in its routing table; and it does not require the **default-information originate** OSPF router subcommand to be configured as well.
- The ABR of an NSSA does not generate a default route by default. The **area area-id nssa default-information-originate [metric metric] [metric-type metric-type]** must be configured on the ABR to generate a Type-7 NSSA-External-LSA with Link-State ID and Network Mask of 0.0.0.0 into the NSSA.
- Another way to generate a default route into an NSSA is the **area area-id nssa no-summary** OSPF router subcommand. As such, the NSSA becomes a totally stubby NSSA. The default route is automatically generated by an NSSA ABR into the totally stubby NSSA as a Type-3 Network-Summary-LSA without the **default-information-originate** keyword of the **area {area-id} nssa** OSPF router subcommand.
- When a stub area has multiple ABRs (or exit points), it is important to ensure the stub routers select the desired ABR to reach the networks outside the stub area. The default route metrics should be identical on all ABRs when the stub routers are allowed to choose the closest ABR; while the ABRs should have different default route metrics (configured with the **area {area-id} default-cost {cost}** router subcommand) when implementing a primary-backup ABRs setup.

## OSPF Virtual Links

- The OSPF 2-tier or 2-layer area hierarchical structure requires that all areas to be directly connected to the backbone area; and the backbone area must be contiguous or else some OSPF areas in the OSPF routing domain will be unreachable. However, the backbone area needs not to be physically contiguous; backbone connectivity can be established and maintained using a logical OSPF virtual link between 2 ABRs. Virtual links belong to the backbone area.

**Note:** The next section explains why all OSPF areas must be connected to the backbone area.



**Figure 10-7:** OSPF Virtual Links

- The idea of a OSPF virtual link is to extend the backbone area across non-backbone area. OSPF virtual links are part of the OSPF open standard and are often being implemented to:
  - i) Connect a disconnected area to the backbone area through a non-backbone area.
  - ii) Connect the 2 separate parts of a partitioned or discontinuous backbone area through a non-backbone area.

The non-backbone area through which a virtual link is configured is known as a **transit area**. A transit area must have full routing information and cannot be a stub area. However, a GRE tunnel can be used instead of a virtual link to encapsulate native OSPF packets across a stub area.

- **Note:** OSPF virtual links should only be used as temporary connections to fix unavoidable network topology problems and should not be part of an initial OSPF network design. A virtual link shows the part of a network that requires review and reengineering. Permanent virtual links show a sign of poorly designed network, as well as add a layer of complexity and troubleshooting difficulty to any network. We should avoid them by ensuring that backbone areas are designed with redundant links to prevent partitioning. When merging networks, sufficient planning is important to make sure all areas are directly link to the backbone area.
- An OSPF virtual link is similar to a standard OSPF adjacency; with the exception that the routers do not have to be directly connected on the same network segment to form an adjacency. A virtual link is interpreted as a point-to-point unnumbered connection.
- The OSPF Hello mechanism works in the same way for both standard and virtual links, in which Hellos are sent out in every 10 seconds. The LSA updates work differently over virtual links. An LSA usually being refreshed every 30 minutes; however, an LSA learnt through a virtual link have the DoNotAge (DNA) bit set, which means that it is not aged out when held in the LSDB. The DNA mechanism suppresses the periodic LSA refresh reflooding over a virtual link.
- The **area {area-id} virtual-link {router-id} [authentication [message-digest | null]] [hello-interval sec] [retransmit-interval sec] [transmit-delay sec] [dead-interval sec] [authentication-key auth-key | message-digest-key key-id md5 key]** OSPF router subcommand defines an OSPF virtual link. It must include the transit area ID (either a decimal value or dotted-decimal notation similar to an IP address) and the Router ID of the corresponding virtual link neighbor to properly configure a virtual link. Utilize the **show ip ospf EXEC** command to ensure the correct Router ID configuration.



- Below describes the parameters available for the **area area-id virtual-link router-id** command:

Parameter	Description
<i>area-id</i>	Specifies an area as the transit area for the virtual link. It can be either a decimal value or in dotted-decimal notation format as like an IP address.
<i>router-id</i>	Specifies the Router ID of the virtual link neighbor.
<b>authentication</b>	Specifies an authentication type.
<b>retransmit-delay</b>	Specifies the interval between LSA retransmissions for adjacencies belonging to the interface. The value must be greater than the expected round-trip delay between any 2 routers on the attached network. The default value is 5 seconds.
<b>transmit-delay</b>	Specifies the estimated time to send an LSU packet out an interface. Its value must be greater than 0. The LSAge of the LSAs in the LSU packets will be incremented by this value before transmission. The default value is 1 second.
<b>authentication-key</b>	Specifies the password used for simple password authentication. The password is a continuous string up to 8 characters.
<b>message-digest-key</b>	Specifies the key ID and key (password) for MD5 authentication. The key is a continuous string up to 16 characters.

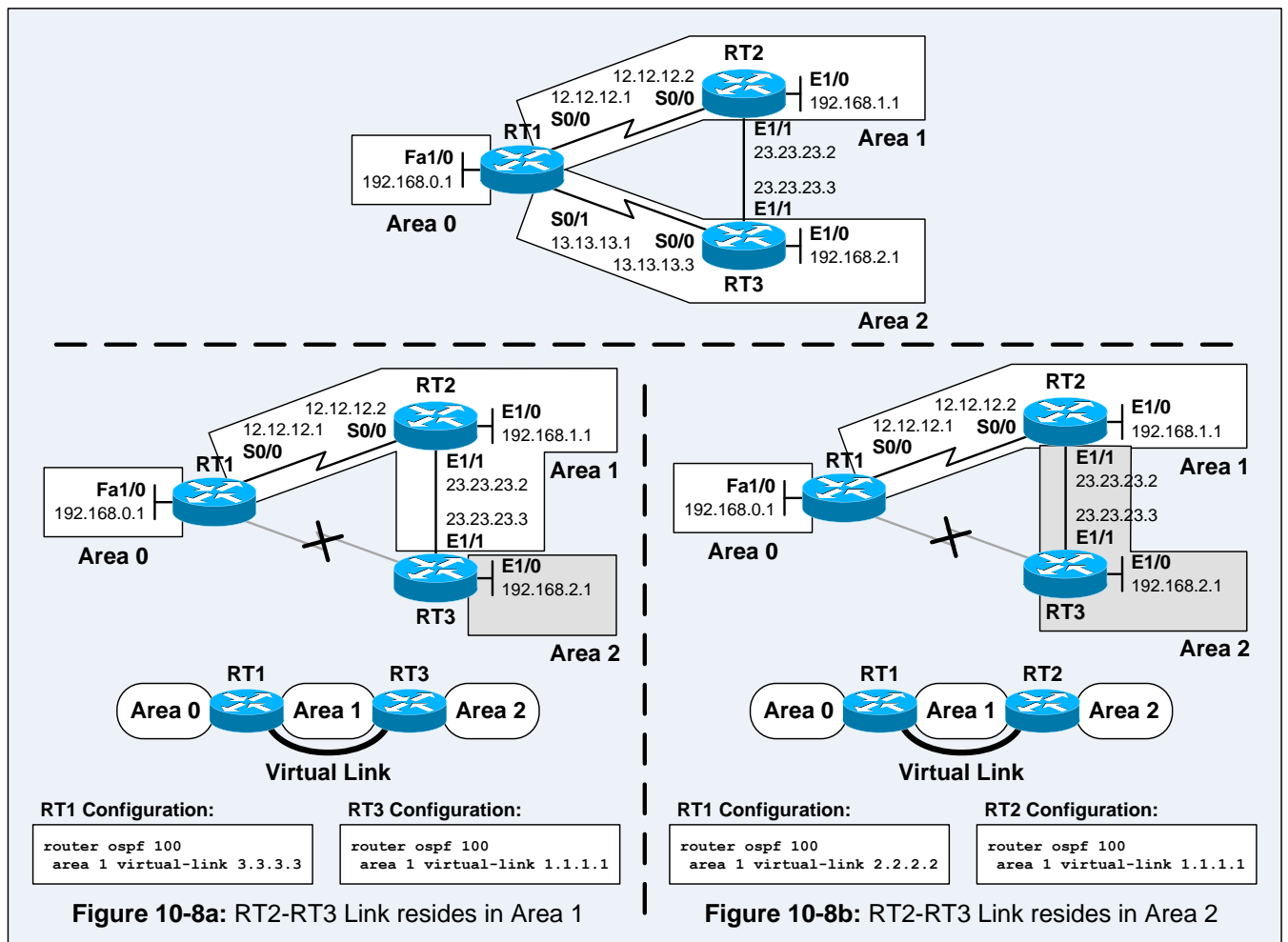


Figure 10-8: Network Setup for Disconnected OSPF Area



- Figure 10-8 shows a typical OSPF network and 2 network setups of disconnected OSPF areas when the link between RT1 and RT3 fails. 2 configuration options are available to connect the disconnected OSPF area back to the backbone area.
- Below shows the routing table and OSPF LSDB on RT1 after a virtual link is configured on RT1 and RT3 for the scenario shown on Figure 10-8a:

```

RT1#sh ip route

Gateway of last resort is not set

    23.0.0.0/24 is subnetted, 1 subnets
O       23.23.23.0 [110/74] via 12.12.12.2, 00:00:47, Serial0/0
C       192.168.0.0/24 is directly connected, FastEthernet1/0
    12.0.0.0/24 is subnetted, 1 subnets
C       12.12.12.0 is directly connected, Serial0/0
O       192.168.1.0/24 [110/74] via 12.12.12.2, 00:00:47, Serial0/0
O IA 192.168.2.0/24 [110/84] via 12.12.12.2, 00:00:37, Serial0/0
RT1#
RT1#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address      Interface
3.3.3.3          0    FULL/ -         -           23.23.23.3  OSPF_VL0
2.2.2.2          0    FULL/ -         00:00:38   12.12.12.2  Serial0/0
RT1#
RT1#sh ip ospf virtual-links
Virtual Link OSPF_VL0 to router 3.3.3.3 is up
  Run as demand circuit
DoNotAge LSA allowed.
Transit area 1, via interface Serial0/0, Cost of using 74
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
  Hello due in 00:00:02
Adjacency State FULL (Hello suppressed)
  Index 1/1, retransmission queue length 0, number of retransmission 1
  First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
  Last retransmission scan length is 1, maximum is 1
  Last retransmission scan time is 0 msec, maximum is 0 msec
RT1#
RT1#sh ip ospf database

        OSPF Router with ID (1.1.1.1) (Process ID 100)

                Router Link States (Area 0)

Link ID        ADV Router    Age           Seq#           Checksum Link count
1.1.1.1        1.1.1.1      46           0x80000002    0x00151D 2
3.3.3.3        3.3.3.3      5            (DNA) 0x80000002    0x00D8A8 1

                Summary Net Link States (Area 0)

Link ID        ADV Router    Age           Seq#           Checksum
12.12.12.0    1.1.1.1      319          0x80000001    0x003C98
12.12.12.0    3.3.3.3      6            (DNA) 0x80000001    0x00645E
23.23.23.0    1.1.1.1      128          0x80000003    0x000F98
23.23.23.0    3.3.3.3      6            (DNA) 0x80000001    0x00548D
192.168.1.0   1.1.1.1      309          0x80000001    0x0095EE
192.168.1.0   3.3.3.3      6            (DNA) 0x80000001    0x003B77
192.168.2.0   3.3.3.3      6            (DNA) 0x80000001    0x00CBEF

--- output omitted ---

```

- Below shows the routing table and OSPF LSDB on RT1 after a virtual link is configured on RT1 and RT2 for the scenario shown on Figure 10-8b:

```

RT1#sh ip route

Gateway of last resort is not set

    23.0.0.0/24 is subnetted, 1 subnets
O IA   23.23.23.0 [110/74] via 12.12.12.2, 00:00:34, Serial0/0
C     192.168.0.0/24 is directly connected, FastEthernet1/0
    12.0.0.0/24 is subnetted, 1 subnets
C     12.12.12.0 is directly connected, Serial0/0
O     192.168.1.0/24 [110/74] via 12.12.12.2, 00:00:44, Serial0/0
O IA   192.168.2.0/24 [110/84] via 12.12.12.2, 00:00:34, Serial0/0
RT1#
RT1#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address        Interface
2.2.2.2          0    FULL/-          -            12.12.12.2    OSPF_VL0
2.2.2.2          0    FULL/-          00:00:39    12.12.12.2    Serial0/0
RT1#
RT1#sh ip ospf virtual-links
Virtual Link OSPF_VL0 to router 2.2.2.2 is up
  Run as demand circuit
  DoNotAge LSA allowed.
  Transit area 1, via interface Serial0/0, Cost of using 64
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:05
  Adjacency State FULL (Hello suppressed)
  Index 1/1, retransmission queue length 0, number of retransmission 1
  First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
  Last retransmission scan length is 1, maximum is 1
  Last retransmission scan time is 0 msec, maximum is 0 msec
RT1#
RT1#sh ip ospf database

        OSPF Router with ID (1.1.1.1) (Process ID 100)

        Router Link States (Area 0)

Link ID        ADV Router    Age           Seq#           Checksum Link count
1.1.1.1        1.1.1.1      43            0x80000002    0x003E02 2
2.2.2.2        2.2.2.2      5             (DNA) 0x80000002    0x00D4E0 1

        Summary Net Link States (Area 0)

Link ID        ADV Router    Age           Seq#           Checksum
12.12.12.0     1.1.1.1      358           0x80000001    0x003C98
12.12.12.0     2.2.2.2      6             (DNA) 0x80000001    0x001EB2
23.23.23.0     2.2.2.2      6             (DNA) 0x80000001    0x007273
192.168.1.0    1.1.1.1      348           0x80000001    0x0095EE
192.168.1.0    2.2.2.2      6             (DNA) 0x80000001    0x00F4CB
192.168.2.0    2.2.2.2      6             (DNA) 0x80000001    0x004E67

--- output omitted ---

```

- **Note:** OSPF does not require that the Router ID IP address to be existed in the IP routing table. As a result, the Router ID configured in the **area virtual-link** command may not be pingable, but the virtual link still work.

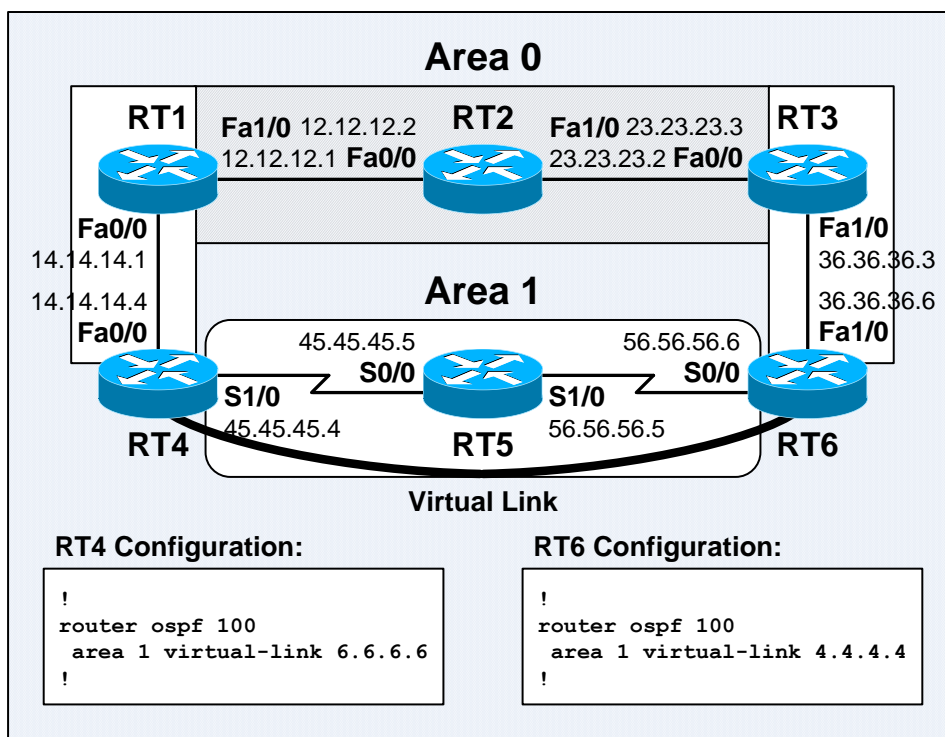


Figure 10-9: Network Setup for Partitioned Backbone Area

- Figure 10-9 shows a sample implementation of OSPF virtual link – RT2 which resides in Area 0 has failed and the routers within the OSPF routing domain have lost their network connectivity. **Note:** OSPF virtual links should only be used as temporary connections to fix unavoidable network topology problems.
- Below shows the output of the **show ip ospf neighbor** and **show ip ospf virtual-links** EXEC commands on RT4 upon configured the OSPF virtual link:

```
RT4#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address        Interface
6.6.6.6          0     FULL/-         -           56.56.56.6    OSPF_VL0
1.1.1.1          1     FULL/BDR       00:00:38   14.14.14.1    FastEthernet0/0
5.5.5.5          0     FULL/-         00:00:39   45.45.45.5    Serial1/0
RT4#
RT4#sh ip ospf virtual-links
Virtual Link OSPF_VL0 to router 6.6.6.6 is up
  Run as demand circuit
  DoNotAge LSA allowed.
  Transit area 1, via interface Serial1/0, Cost of using 128 [1]
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
  Hello due in 00:00:02
  Adjacency State FULL (Hello suppressed)
  Index 2/3, retransmission queue length 0, number of retransmission 0
  First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
  Last retransmission scan length is 0, maximum is 0
  Last retransmission scan time is 0 msec, maximum is 0 msec
RT4#
```

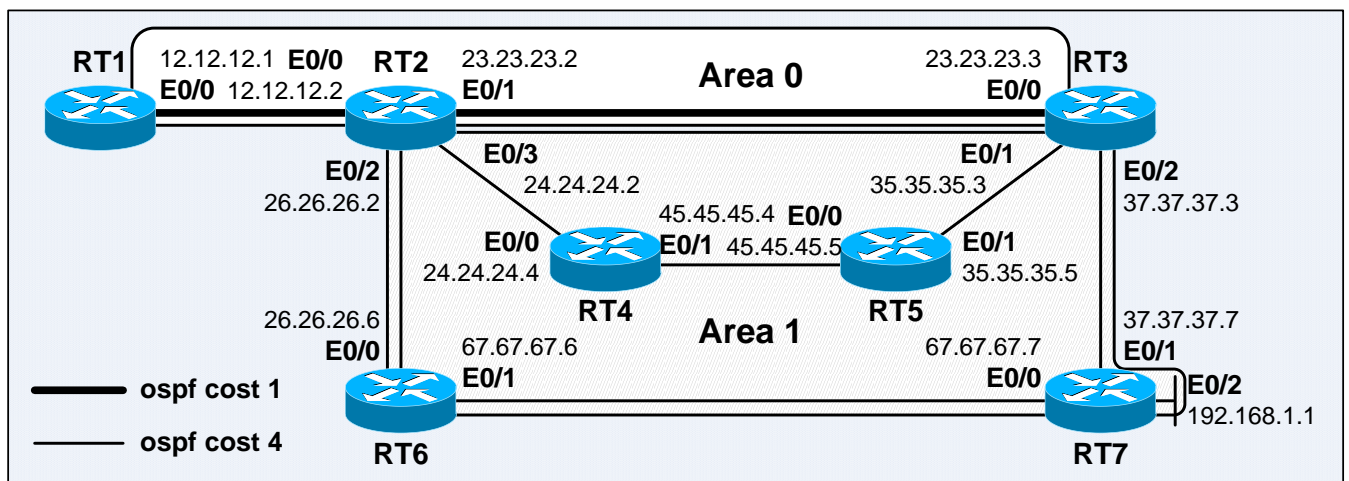
[1] – The cost to reach the virtual link neighbor through the intra-area path; it cannot be configured.

- **Note:** Configure authentication on a virtual link when the backbone area 0 is using authentication! A virtual link is really an extension of the backbone area; if the backbone area is running authentication, the virtual link must be configured for authentication as well.
- A virtual link is treated as an unnumbered point-to-point network that belongs to the backbone and joins the 2 area border routers. An adjacency will be established over the virtual link. When the adjacency is established, the virtual link will be included in backbone router-LSAs, and OSPF packets belong to the backbone area will flow over the adjacency – *virtual adjacency*.
- **Note:** When configuring a virtual link over an interface which has the maximum OSPF cost – 65536 or 0xffff, the virtual link will not come up. RFC 2328 – OSPF Version 2 quotes that a virtual link whose underlying path has cost greater than hexadecimal 0xffff (the maximum size of an interface cost in a router-LSA) should be considered inoperational.

### When OSPF Becomes a Distance Vector Routing Protocol

- In this final section of OSPF main text we revisit some basic and important concepts of OSPF. We were always being told that OSPF is a fast converging loop-free link-state routing protocol. However, to be precise, OSPF is a link-state routing protocol only within an area (intra-area); but almost a distance-vector routing protocol between areas (inter-area).
- OSPF requires all traffic between non-backbone areas to pass through the backbone area. The reason behind this leads us back to the fundamental concepts of OSPF.
- Every OSPF router floods information about its links, and its neighbors to other OSPF routers. Each router then builds an identical link-state database based on the flooded information, and then independently runs the SPF calculation upon its LSDB to derive the SPF tree, which is sort of a map of shortest paths to other routers (local calculation using distributed information).
- One of the advantages of LS protocols is that the LSDB provides a “view” of the entire network, which in turn able to prevent most routing loops. This is in contrast to DV protocols, in which routing information is being propagated on a hop-by-hop basis throughout the network, and a calculation is performed at each hop (distributed calculation using local information). Each router along a path is dependent on the router before it to perform its calculations correctly and then pass along the results. When a router advertises the routing information it learnt to its neighbors, it is basically saying “I know how to reach these destinations.”. Since every DV router knows only what its neighbors tell it, and has no “view” of the network beyond their neighbors, the protocol is vulnerable to loops.
- Those were the days when memory resources and CPU processing power were limited; it became a scaling problem when a link-state routing domain grew larger with a large number of routers, and large size of the LSDBs. The problem is solved by partitioning the routing domain into areas. The flooding occurs within the boundary of an area, and the LSDBs contain only information from other routers within the area. This means that the SPF tree on each router only describes only the paths to other routers within an area; and there are separate LSDBs for different areas.
- OSPF areas are connected by one or more area border routers (ABRs) which maintain separate LSDBs and calculate separate SPF trees for each area that they reside in. An ABR is a member of multiple areas. It advertises the prefixes it learnt in an area to other areas by flooding Type-3 LSAs into those areas which basically saying “I know how to reach these destinations.”.  
**Note:** Another main LS protocol – IS-IS, connects areas somewhat differently.

- The last concept described earlier is not link-state, it is distance-vector! The routers in an area have no “view” beyond the ABR, and rely upon the ABR to tell them the prefixes it can reach. The SPF calculation on the routers within an area derives a SPF tree that depicts all prefixes beyond the ABR as leaf subnets connected to the ABR with some specified costs.
- This leads us to the answer to the question why OSPF requires all traffic between non-backbone areas to pass through the backbone area in a hub-and-spoke topology. Because inter-area OSPF is distance-vector, it is vulnerable to routing loops. Hence OSPF specifies that traffic from one area can only reach another area through the backbone area in order to maintain a loop-free inter-area routing topology, in which no loops will be formed between the areas.
- Jeff Doyle, the author of the Routing TCP/IP series, had involved in many discussions regarding the scalability of OSPF. He came across a single-area OSPF network with over 1000 routers, yet still operates without any performance or memory problem. Always design OSPF networks with the single-area mindset and add non-backbone areas only when they are really necessary. Defending a single-area design is a matter of convincing that multiple areas are not necessary. 😊  
A cardinal rule of network design is to start as simple as possible and then add complexities only when they are the simplest solution to an identified problem. – Jeff Doyle.
- Suboptimal routing can occur in OSPF area routing due to the fact that an internal router not in the destination area will use the routing path via the first segment (ABR) to the destination area.



**Figure 10-10: Suboptimal OSPF Inter-Area Routing**

- Figure 10-10 shows a sample OSPF network with high-speed high-capacity links (OSPF cost 1), in the backbone area, and low-speed low-capacity links (OSPF cost 4) in Area 1. When all routers and links reside in a single area, plain shortest path routing will follow the backbone as much as possible; therefore the traffic from RT1 to 192.168.1.0/24 behinds RT7 will follow the route: RT1 – RT2 – RT3 – RT7.
- However, due to inter-area routing and RT1 does not reside in the destination area (Area 1), RT2 will route traffic via RT6 when the traffic arrives at RT2, as it resides in the destination area. The routing path will be changed to RT1 – RT2 – RT6 – RT7; even it has higher accumulated OSPF path cost than the optimal path (13 against 10)!

## Route Redistribution and Manipulating Routing Updates

- Simple routing protocols work well for simple networks. However as a network grows and becomes more complex, and the original routing protocol may not be the best choice anymore and it is necessary to change the routing protocols. Transition between routing protocols must be well-planned and implemented gradually, in which multiple routing protocols may run in the network for some time.
- Whatever the reason for changing the routing protocol, network administrators must manage the transition and migration carefully and thoughtfully. Below lists some important points for that:
  - An accurate network topology diagram and an inventory of all network devices are critical for success.
  - The new routing protocol will definitely have different requirements and capabilities. Network administrators must understand and know what must be changed and create a detailed migration plan prior to implement any changes.
  - There will likely be a time when both routing protocols are running in the network during the transition and migration. Route redistribution strategy must be carefully planned to avoid disrupting production traffic or causing suboptimal routing.
  - Determine the timing of the migration – change the entire network at once, or done in stages?
  - Will the transition and migration require the change of the IP addressing scheme?
- Networks may run multiple routing protocols as part of their design, not only as part of a transition and migration. Therefore, route redistribution might be required in such cases as well.
- Resources and schedules must be well-considered when migrating multiple remote sites to a new IP address space. The address plan created for the migration must be well-documented and accessible for review and reference by all networking personnel. Those documents helps to settle any questions or conflicts when there is any. Ex: The new address space might have portions already in use and not known by the designer. Having other networking personnel to review and agree upon the address assignments helps prevent problems during the implementation phase.
- Plan the implementation steps after the IP addressing scheme has been determined. Carefully consider the following for successful implementation of a new IP addressing scheme:
  - **Host addressing.** If host IP addresses are statically assigned, this is an excellent time to migrate to use DHCP. If DHCP is already in use, configures the new IP address scopes and the changes of IP addressing will be transparent to most end users.
  - **Access lists and other filters.** Firewalls and other types of traffic filters must have been configured for the old IP addresses. It is important to have complete documentation of all the traffic filters so that they can be updated upon using the new IP addresses. Any route filtering based on the old IP addresses must also be updated.
  - **Network Address Translation (NAT).** If NAT is being used in the network, it must also be modified to recognize the process the new IP addresses. The new IP addresses also might need to be translated to different outside local addresses.
  - **Domain Name System (DNS).** If DNS servers are being used in the network, decide the mappings that must be modified to reflect the new IP addresses.
  - **Timing.** Changes are typically done in stages for a large network. Determine to start at the core and work outward or start at the edges and work inward. Be sure to allocate enough time to test and verify the new configuration.
  - **Transition strategy.** Decide whether to use only the new IP addressing during the migration or whether the old and new IP addressing schemes exist during the transition. Secondary IP addresses may be configured on router interfaces for the second approach.

- **Note:** Some routing protocols have issues with secondary addresses – EIGRP and OSPF use the primary IP address of an interface as the source of their packets. They expect the routers reside on the same subnet. They do not form a neighbor relationship with a router on the wrong subnet. Uses the IP addresses from the same subnet for the primary addresses on neighboring router; do not use the IP addresses from the same subnet for the primary address on one router and the secondary address on its neighbor.
- Consider the following steps when planning the migration to a new routing protocol:
  - Safe-guard the latest configurations of all network devices, as well as network documentation that include information about traffic-flow paths to ensure that the changes will not create suboptimal paths or routing loops. The documentation should also include baseline statistics for data flows.
  - Define a clear and comprehensive timeline for all migration steps.
  - Determine which routing protocol is the core and which is the edge, and decide whether starting the migration at the core of the network and working out to the edges; or starting at an edge and working in toward the network core. Each approach has its pros and cons.
  - Identify the boundary routers between the multiple running routing protocols. Migrating to a new routing protocol often requires route redistribution between the old and new routing protocols.
  - Determine how to redistribute information between the core and edge routing protocols.
  - Verify that all devices support the new routing protocol. Perform Cisco IOS software updates when deem fit prior to the migration.
  - Implement and test the routing design solution in a lab environment. The migration strategy should be tested in an environment that is as realistic as possible to identify and correct any problem ahead.

## Route Redistribution

- Below are some possible reasons to run multiple routing protocols within an organization:
  - Migrating from an older IGRP to a new IGP. Multiple redistribution boundaries may exist until the old protocol has been completely replaced by the new protocol.
  - Want to use another protocol but need to keep the old protocol due to the limitations of end systems. Ex: Legacy UNIX host-based routers might support only RIP.
  - Merging of organizations with different IGPs.
  - Some departments do not want to upgrade their routers to support a new routing protocol.
  - When there is a mixed-router vendor network environment, the Cisco-specific routing protocol – EIGRP can be deployed in the Cisco portion of the network; and a public standard-based routing protocol – OSPF for non-Cisco devices.
- Cisco IOS allows internetworks using different routing protocols to exchange routing information through a feature called **route redistribution**. Redistribution is the capability of boundary routers that interconnecting different routing domains to exchange and advertise routing information between routing domains. The boundary router must be running all the routing protocols that intend to exchange routes. Routers can redistribute static routes and directly-connected networks as well as routes from other routing protocols into a routing domain.
- Redistribution is always performed **outbound**. The router that performing redistribution does not change its routing table. Ex: After configured redistribution between EIGRP and OSPF, the EIGRP process on the boundary router advertises the OSPF routes in its routing table as EIGRP routes to its EIGRP neighbors, and vice versa. The neighbors of the boundary router interpret the redistributed routes as external routes.

- Routes must be in the routing table for them to be redistributed; this rule often confused us. ☺  
For instance, a router learns about a network via EIGRP and OSPF; only the EIGRP route will be inserted into the routing table, as it has a lower administrative distance. Suppose RIP is also running on this boundary router and redistributing OSPF routes into RIP. The mentioned network will not be redistributed into RIP, as it is considered as an EIGRP route instead of an OSPF route in the routing table.
- Route redistribution is powerful, yet adds complexity and increase confusions upon a network; therefore it should be used only when necessary. Below lists some main issues of redistribution:
  - **Routing feedback or domain loop.** When multiple boundary routers are performing redistribution, routing information received from an AS can be sent back to the same AS.
  - **Incompatible routing information.** Since each routing protocol uses different metrics to determine the best path and the metric information cannot be translated exactly between routing protocols, path selection using the redistributed information may not be optimal. The relative metrics of redistributed routes are lost upon route redistribution. Redistributed routes should be assigned a **seed metric** that is higher than the largest native metric for all routes in the receiving AS in order to prevent suboptimal routing.
  - **Inconsistent convergence times.** Different routing protocols converge at different rates. Proper planning ensures that these issues do not cause problems in the network.
- Routing protocols employ different metric structures and algorithms that are incompatible with each other. In a network in which multiple routing protocols are present, the exchange of route information and the capability to select the best path across the multiple protocols is critical. Cisco IOS considers the following 2 parameters (in sequence) to select the best path when it learns multiple routes to the same destination from different routing protocols.
  - **Administrative distance** that is used to rate the believability of a routing protocol. Each routing protocol is prioritized in order from most to least believable or reliable using the AD value. A router uses this criterion to determine which routing protocol to believe when multiple routing protocols provide routing information to the same destination. Lower administrative distance values indicate greater believability.
  - **Routing metric** – A value that represents the cost from the router and the destination.
- When implementing route redistribution, it is occasionally need to modify the administrative distance of a routing protocol so that its routes are being preferred.
- Redistributed routes are not directly connected to a router; they are learned from other routing protocols. A boundary router is able to translate the metric from the source routing protocol into the destination routing protocol. This metric is referred to as the **seed / initial / default metric**. It is defined upon redistribution configuration. After the seed metric for a redistributed route is defined, the metric increments normally within the routing domain; except for OSPF E2 routes, which hold their seed metric throughout an OSPF routing domain.
- The **default-metric** router subcommand defines the seed metric for all redistributed routes. Cisco IOS defines the seed metric as part of the **redistribute** router subcommand either with the metric option or using a route map.
- A redistributing router must assign a metric upon the redistributed routes. For directly connected networks, the seed metric is derived from the characteristics of the router interface. For RIP, the seed metric starts with a hop count of 0. For IGRP and EIGRP, the seed metric is based on the interface bandwidth and delay. For OSPF, the seed metric is based on the bandwidth of the interface. For IS-IS, the seed metric is 0. The seed metric increments as usual as the routing information being propagated from router to router in the routing domain.



- Below shows the default seed metric value for routes that are redistributed into each IP routing protocol. RIP, IGRP, and EIGRP interpret the seed metric of 0 as infinity by default. Infinity metric indicates that the route is unreachable and therefore should not be advertised. Hence, a seed metric must be specified when redistributing routes into RIP, IGRP, and EIGRP; or the redistributed routes will not be advertised.

Protocol that Route is Redistributed Into	Default Seed Metric
RIP	Infinity (no routes are inserted into the routing table)
IGRP / EIGRP	Infinity (no routes are inserted into the routing table)
OSPF	20 for all except BGP, which is 1
IS-IS	0 (routes are inserted into the routing table)
BGP	The BGP MED metric is set to the IGP metric value (maintain)

For OSPF, the redistributed routes have a default External Type 2 (E2) metric of 20; except for redistributed BGP routes, which have a default External Type 2 (E2) metric of 1. For IS-IS, the redistributed routes have a default metric of 0. However, IS-IS does not treat a seed metric of 0 as unreachable as with RIP, IGRP, and EIGRP.

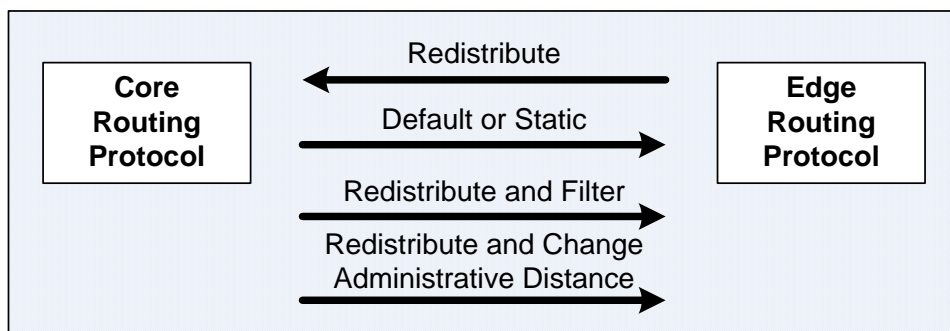
For BGP, the redistributed routes maintain the IGP metrics.

**Note:** Redistributing between EIGRP processes is not required to define the metric using the **metric** parameter in the **redistribute** router subcommand or the **default-metric** command; EIGRP conserves the metric upon redistributing between EIGRP processes.

- Below are 2 methods of performing route redistribution:
  - **One-way redistribution** – Passes a default route into a routing protocol and redistributes only the networks learned from the routing protocol into other routing protocols. This is the safest way to perform redistribution, which redistributes routes in only one direction, on only one boundary router within the network. However, note that having only a single boundary router introduces a single point of failure in the network.
 

**Tips:** Implement default routes to facilitate the use of one-way redistribution, if necessary.

Ex: When redistributing between an OSPF routing domain and a RIPv2 protocol domain, use the **default-information originate [always]** OSPF router subcommand to generate a default route into OSPF domain, and redistribute the OSPF routes into the RIPv2 domain.
  - **Two-way redistribution** – Redistributes all routes between the 2 routing protocols.
- Network engineers must determine the core and edge routing protocols. The core routing protocol is the main routing protocol running in the network. During the transition period between routing protocols, the core and edge routing protocols are the new and old routing protocol respectively. For networks that run multiple routing protocols all the time, the core is usually the more advanced routing protocol.
- If redistribution must be performed in both directions, and/or on multiple boundary routers, the redistribution configuration should be tuned to avoid suboptimal routing and domain loops.



**Figure 11-1: Route Redistribution Techniques**

- Below are some redistribution techniques that can be applied depends upon the network design:
  - Redistribute a default route from the core routing protocol into the edge routing protocol, and redistributes routes from the edge routing protocol into the core routing protocol. This technique helps prevent suboptimal routing, route feedback, and domain loop.
  - Redistribute multiple static routes about the core routing protocol into the edge routing protocol, and vice versa. This method works if there is only a redistribution point (boundary router); multiple redistribution points may cause route feedback / domain loop.
  - Redistribute routes from the core routing protocol into the edge routing protocol with filtering to block out inappropriate routes. For instance, when there are multiple boundary routers, routes redistributed from the edge routing protocol at a boundary router should not be redistributed from the core routing protocol back into the edge routing protocol on another boundary router (or redistribution point).
  - Redistribute all routes from the core routing protocol into the edge routing protocol, and from the edge routing protocol into the core routing protocol; and then modify the administrative distance associated with the redistributed routes so that they are not the selected routes when multiple routes exist for the same destination. For instance, the administrative distance might need to be modified if the route learned by the core routing protocol is better but has a higher (less believable) administrative distance.
  
- Redistribution supports all routing protocols. Static and directly connected routes can also be redistributed to allow the routing protocol to advertise these routes. Routes are redistributed **into** a routing protocol; therefore the redistribute router subcommand is configured under the routing protocol that is going to **receive** the redistributed routes.

```

Router(config-router)#redistribute ?
  bgp          Border Gateway Protocol (BGP)
  connected    Connected
  eigrp        Enhanced Interior Gateway Routing Protocol (EIGRP)
  isis         ISO IS-IS
  iso-igrp     IGRP for OSI networks
  metric       Metric for redistributed routes
  mobile       Mobile routes
  odr          On Demand stub Routes
  ospf         Open Shortest Path First (OSPF)
  rip          Routing Information Protocol (RIP)
  route-map    Route map reference
  static       Static routes
  <cr>

Router(config-router)#

```

- Consider the following points before implementing redistribution:
  - Cisco IOS allows only redistribution between routing protocols that support the same protocol stack. Ex: Redistribution between IP RIP and OSPF is possible because they both support the TCP/IP stack. Redistribution between IPX RIP and OSPF because IPX RIP supports the IPX/SPX (Internetwork Packet Exchange / Sequenced Packet Exchange) stack which OSPF does not support. Although EIGRP has different protocol-independent modules for IP, IPX, and AppleTalk, routes cannot be redistributed between them, because each PDM supports a different protocol stack.
  - Redistribution configuration varies upon the different routing protocols. Some routing protocols require a metric to be specified upon redistribution configuration.

- The **redistribute** {*src-protocol*} [*proc-id* | *isis-area-tag*] {**level-1** | **level-1-2** | **level-2**} [**match** {**internal** | **external** [**1** | **2**] | **nssa-external** [**1** | **2**] }] [**metric** {*metric-value* | **transparent**}] [**metric-type** *metric-type*] [**tag** *tag-value*] [**route-map** *map-tag*] [**subnets**] router subcommand redistributes routes from a routing domain into another routing domain.

<i>src-protocol</i>	Source protocol from which routes are being redistributed. It can be one of the <b>bgp</b> , <b>connected</b> , <b>eigrp</b> , <b>isis</b> , <b>ospf</b> , <b>rip</b> , or <b>static [ip]</b> keywords. The static keyword is used to redistribute IP static routes. The ip keyword is used when redistributing IP static routes into the Integrated IS-IS routing protocol.
<i>proc-id</i> , <i>isis-area-tag</i>	For the <b>bgp</b> or <b>eigrp</b> keyword, it is an autonomous system number. For the <b>ospf</b> keyword, it is an OSPF routing process ID. For the <b>isis</b> keyword, it is an area tag value of a routing process. It is not applicable for the <b>rip</b> keyword.
<b>level-1</b> , <b>level-1-2</b> , <b>level-2</b>	Specifies the routes of the corresponding IS-IS routing levels that are being redistributed into other IP routing protocols.
<b>internal</b> , <b>external</b> , <b>nssa-external</b>	Used to specify the type of OSPF routes that are to be redistributed.
<i>metric-value</i>	Defines the seed metric for redistributed routes.
<b>transparent</b>	For RIP to use the routing table metric as the RIP metric for redistributed routes.
<i>type-value</i>	For OSPF, specifies the external route type – E1 or E2 (default) associated with the redistributed routes. For IS-IS, it can be one of the following 2 keywords: <ul style="list-style-type: none"> <li>▪ <b>internal</b> – Assign an IS-IS metric that is &lt; 63 (default).</li> <li>▪ <b>external</b> – Assign an IS-IS metric that is &gt; 64 &lt; 128.</li> </ul>
<i>tag-value</i>	An optional 32-bit unsigned decimal value attached to an OSPF external route. This is not being used by OSPF itself. It may be used to communicate information between ASBRs. If none is specified, then the source autonomous system number is used for routes redistributed from BGP and EGP. Not applicable to other protocols.
<i>map-tag</i>	Specifies a route map that filters the routes redistributed from the source routing protocol. If not specified, all routes are redistributed. If the specified route map does not exist, no routes are redistributed.
<b>subnets</b>	Indicates the scope of redistribution for the specified protocol when redistributing routes into OSPF. If this keyword was omitted, only classful networks (no subnets) will be redistributed into the OSPF domain. Omitting this keyword is a common OSPF configuration error.

- Redistribution into OSPF and IS-IS can also be limited to a number of prefixes using the **redistribute maximum-prefix** {*max-value*} [*threshold*] [**warning-only** | **withdraw**] router subcommand. The *threshold* value defaults to log a warning message at 75 percent of the defined *max-value*. No further routes will be redistributed after reaching the defined *max-value*. If the **warning-only** keyword is configured, redistribution is not limited to the *max-value*; it simply logs another warning message when the redistributed prefixes reached the *max-value*. The optional **withdraw** keyword that only available for IS-IS will cause IS-IS to rebuild LSPs without the external (redistributed) IP prefixes. This command was introduced in Cisco IOS 12.0(25)S Release and was integrated into Cisco IOS 12.2(18)S and 12.3(4)T Releases and later.

- When redistributing into RIP, the default metric is infinity; except when redistributing a static (including a static default route defined using the **ip route 0.0.0.0 0.0.0.0** global command) or directly-connected network, in which the default metric is 1.
- When redistributing routes into EIGRP, the default metric is infinity; except when redistributing a static or directly-connected route into EIGRP, the default metric is equal to the metric of the associated static route or directly-connected interface. A metric must be specified to ensure that the routes are redistributed. The redistributed routes appeared as external EIGRP (D EX) routes in the routing table. External EIGRP routes have a higher administrative distance (170) than internal EIGRP routes (90); therefore internal EIGRP routes are preferred over external EIGRP routes.
- When redistributing into OSPF, the default metric is 20 (except for BGP routes, which is 1), the default external metric type is E2, and subnets are not redistributed by default. If the **subnets** keyword is used. If this keyword were omitted, no subnets would be redistributed into the OSPF routing domain. Omitting this keyword is a common OSPF configuration error.
- External routes are introduced into IS-IS as Level 2 with a metric of 0 by default. When redistributing IS-IS routes into other routing protocols, options are available to include L1, L2, or both L1 and L2 routes. If no level is specified, all routes are being redistributed.

```

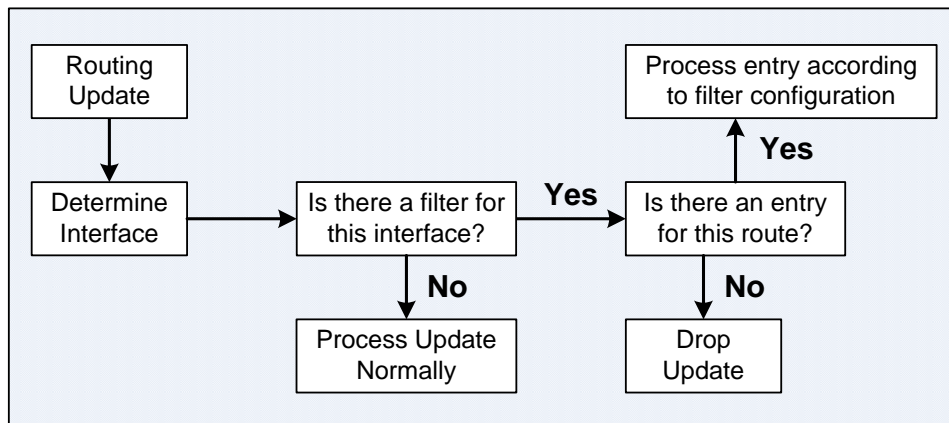
Router(config)#router ospf 100
Router(config-router)#redistribute isis ?
  WORD          ISO routing area tag
  level-1       IS-IS level-1 routes only
  level-1-2     IS-IS level-1 and level-2 routes
  level-2       IS-IS level-2 routes only
  --- output omitted ---

```

- The **default-metric** *{metric-value | {bandwidth delay reliability loading mtu}}* router subcommand specifies the default metric that applied upon all the redistributed routes. Using this command eliminates the need to define metrics separately for each redistribution.
- The **metric** parameter in the **redistribute** router subcommand can set a different default metric for each routing protocol that is being redistributed. The metric value configured in such way overrides the value specified in the **default-metric** command for a particular routing protocol.
- The **passive-interface** *{type num | default}* router subcommand prevents routing updates for a routing protocol from being sent out a router interface, but still receives routing updates on that interface (RIP and IGRP); and prevents Hello packets from being sent out a router interface, therefore do not establish unwanted neighbor relationship with other routers (EIGRP and OSPF).
- The **default** keyword set all router interfaces for as passive interfaces a routing protocol. The **passive-interface default** router subcommand is often implemented in distribution routers in Internet service providers and large enterprise networks, which have more than 100 interfaces. Prior to the introduction of the **Default Passive Interface** feature in Cisco IOS Release 12.0, network administrators would enable the routing protocol on all interfaces and then manually define the interfaces that did not require adjacency using many **passive-interface** commands. The **passive-interface default** command is now available to set all interfaces to passive, followed by enable interfaces that require adjacencies using the **no passive-interface** *{intf-type intf-num}* command.

## Manipulating Routing Updates

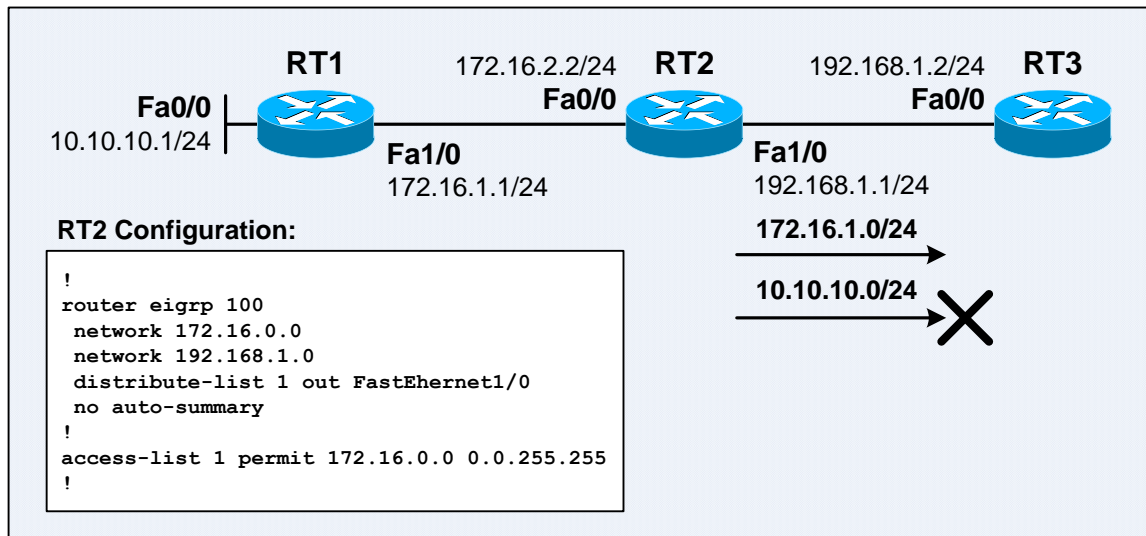
- Routing updates compete with user data for bandwidth and router resources; yet they are critical as they carry information for routers to make routing decisions. Network administrators must control and tune routing updates to ensure that the network operates efficiently. Network information must be sent where it is needed and filtered from when it is not needed. No single route filtering method fits all situations; therefore it is essential to know all the following available methods for manipulating routing updates:
  - **Distribute lists** – apply access lists upon routing updates to filter unnecessary routes.
  - **Route maps** – powerful but complicated route filtering and manipulation tools.
  - **Administrative distance** – controlling the route preference.
  
- Blocking the advertisement of certain routes (route filtering) is a solution that is often being implemented to prevent domain loops when implementing two-way route redistribution with multiple redistribution points.
  
- Access list are configured in global configuration mode; and the associated distribute list is configured under a routing protocol process. The access list should permit the networks that will be received, advertised, or redistributed and deny the networks that will remain hidden. The router then applies the access list upon routing updates for the routing protocol. The **distribute-list** {[acl-num | acl-name] | **prefix** {ip-prefix-name} | **route-map** {map-tag}} {in [intf-type intf-num] | out [intf-type intf-num | routing-process [as-num]]} router subcommand filters routing updates based on incoming interface, outgoing interface, and redistribution from another routing protocol.



**Figure 11-2:** Distribute List Processing Based on the Incoming or Outgoing Interface

- Routing updates can be controlled at both the interface and routing protocol levels. Figure above shows the process of a router when filtering routing updates using a distribute list that is based on the incoming or outgoing interface. Below lists the steps of the processing:
  - The router receives or prepares to send a routing update about one or more networks.
  - The router determines the interface on which an incoming routing update has arrived; or the interface out of which an outgoing routing update should be advertised.
  - The router determines if a filter (distribute list) is associated with the interface.
  - If a filter (distribute list) is not associated with the interface, the update is processed normally.
  - If a filter (distribute list) is associated with the interface, the router processes the access list referenced by the distribute list for a match upon the route specified in the routing update.
  - If there is a match in the access list, the route entry is processed as configured – which is either permitted or denied by the matching access list statement.
  - If no match is found in the access list, the implicit **deny any** at the end of the access list drops the route entry.

- The **distribute-list out** router subcommand cannot be used with link-state routing protocols to block outbound LSAs for an interface. The routes are not inserted in the local routing table, but are still placed in the link-state database.
- The **distribute-list in** router subcommand filters routing updates **going into** the interface specified in the command into the routing process under which it is configured. The **distribute-list out** router subcommand filters routing updates **going out** from the interface or routing protocol specified in the command, into the routing process under which it is configured.



**Figure 19-3: IP Route Filtering**

- Below shows the routing table on RT3 before and after the route filtering configuration on RT2:

```

RT3#sh ip route

Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 1 subnets
D    172.16.1.0 [90/30720] via 192.168.1.1, 00:00:08, FastEthernet0/0
    10.0.0.0/24 is subnetted, 1 subnets
D    10.10.10.0 [90/33280] via 192.168.1.1, 00:00:08, FastEthernet0/0
C     192.168.1.0/24 is directly connected, FastEthernet0/0
RT3#
00:05:20: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 192.168.1.1
(FastEthernet0/0) is down: Interface Goodbye received
00:05:25: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 192.168.1.1
(FastEthernet0/0) is up: new adjacency
RT3#sh ip route

Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 1 subnets
D    172.16.1.0 [90/30720] via 192.168.1.1, 00:00:03, FastEthernet0/0
C     192.168.1.0/24 is directly connected, FastEthernet0/0
RT3#

```

- The alternative way to achieve the filtering of network 10.0.0.0 is deny network 10.0.0.0 and permit other networks. This is an efficient approach if the routing information contained multiple networks but only network 10.0.0.0 needed to be filtered.

```

access-list 1 deny 10.0.0.0 0.255.255.255
access-list 1 permit any

```

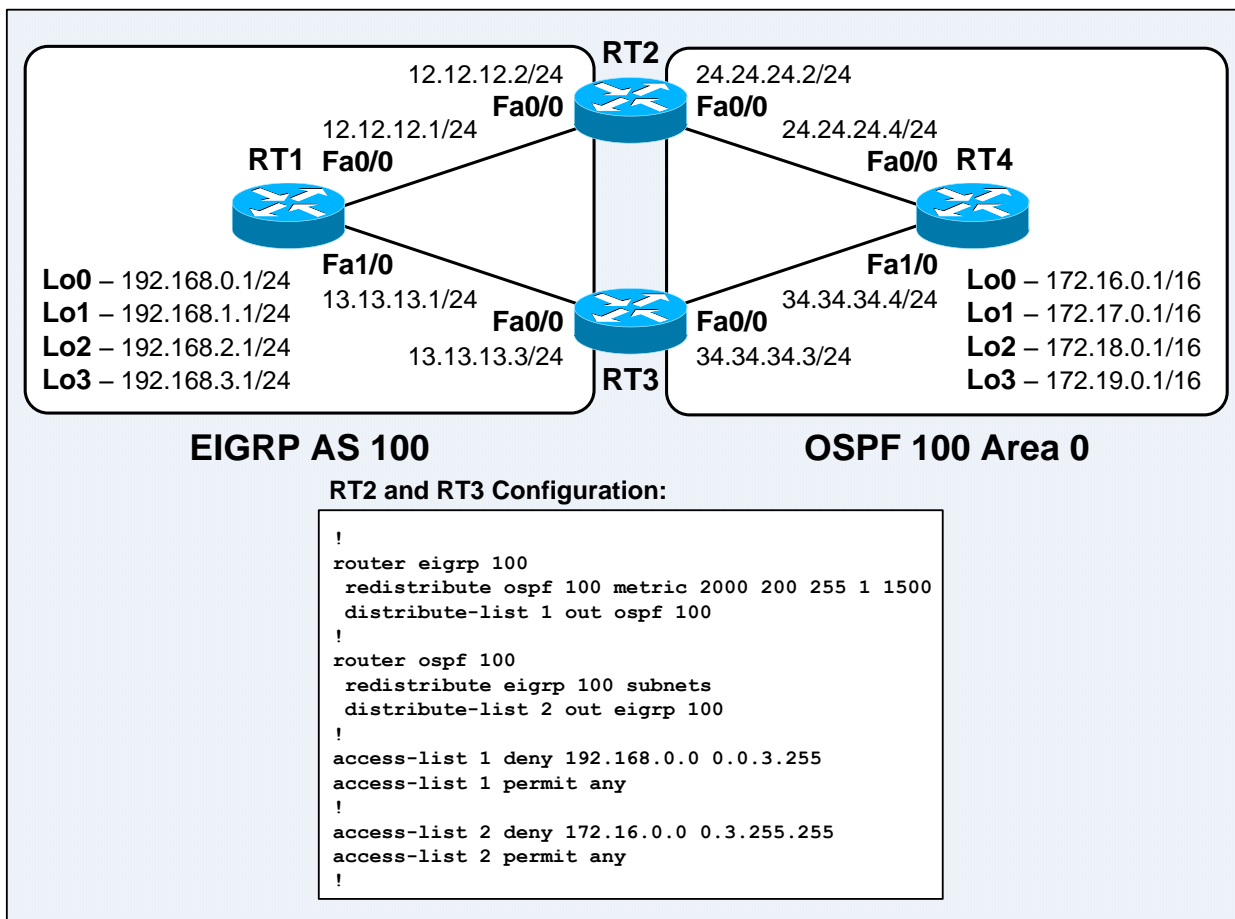


Figure 11-4: Distribute Lists For Two-way / Mutual Redistribution

- The sample network above implemented two-way redistribution between EIGRP and OSPF. Below shows that RT2 will redistribute the OSPF routes 172.16.0.0/22, which redistributed by RT3 from OSPF into EIGRP, back to RT4 when modifying the administrative distance for EIGRP external routes and removing the distribute list for OSPF on RT2:

```

RT2#sh ip route

Gateway of last resort is not set

  12.0.0.0/24 is subnetted, 1 subnets
C    12.12.12.0 is directly connected, FastEthernet0/0
  13.0.0.0/24 is subnetted, 1 subnets
D    13.13.13.0 [90/30720] via 12.12.12.1, 00:02:38, FastEthernet0/0
  24.0.0.0/24 is subnetted, 1 subnets
C    24.24.24.0 is directly connected, FastEthernet1/0
  34.0.0.0/24 is subnetted, 1 subnets
O    34.34.34.0 [110/2] via 24.24.24.4, 00:01:24, FastEthernet1/0
O    172.16.0.0/16 [110/2] via 24.24.24.4, 00:01:24, FastEthernet1/0
O    172.17.0.0/16 [110/2] via 24.24.24.4, 00:01:24, FastEthernet1/0
O    172.18.0.0/16 [110/2] via 24.24.24.4, 00:01:24, FastEthernet1/0
O    172.19.0.0/16 [110/2] via 24.24.24.4, 00:01:24, FastEthernet1/0
D    192.168.0.0/24 [90/156160] via 12.12.12.1, 00:02:38, FastEthernet0/0
D    192.168.1.0/24 [90/156160] via 12.12.12.1, 00:02:38, FastEthernet0/0
D    192.168.2.0/24 [90/156160] via 12.12.12.1, 00:02:38, FastEthernet0/0
D    192.168.3.0/24 [90/156160] via 12.12.12.1, 00:02:38, FastEthernet0/0
RT2#

```

```

RT2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT2(config)#router eigrp 100
RT2(config-router)#distance eigrp 90 80
RT2(config-router)#
00:04:25: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 12.12.12.1
(FastEthernet0/0) is down: route configuration changed
00:04:29: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 12.12.12.1
(FastEthernet0/0) is up: new adjacency
RT2(config-router)#do sh ip route

Gateway of last resort is not set

    12.0.0.0/24 is subnetted, 1 subnets
C       12.12.12.0 is directly connected, FastEthernet0/0
    13.0.0.0/24 is subnetted, 1 subnets
D       13.13.13.0 [90/30720] via 12.12.12.1, 00:00:06, FastEthernet0/0
    24.0.0.0/24 is subnetted, 1 subnets
C       24.24.24.0 is directly connected, FastEthernet1/0
    34.0.0.0/24 is subnetted, 1 subnets
D EX    34.34.34.0 [80/1336320] via 12.12.12.1, 00:00:05, FastEthernet0/0
D EX 172.16.0.0/16 [80/1336320] via 12.12.12.1, 00:00:05, FastEthernet0/0
D EX 172.17.0.0/16 [80/1336320] via 12.12.12.1, 00:00:05, FastEthernet0/0
D EX 172.18.0.0/16 [80/1336320] via 12.12.12.1, 00:00:05, FastEthernet0/0
D EX 172.19.0.0/16 [80/1336320] via 12.12.12.1, 00:00:05, FastEthernet0/0
D       192.168.0.0/24 [90/156160] via 12.12.12.1, 00:00:06, FastEthernet0/0
D       192.168.1.0/24 [90/156160] via 12.12.12.1, 00:00:06, FastEthernet0/0
D       192.168.2.0/24 [90/156160] via 12.12.12.1, 00:00:06, FastEthernet0/0
D       192.168.3.0/24 [90/156160] via 12.12.12.1, 00:00:06, FastEthernet0/0
RT2(config-router)#
RT2(config-router)#do sh access-lists
Standard IP access list 1
    10 deny 192.168.0.0, wildcard bits 0.0.3.255 (4 matches)
    20 permit any (24 matches)
Standard IP access list 2
    10 deny 172.16.0.0, wildcard bits 0.3.255.255 (4 matches)
    20 permit any (19 matches)
RT2(config-router)#
RT2(config-router)#router ospf 100
RT2(config-router)#no distribute-list 2 out eigrp 100
RT2(config-router)#do sh ip ospf database

--- output omitted ---
                                Type-5 AS External Link States

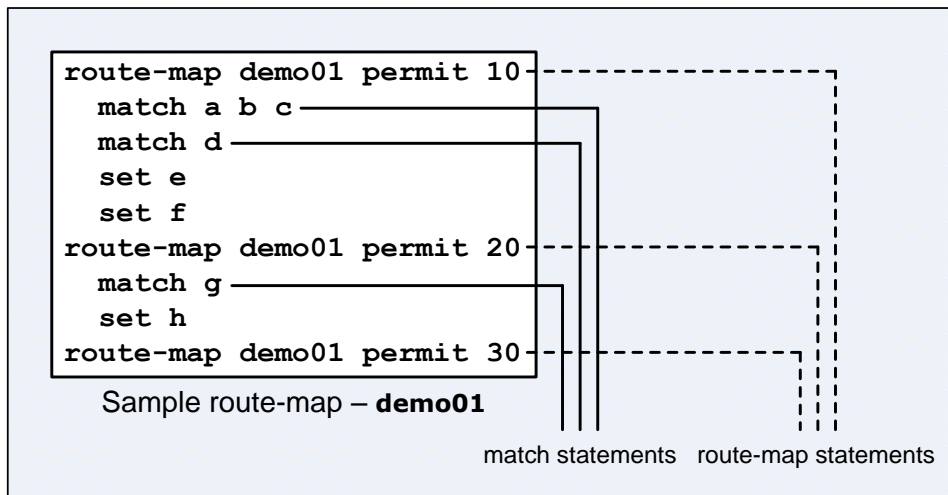
Link ID          ADV Router      Age             Seq#            Checksum Tag
12.12.12.0       2.2.2.2        276            0x80000001     0x00F97A 0
12.12.12.0       3.3.3.3        250            0x80000003     0x00D796 0
13.13.13.0       2.2.2.2        45             0x80000005     0x00CD9F 0
13.13.13.0       3.3.3.3        265            0x80000001     0x00B7B5 0
34.34.34.0       2.2.2.2        50             0x80000001     0x00DE53 0
172.16.0.0       2.2.2.2        3              0x80000001     0x0026B5 0
172.17.0.0       2.2.2.2        3              0x80000001     0x001AC0 0
172.18.0.0       2.2.2.2        3              0x80000001     0x000ECB 0
172.19.0.0       2.2.2.2        3              0x80000001     0x0002D6 0
192.168.0.0     2.2.2.2        45             0x80000005     0x00F139 0
192.168.0.0     3.3.3.3        250            0x80000003     0x00D751 0
192.168.1.0     2.2.2.2        45             0x80000005     0x00E643 0
192.168.1.0     3.3.3.3        250            0x80000003     0x00CC5B 0
192.168.2.0     2.2.2.2        45             0x80000005     0x00DB4D 0
192.168.2.0     3.3.3.3        250            0x80000003     0x00C165 0
192.168.3.0     2.2.2.2        45             0x80000005     0x00D057 0
192.168.3.0     3.3.3.3        250            0x80000003     0x00B66F 0
RT2(config-router)#

```



- Using a single router to redistribute routes means introducing a single point of failure that can cause production issues upon hardware failure. Most redistribution scenarios implement a minimum of 2 routers performing redistribution for redundancy and even for load sharing.
- The existence of multiple redistribution points between 2 routing domains introduces some complex and tricky issues, in which a route from a routing domain can be redistributed into another routing domain, and then being redistributed back into the original routing domain. **Domain loop** occurs when the twice-redistributed route is redistributed back into the original routing domain with a relatively low metric and being preferred over the route that was advertised only internal to that routing domain. Configuring higher metrics upon redistributed routes is often used to prevent domain loop.
- Interestingly, EIGRP and OSPF with default settings is not prone to domain loop problems when either one of them is one of the routing protocols that undergo a two-way mutual redistribution. The default EIGRP administrative distances values (90 for internal; 170 for external) defeats the domain loop problem when redistributing between EIGRP and OSPF. OSPF always prefers internal routes over E1 routes, and E1 routes over E2 routes, before even considering the metrics.
- Distribute lists hides network information, which can be considered a drawback in some setups. Ex: In a network with redundant paths, a distribute list might permit routing updates for only specific routes to avoid routing loops. Other routers might not know about other paths to reach the filtered networks. So when the primary path goes down, the backup paths are not used, as other routers do not know they exist. When redundant paths exist, other techniques, eg: manipulating the administrative distance or metric, should be used instead of distribute lists, to enable the use of an alternative path (with a worse administrative distance or metric) when the primary path goes down.
- Cisco recommended using route maps to manipulate and control routing updates. All IP routing protocols can use route maps for redistribution filtering.
- Route maps are complex ACLs that use **match** commands to test some conditions upon interesting packets or routes. Once the conditions are matched, the actions specified by **set** commands will be taken to modify the attributes of the packet or routes.
- A route map is a collection of route map statements that have the same route map name. Within a route map, each route map statement is numbered and can be edited individually. Like an access list, there is an implicit **deny any** at the end of a route map. The consequences of this deny depend upon the usage of the route map.
- The **route-map** {*map-tag*} [**permit** | **deny**] [*seq-num*] global configuration command can be used to define the conditions for processing. The *map-tag* is the name of the route map. The **permit** and **deny** are optional parameters that specify the action to be taken when a route map match conditions are met. The optional sequence number indicates the position for a new route map statement in an already existed route map (used for inserting or deleting specific route map statements in a route map).
- A route map referenced by the **redistribute** router subcommand always attempts to filter routes. If a particular route-map statement with the **permit** action matches a particular route, the route is redistributed as controlled by the **set** actions; for policy routing, the packet is policy routed. If a particular route-map statement with the **deny** action matches a particular route, the route is filtered – not redistributed; for policy routing, the packet is not policy routed.

- A single **match** statement may contain multiple conditions; just a **single** condition needs to be true for the **match** statement to be considered matched. (Logical OR)  
A single route map statement may contain multiple **match** statements; **all match** statements in the route map statement must be true for the route map statement to be considered matched.  
**Multiple match conditions → A match statement / clause.** (Logical AND)  
**Multiple match statements / clauses → A route map statement.**  
**Multiple route map statements → A route map.**



**Figure 11-5: Route Map Interpretation**

- The sample route map named **demo01** in Figure 11-5 is interpreted as:

```

if ((a or b or c) and d)
  set e and f
else if (g)
  set h
else
  set nothing

```
- **Note:** The default action for the **route-map** command is **permit**, with sequence number of 10. The actions defined with the **set {condition}** route map configuration command will be effective only when the action of the route map is **permit**.  
**Note:** Do not leave out the *seq-num* when editing and adding statements in a route map list, or else only the 1st statement with the sequence number of 10 will always be referred to. Route map sequence numbers **do not automatically increment** as with ACL configuration!
- Route maps are being used for a variety of purposes. Several common usages of route maps are:
  - **Route filtering during redistribution.** Redistribution often requires route filtering. Although distribute lists can be used for this purpose, route maps offer greater flexibilities for matching and manipulating routing updates using **match** criteria and **set** actions.
  - **Policy-Based Routing (PBR).** Route maps are able to match source and destination addresses, protocol types, and end-user applications through transport layer port numbers. When a match occurs, a **set** action can be used to define the interface or next-hop address to which the packet should be forwarded. PBR provides an ability to define routing policy rather than rely upon the routing table for basic destination-based routing.
  - **NAT.** Route maps provides better control upon defining the NAT addresses as well as detailed **show** commands that available to monitor the address-translation process.
  - **BGP.** Route maps are the primary tools used for implementing BGP routing policies. Network administrators assign route maps to specific BGP sessions / neighbors to control which routes are allowed to flow in and out of the BGP process. In addition to filtering, route maps also provide sophisticated manipulation upon BGP path attributes.

- Route maps use the **match** subcommand to identify routes. The **match** command can refer to ACLs and prefix-lists to match anything matchable by them. Below lists the **match** commands that matter when using route maps for redistribution.

<b>match interface</b> { <i>intf-type intf-num</i> } [... <i>intf-type intf-num</i> ]	Matches routes that outgoing from one of the specified interfaces.
<b>match ip address</b> {[ <i>acl-num</i>   <i>acl-name</i> ]   <b>prefix-list</b> { <i>prefix-name</i> }}	Matches routes that matched by the access list or prefix list. [*]
<b>match ip next-hop</b> {[ <i>acl-num</i>   <i>acl-name</i> ]   <b>prefix-list</b> { <i>prefix-name</i> }}	Matches routes that have the next-hop address matched by the access list or prefix list. [*]
<b>match ip route-source</b> {[ <i>acl-num</i>   <i>acl-name</i> ]   <b>prefix-list</b> { <i>prefix-name</i> }}	Matches routes that advertised by the IP address (router) that matched by the access list or prefix list. [*]
<b>match metric</b> { <i>metric</i> } [... <i>metric</i> ]	Matches routes with the specified metrics.
<b>match route-type</b> { <b>internal</b>   <b>external</b> [ <b>type-1</b>   <b>type-2</b> ]   <b>level-1</b>   <b>level-2</b>   <b>local</b>   <b>nssa-external</b> }	Matches routes with the specified EIGRP, OSPF, IS-IS, and BGP route types.
<b>match tag</b> { <i>tag-value</i> } [... <i>tag-value</i> ]	Matches the route tag that set by another router.

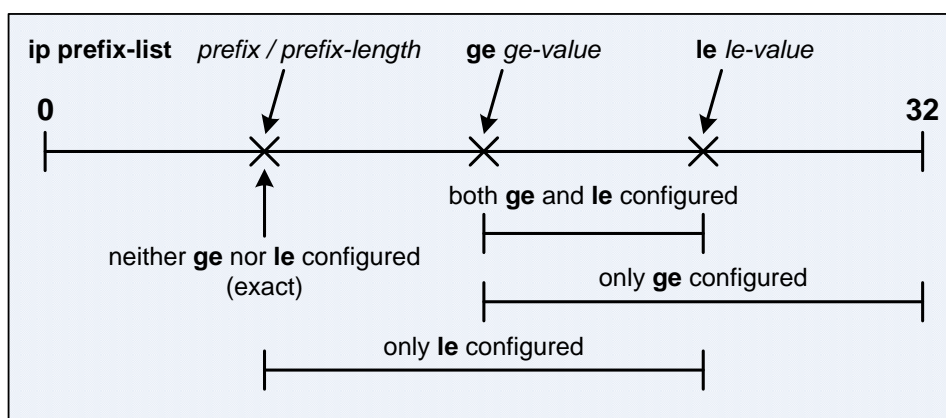
[\*] – Multiple access lists or prefix lists can be associated with a single **match** action.

- Below lists the **set** actions that matter when using route maps for redistribution.

<b>set metric</b> { <i>metric-value</i>   <i>bandwidth</i> <i>delay</i> <i>reliability</i> <i>loading</i> <i>mtu</i> }	Sets the metric for RIP, OSPF, IS-IS, and EIGRP routes.
<b>set metric-type</b> { <b>type-1</b>   <b>type-2</b>   <b>internal</b>   <b>external</b> }	Sets the type (E1 or E2) for OSPF external routes and IS-IS routes.
<b>set tag</b> { <i>tag-value</i> }	Sets the tag value for the redistributed routes.

- **Prefix lists** are used to match IP prefixes, with the capability to match an exact prefix length or a prefix range. Prefix lists are often used as the alternative over access lists and distribute lists. Prefix lists are faster and less CPU-intensive than regular access lists and distribute lists. Prefix list entries can be deleted and added individually.
- The formats of a prefix list entry and an IP access control list (ACL) entry are similar. A prefix list entry consists of a name, an action (deny or permit), the prefix number, and the prefix length. The syntax of the command is **ip prefix-list** {*list-name*} [**seq** *seq-num*] {**deny** | **permit**} {*prefix/length*} [**ge** *ge-value*] [**le** *le-value*]. The network number can be any valid IP address or prefix, while the bit mask can be a number from 0 to 32. The prefix is automatically converted to match the prefix length value, eg: entering 10.11.12.0/8 would result in 10.0.0.0/8.  
**Note:** If a prefix is permitted, the route will be used; if a prefix is denied, the route is not used.
- The basic form of prefix list assumes an exact match of both prefix number and prefix length. Additional parameters are required to match a range of prefixes. When a prefix range ends at /32, the *ge-value* (greater-than-or-equal-to) can be specified. The *ge-value* must be greater than the length specified by the *prefix/length* parameter, and less than 32. When the **ge** parameter is specified, the prefixes with mask length from the *ge-value* to 32 (inclusive) will be matched.
- If the prefix length does not end at /32, the **le** (less-than-or-equal-to) parameter must be specified. When both the **ge** and **le** parameters are specified, the prefixes with mask length between the *ge-value* and *le-value* (inclusive) will be matched. The specified *ge-value* and *le-value* must satisfy the following condition:

$$prefix-length < ge-value < le-value \leq 32$$



**Figure 11-6:** Representation of Prefix Length Ranges for the **ip prefix-list** Command

- Below is an example of using both the **ge** and **le** parameters to match a portion of 172.16.1.0/24:  

```
ip prefix-list pl-test permit 172.16.1.0/24 ge 25 le 30
```

Note that 172.16.1.0/24 and all the /31s and /32s are not in the range.

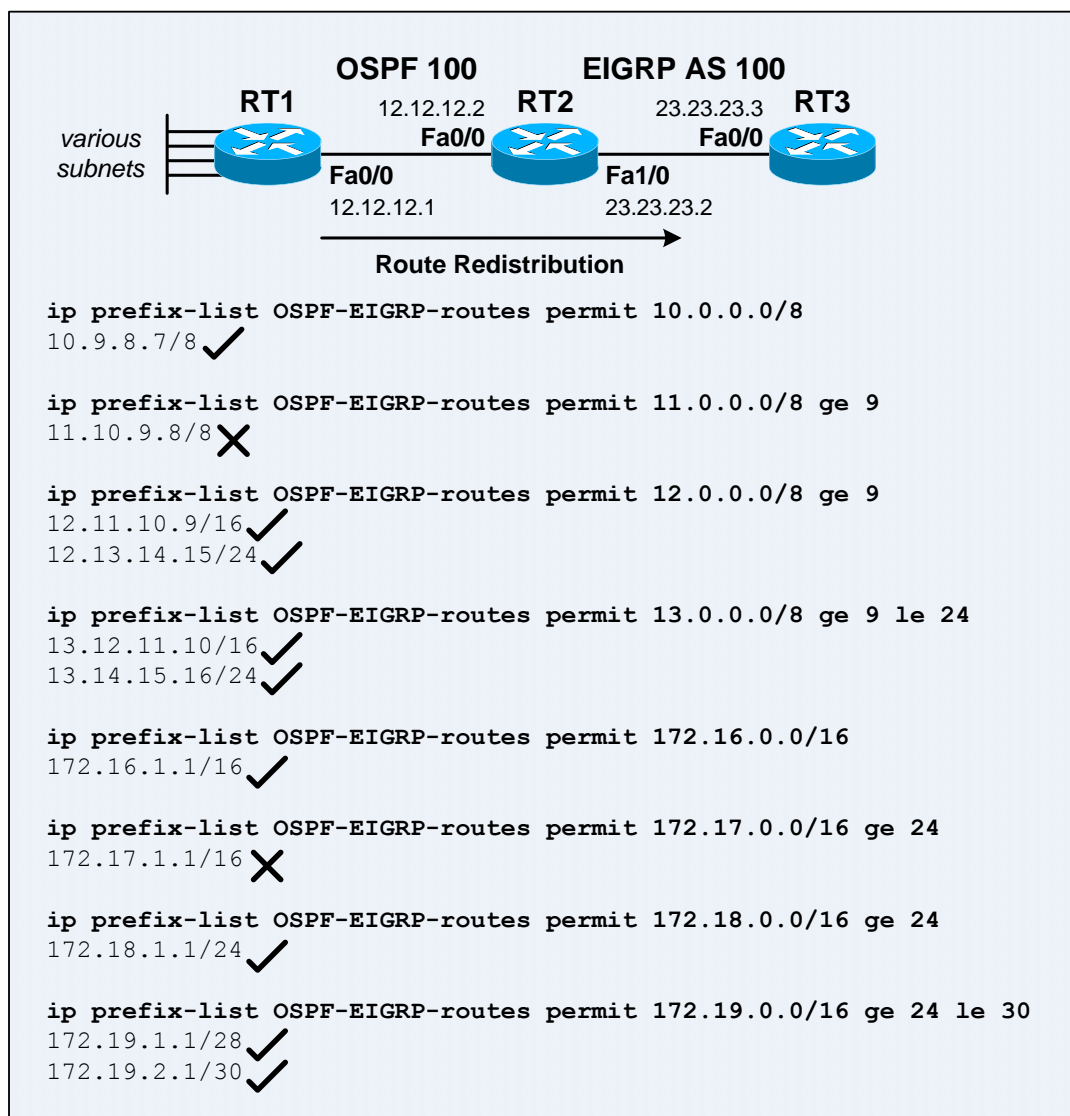
Below lists the prefixes that are being matched by the prefix range:

2 /25s	172.16.1.0/25, 172.16.1.128/25.
4 /26s	172.16.1.0/26, 172.16.1.64/26, 172.16.1.128/26, 172.16.1.192/26.
8 /27s	172.16.1.0/27, 172.16.1.32/27 ... 172.16.1.192/27, 172.16.1.224/27.
16 /28s	172.16.1.0/28, 172.16.1.16/28 ... 172.16.1.224/28, 172.16.1.240/28.
32 /29s	172.16.1.0/29, 172.16.1.8/29 ... 172.16.1.240/29, 172.16.1.248/29.
64 /30s	172.16.1.0/30, 172.16.1.4/30 ... 172.16.1.248/30, 172.16.1.252/30.

- When a prefix list is configured without a sequence number, the default sequence number of 5 will be applied to the prefix list, and subsequent prefix list entries will be incremented by 5, eg: 5, 10, 15, etc. If a sequence number is entered for the first prefix list entry but not subsequent entries, the subsequent entries will also be incremented by 5, eg: if the first configured sequence number is 3, then the subsequent sequence numbers will be 8, 13, 18, etc.

- Below lists some examples of prefix lists:

<pre>ip prefix-list pl-test permit 0.0.0.0/0</pre>	A prefix list entry configured to match only the default route 0.0.0.0/0.
<pre>ip prefix-list pl-test permit 0.0.0.0/0 le 32</pre>	A prefix list entry configured to match any address or subnet – match all (permit any any).
<pre>ip prefix-list pl-test permit 0.0.0.0/0 ge 8 le 24</pre>	A prefix list entry configured to match any prefix that has a prefix length from 8 to 24 bits.
<pre>ip prefix-list pl-test permit 0.0.0.0/0 ge 30 le 30</pre>	A prefix list entry configured to match any prefix with prefix length of 30.
<pre>ip prefix-list pl-test permit 172.16.1.0/24</pre>	A prefix list entry configured to match the 172.16.1.0/24 subnet.
<pre>ip prefix-list pl-test permit 10.0.0.0/8 le 24</pre>	A prefix list entry configured to match subnets from the 10.0.0.0/8 network that have a prefix length that is less than or equal to 24 bits.
<pre>ip prefix-list pl-test permit 10.0.0.0/8 ge 25</pre>	A prefix list entry configured to match subnets from the 10.0.0.0/8 network that have a prefix length that is greater than or equal to 25 bits.
<pre>ip prefix-list pl-test permit 10.0.0.0/8 le 32</pre>	A prefix list entry configured to match any prefix from the 10.0.0.0/8 network.



**Figure 11-7: Network Setup for IP Prefix Lists**

- The sample network above was setup to observe how RT2 uses prefix lists to determine which subnets to be redistributed from OSPF into EIGRP.
- Below shows the configuration on RT2:

```

!
router ospf 100
 network 12.12.12.2 0.0.0.0 area 0
!
router eigrp 100
 redistribute ospf 100 route-map OSPF-EIGRP
 network 23.23.23.2 0.0.0.0
 default-metric 10000 100 255 1 1500
 no auto-summary
!
ip prefix-list OSPF-EIGRP-routes seq 5 permit 10.0.0.0/8
ip prefix-list OSPF-EIGRP-routes seq 10 permit 11.0.0.0/8 ge 9
ip prefix-list OSPF-EIGRP-routes seq 15 permit 12.0.0.0/8 ge 9
ip prefix-list OSPF-EIGRP-routes seq 20 permit 13.0.0.0/8 ge 9 le 24
ip prefix-list OSPF-EIGRP-routes seq 25 permit 172.16.0.0/16
ip prefix-list OSPF-EIGRP-routes seq 30 permit 172.17.0.0/16 ge 24
ip prefix-list OSPF-EIGRP-routes seq 35 permit 172.18.0.0/16 ge 24
ip prefix-list OSPF-EIGRP-routes seq 40 permit 172.19.0.0/16 ge 24 le 30
!
route-map OSPF-EIGRP permit 10
 match ip address prefix-list OSPF-EIGRP-routes
!

```

- Below shows the routing table on RT3:

```

RT3#sh ip route

Gateway of last resort is not set

    23.0.0.0/24 is subnetted, 1 subnets
C       23.23.23.0 is directly connected, FastEthernet0/0
D EX   172.16.0.0/16 [170/284160] via 23.23.23.2, 00:01:29, FastEthernet0/0
       172.19.0.0/16 is variably subnetted, 2 subnets, 2 masks
D EX   172.19.2.0/30 [170/284160] via 23.23.23.2, 00:00:04, FastEthernet0/0
D EX   172.19.1.0/28 [170/284160] via 23.23.23.2, 00:00:04, FastEthernet0/0
       172.18.0.0/24 is subnetted, 1 subnets
D EX   172.18.1.0 [170/284160] via 23.23.23.2, 00:00:24, FastEthernet0/0
D EX   10.0.0.0/8 [170/284160] via 23.23.23.2, 00:04:52, FastEthernet0/0
       12.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
D EX   12.11.0.0/16 [170/284160] via 23.23.23.2, 00:03:06, FastEthernet0/0
D EX   12.12.12.0/24 [170/284160] via 23.23.23.2, 00:03:06, FastEthernet0/0
D EX   12.13.14.0/24 [170/284160] via 23.23.23.2, 00:03:06, FastEthernet0/0
       13.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
D EX   13.12.0.0/16 [170/284160] via 23.23.23.2, 00:02:35, FastEthernet0/0
D EX   13.14.15.0/24 [170/284160] via 23.23.23.2, 00:02:35, FastEthernet0/0
RT3#

```

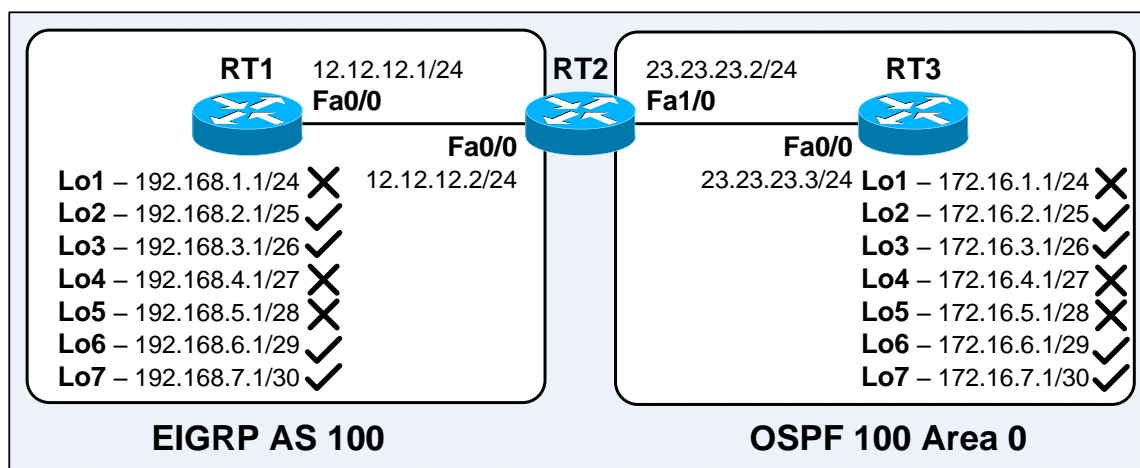


Figure 11-8: Route Filtering using Route Maps

- The sample network above demonstrates the flexibility of filtering redistributed routes using route maps. Only certain prefixes are being redistributed from EIGRP to OSPF, and vice versa.
- The route maps simply need to have route map statements with **deny** and **permit** actions to match the routes to be filtered and not to be filtered correspondingly. There are 2 different approaches to perform the mentioned task:
  - **Approach #1:** Begin with a match of the routes to be filtered using extended IP ACLs or IP prefix lists, with a **deny** action for the routes to be filtered. Followed by a **permit** statement with no **match** command at all, matching and allowing all remaining routes.
  - **Approach #2:** Begin with a match of routes not to be allowed using extended IP ACLs or IP prefix lists, with a **permit** action for the routes to be allowed. Followed by using the implicit **deny all** at the end of the route map to filter unwanted routes.

- Below shows the configuration on RT2. It uses approach #1 to filter routes from EIGRP to OSPF, and approach #2 to filter routes from OSPF to EIGRP.

```

! Filtering redistributed routes from EIGRP to OSPF (Approach #1):
!
ip access-list extended match-192.168.1.0_24
  permit ip host 192.168.1.0 host 255.255.255.0
!
ip access-list extended match-192.168.4.0_27*192.168.5.0_28
  permit ip host 192.168.4.0 host 255.255.255.224
  permit ip host 192.168.5.0 host 255.255.255.240
!
route-map redist-eigrp*ospf deny 10
  match ip address match-192.168.1.0_24
!
route-map redist-eigrp*ospf deny 20
  match ip address match-192.168.4.0_27*192.168.5.0_28
!
route-map redist-eigrp*ospf permit 100
!
router ospf 100
  redistribute eigrp 100 subnets route-map redist-eigrp*ospf
!
! =====
! Filtering redistributed routes from OSPF to EIGRP (Approach #2):
!
ip prefix-list match-ospf-routes seq 5 permit 172.16.2.0/23 ge 25 le 26
ip prefix-list match-ospf-routes seq 10 permit 172.16.6.0/23 ge 29 le 30
!
route-map redist-ospf*eigrp permit 10
  match ip address prefix-list match-ospf-routes
!
router eigrp 100
  redistribute ospf 100 metric 2000 200 255 1 1500 route-map redist-ospf*eigrp
!

```

- Below shows 2 alternative configurations for Approach #1 to filter routes from EIGRP to OSPF.

```

! Approach #1 - Alternative #1:
!
ip access-list extended match-192.168.1.0_24
  permit ip 192.168.1.0 0.0.0.255 host 255.255.255.0
!
ip access-list extended match-192.168.4.0_27*192.168.5.0_28
  permit ip 192.168.4.0 0.0.0.31 host 255.255.255.224
  permit ip 192.168.5.0 0.0.0.15 host 255.255.255.240
!
route-map redist-eigrp*ospf deny 10
  match ip address match-192.168.1.0_24 match-192.168.4.0_27*192.168.5.0_28
!
route-map redist-eigrp*ospf permit 100
!
router ospf 100
  redistribute eigrp 100 subnets route-map redist-eigrp*ospf
!

```

**! Approach #1 - Alternative #2:**

```
!  
ip prefix-list match-eigrp-routes seq 5 permit 192.168.1.0/24  
ip prefix-list match-eigrp-routes seq 10 permit 192.168.4.0/23 ge 27 le 28  
!  
route-map redist-eigrp*ospf deny 10  
  match ip address prefix-list match-eigrp-routes  
!  
route-map redist-eigrp*ospf permit 100  
!  
router ospf 100  
  redistribute eigrp 100 subnets route-map redist-eigrp*ospf  
!
```

- Routing tables on RT1 and RT3 after implemented the redistribution configuration on RT2:

**RT1#sh ip route**

Gateway of last resort is not set

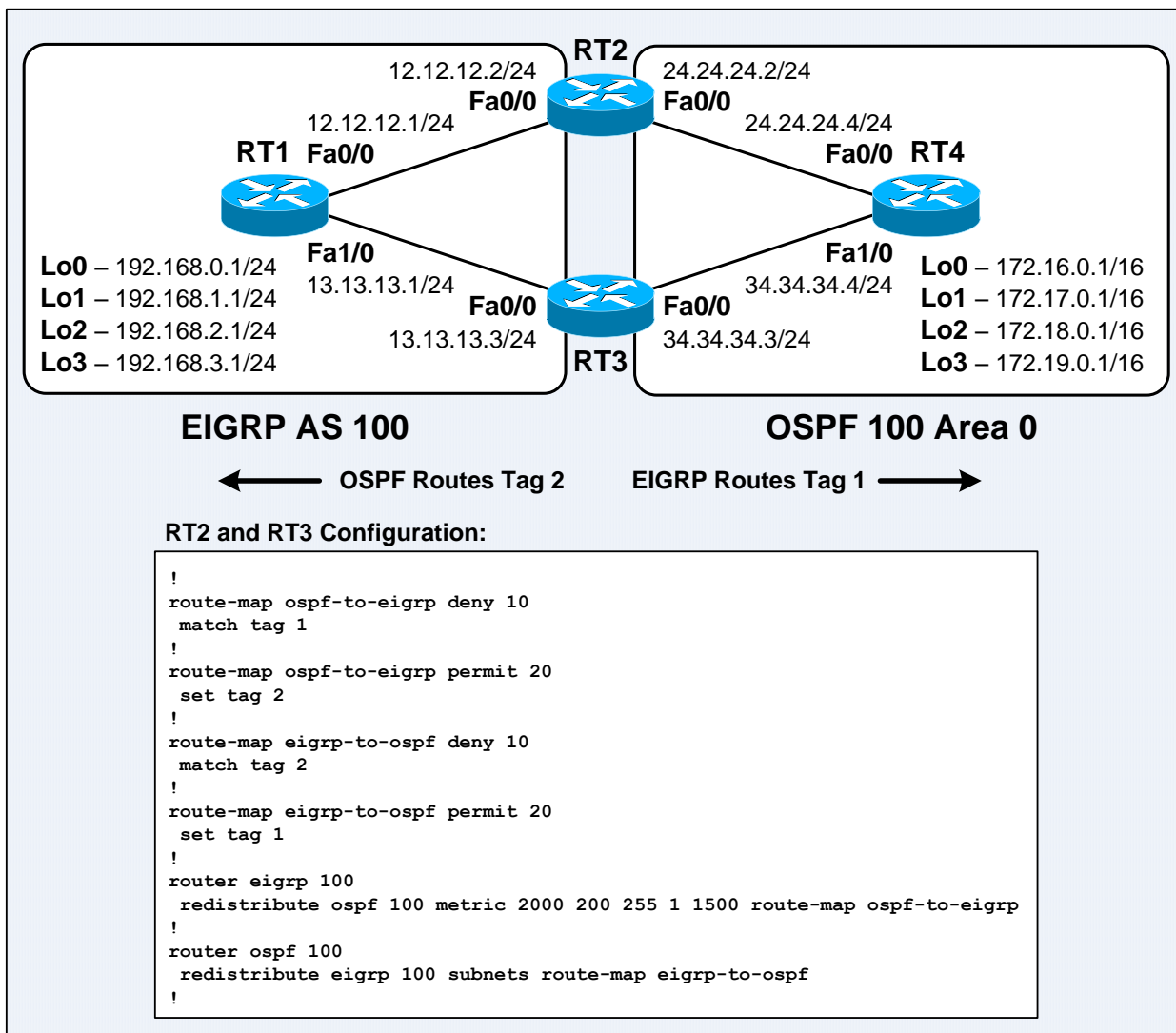
```
    12.0.0.0/24 is subnetted, 1 subnets  
C       12.12.12.0 is directly connected, FastEthernet0/0  
    172.16.0.0/16 is variably subnetted, 4 subnets, 4 masks  
D EX    172.16.2.0/25 [170/1333760] via 12.12.12.2, 00:00:30, FastEthernet0/0  
D EX    172.16.3.0/26 [170/1333760] via 12.12.12.2, 00:00:30, FastEthernet0/0  
D EX    172.16.6.0/29 [170/1333760] via 12.12.12.2, 00:00:30, FastEthernet0/0  
D EX    172.16.7.0/30 [170/1333760] via 12.12.12.2, 00:00:30, FastEthernet0/0  
C       192.168.1.0/24 is directly connected, Loopback1  
    192.168.2.0/25 is subnetted, 1 subnets  
C       192.168.2.0 is directly connected, Loopback2  
    192.168.3.0/26 is subnetted, 1 subnets  
C       192.168.3.0 is directly connected, Loopback3RT1#  
    192.168.4.0/27 is subnetted, 1 subnets  
C       192.168.4.0 is directly connected, Loopback4  
    192.168.5.0/28 is subnetted, 1 subnets  
C       192.168.5.0 is directly connected, Loopback5  
    192.168.6.0/29 is subnetted, 1 subnets  
C       192.168.6.0 is directly connected, Loopback6  
    192.168.7.0/30 is subnetted, 1 subnets  
C       192.168.7.0 is directly connected, Loopback7  
=====
```

**RT3#sh ip route**

Gateway of last resort is not set

```
    12.0.0.0/24 is subnetted, 1 subnets  
O E2    12.12.12.0 [110/20] via 23.23.23.2, 00:00:35, FastEthernet0/0  
    23.0.0.0/24 is subnetted, 1 subnets  
C       23.23.23.0 is directly connected, FastEthernet0/0  
    172.16.0.0/16 is variably subnetted, 7 subnets, 7 masks  
C       172.16.1.0/24 is directly connected, Loopback1  
C       172.16.2.0/25 is directly connected, Loopback2  
C       172.16.3.0/26 is directly connected, Loopback3  
C       172.16.4.0/27 is directly connected, Loopback4  
C       172.16.5.0/28 is directly connected, Loopback5  
C       172.16.6.0/29 is directly connected, Loopback6  
C       172.16.7.0/30 is directly connected, Loopback7  
    192.168.2.0/25 is subnetted, 1 subnets  
O E2    192.168.2.0 [110/20] via 23.23.23.2, 00:00:35, FastEthernet0/0  
    192.168.3.0/26 is subnetted, 1 subnets  
O E2    192.168.3.0 [110/20] via 23.23.23.2, 00:00:35, FastEthernet0/0  
    192.168.6.0/29 is subnetted, 1 subnets  
O E2    192.168.6.0 [110/20] via 23.23.23.2, 00:00:35, FastEthernet0/0  
    192.168.7.0/30 is subnetted, 1 subnets  
O E2    192.168.7.0 [110/20] via 23.23.23.2, 00:00:35, FastEthernet0/0
```

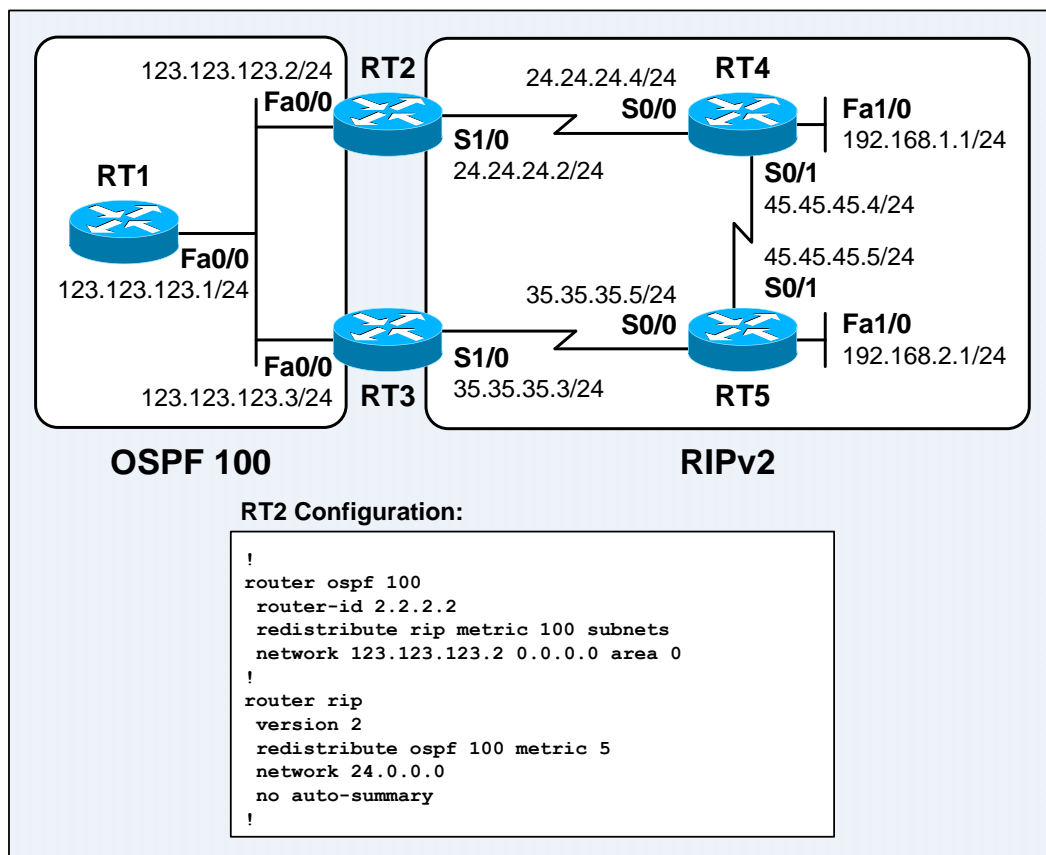




**Figure 11-9: Route Tagging For Two-way / Mutual Redistribution**

- The network setup above is revisited with implement route tagging upon mutual redistribution. The EIGRP and OSPF routing processes on RT2 and RT3 can then perform route filtering upon mutual redistribution.
- A route tag follows the route advertisement, even through another redistribution process. Other route maps can match routes with a route tag to make a route filtering decision.
- With the configuration above, RT2 and RT3 are able to identify OSPF and EIGRP external routes with tags of 1 and 2 respectively. RT2 and RT3 will filter OSPF and EIGRP external routes that advertised back into the original routing domains using route map statements with **deny** action.

- Route selection is sometimes confusing due to route redistribution. Controlling administrative distance is an important and effective method to indicate the preference upon route selection. Changing the default administrative distance values only after careful planning and considered the specific requirements upon the network design and setup.
- The **distance** {*ad-weight*} [*adv-router wildcard-mask* [*acl-num* | *acl-name*]] [**ip**] router subcommand defines administrative distances for all routing protocols except EIGRP and BGP. The optional *adv-router wildcard-mask* pair matches routes according to the IP address(es) of the advertising router(s) that supply the routing information. Uses the address / mask of 0.0.0.0 255.255.255.255 to match any advertising router supplying the routing information. An optional access list can also be referenced to match the specific routes from any matched neighbors to use the specified administrative distance – **Prefix-Based Administrative Distance**. **Note:** The **ip** keyword specifies IP-derived routes for Integrated IS-IS.
- The **distance eigrp** {*internal-distance external-distance*} EIGRP router subcommand defines the administrative distances for EIGRP internal and external routes respectively.
- The **distance ospf** {**external** *dist1* | **inter-area** *dist2* | **intra-area** *dist3*} OSPF router subcommand defines the administrative distances of OSPF routes based on the route type. This command performs the same function as the **distance** command used with an access list. However, this command provides the capability to set an administrative distance for an entire group of routes, rather than specific routes that matched by an access list. A common usage of this command is when implementing OSPF processes with mutual redistribution, which is often required to prefer internal routes from a process over external routes from another process.



**Figure 11-10:** Route Filtering using Administrative Distance

- The routes are being redistributed with metric values higher than the native metrics for routes in both routing domains in order to protect against suboptimal routing.

- Below shows that suboptimal routing occurred on RT3 due to the RIPv2 routes redistributed into OSPF as E2 routes have a lower administrative distance and being preferred over the RIPv2 routes.

```

RT3#sh ip route

Gateway of last resort is not set

    35.0.0.0/24 is subnetted, 1 subnets
C       35.35.35.0 is directly connected, Serial1/0
    24.0.0.0/24 is subnetted, 1 subnets
O E2   24.24.24.0 [110/100] via 123.123.123.2, 00:00:10, FastEthernet0/0
    123.0.0.0/24 is subnetted, 1 subnets
C       123.123.123.0 is directly connected, FastEthernet0/0
O E2   192.168.1.0/24 [110/100] via 123.123.123.2, 00:00:10, FastEthernet0/0
O E2   192.168.2.0/24 [110/100] via 123.123.123.2, 00:00:10, FastEthernet0/0
    45.0.0.0/24 is subnetted, 1 subnets
O E2   45.45.45.0 [110/100] via 123.123.123.2, 00:00:10, FastEthernet0/0
RT3#

```

**Note:** OSPF has an administrative distance of 110; RIPv2 has an administrative distance of 120.

- The redistribution configuration on RT2 has resulted in suboptimal routing to many destinations. RT3 takes the longer (worse) OSPF paths than the more direct RIPv2 paths to those networks.
- Below implements the solution on RT3 by changing the administrative distance for redistributed RIPv2 routes (OSPF external routes) advertised by RT2. When RT3 learn about the networks that matched by the access list from both RIPv2 and OSPF, it selects the routes learned from RIPv2:

```

RT3#debug ip routing
IP routing debugging is on
RT3#
RT3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT3(config)#access-list 1 permit 24.24.24.0
RT3(config)#access-list 1 permit 45.45.45.0
RT3(config)#access-list 1 permit 192.168.1.0
RT3(config)#access-list 1 permit 192.168.2.0
RT3(config)#
RT3(config)#router ospf 100
RT3(config-router)#distance 125 0.0.0.0 255.255.255.255 1
RT3(config-router)#end
RT3#
00:03:56: RT: closer admin distance for 24.24.24.0, flushing 1 routes
00:03:56: RT: add 24.24.24.0/24 via 35.35.35.5, rip metric [120/2]
00:03:56: RT: closer admin distance for 45.45.45.0, flushing 1 routes
00:03:56: RT: add 45.45.45.0/24 via 35.35.35.5, rip metric [120/1]
00:03:56: RT: closer admin distance for 192.168.1.0, flushing 1 routes
00:03:56: RT: add 192.168.1.0/24 via 35.35.35.5, rip metric [120/2]
00:03:56: RT: closer admin distance for 192.168.2.0, flushing 1 routes
00:03:56: RT: add 192.168.2.0/24 via 35.35.35.5, rip metric [120/1]
RT3#
RT3#sh access-list
Standard IP access list 1
    permit 24.24.24.0 (1 match)
    permit 45.45.45.0 (1 match)
    permit 192.168.1.0 (1 match)
    permit 192.168.2.0 (1 match)
RT3#

```

```
RT3#sh ip route
```

```
Gateway of last resort is not set
```

```
    35.0.0.0/24 is subnetted, 1 subnets
C       35.35.35.0 is directly connected, Serial1/0
    24.0.0.0/24 is subnetted, 1 subnets
R       24.24.24.0 [120/2] via 35.35.35.5, 00:00:12, Serial1/0
    123.0.0.0/24 is subnetted, 1 subnets
C       123.123.123.0 is directly connected, FastEthernet0/0
R       192.168.1.0/24 [120/2] via 35.35.35.5, 00:00:12, Serial1/0
R       192.168.2.0/24 [120/1] via 35.35.35.5, 00:00:12, Serial1/0
    45.0.0.0/24 is subnetted, 1 subnets
R       45.45.45.0 [120/1] via 35.35.35.5, 00:00:12, Serial1/0
RT3#
```

- Basically RT3 assigns an administrative distance of 125 upon redistributed routes that matched by access list 1. Note that the **distance** command is implemented under the OSPF process, as the administrative distance should be changed for routes that learned via OSPF, not RIPv2.
- The main advantage of using administrative distance to control route preference is that no path information is lost – the OSPF information still resides in the OSPF LSDB. When the primary path to RIPv2 networks (the RT3 – RT5 link) fails, the OSPF routes reasserts themselves, and RT3 resumes connectivity with those RIPv2 networks through RT2.
- As a conclusion, it is important to know the network design and setup inside out and thoroughly prior to implementing redistribution, and closely monitors the redistributed routes, particularly on networks with redundant paths, as routers are more likely to select suboptimal paths.

*This page is intentionally left blank*

## Policy-Based Routing and IP SLA (Service-Level Agreement)

### Policy-Based Routing

- **Policy-Based Routing** (PBR) provides organizations the flexibility to implement policies for routing and forwarding packets that go beyond the traditional destination-based routing, eg: forwarding packets through to different physical network paths based on source and destination addresses, protocol types, user applications, and the size of packets. PBR is a form of static routing that uses route maps to create a separate process that control packet forwarding in a sophisticated way by matching and changing the attributes of packets.
- Below lists other general usages of PBR:
  - **Source-based transit provider selection** – Organizations can use PBR to route traffic originating from different sets of users through different Internet connections.
  - **Quality of Service (QoS)** – Organizations can provide QoS upon differentiated traffic by setting Type of Service (ToS) or IP Precedence values in the IP packet header on edge routers and use queuing mechanisms to prioritize traffic in the network core or backbone.
  - **Cost savings.** Organizations can temporary direct the bulk traffic associated with specific application to use a higher-bandwidth, high-cost link for a short period of time and resume basic connectivity over a lower-bandwidth, low-cost link for interactive traffic.
  - **Load sharing.** Organizations can distribute traffic among multiple paths based on the characteristics of the applications in addition to the dynamic load-sharing capabilities offered by destination-based routing that the Cisco IOS software has always supported.
- PBRs are applied upon **incoming** packets. Enabling PBR causes the router to evaluate all packets arriving on the interface configured with a route map for that purpose. Packets are being processed by the data plane and forwarded to the appropriate next-hop based on the criteria defined in the route map; and therefore override the normal packet forwarding of the router.
- Most Cisco routers today use **Cisco Express Forwarding (CEF)** by default, which causes them to match the destination address of IP packets with the CEF table instead of the IP routing table. Cisco IOS derives the CEF table from the information in the IP routing table, with much faster entry lookup as compared with using the IP routing table.
- Implementing PBR is basically configures a route map and applies the route map to an interface. The PBR route map statements can be configured with **permit** or **deny** actions.
  - If the action of the statement is **permit**, a packet that meets the match criteria is being policy routed according to the **set** actions.
  - If the action of the statement is **deny**, a packet that meets the match criteria is not being policy routed and is forwarded through the normal destination-based routing.
  - If a packet is not matched by any route map statement, it is not dropped implicit **deny** statement; it is being forwarded through the normal destination-based routing.
  - If do not want to revert to the normal destination-based routing but instead want to drop all packets that are not being matched by any criteria, configure the **set interface Null0** action as the last entry of the route map.
  - If a packet meets the match criteria of a **permit** statement with no **set** action, it is being forwarded through the normal destination-based routing.
- Policy routing does not change the destination of the traffic; it affects only the next hop to which traffic is directed prior to arrive upon the final destination.

- Below lists the **match** commands that matter when using route maps for PBR:

<b>match ip address</b> {[acl-num   acl-name]   <b>prefix-list</b> {ip-prefix-name}}	Matches incoming packets that matched by the access list or prefix list. If multiple access lists or prefix lists are specified, matching any one will result in a match and being policy routed. Standard IP ACLs are used to specify match criteria based on source address; while extended IP ACLs are used to specify match criteria based on source and destination addresses, protocol types, applications, ToS, and IP Precedence.
<b>match length</b> {min max}	Specifies match criteria based on the packet length between the <i>min</i> and <i>max</i> values (inclusive) in bytes. The <i>min</i> and <i>max</i> values are from 0 to 2147483647 (inclusive). Time-sensitive interactive traffic often uses small packets. They can be directly to another link to prevent the terminal sessions from starving the bandwidth resources with bulky traffic that uses large packets (file transfers).

- One or more of the following **set** actions can be specified within a single **match** statement to perform customized packet forwarding requirement when the **match** statement is satisfied. A router evaluates the following **set** actions applicable for PBR in the order they are presented. As soon as a destination next-hop address or interface has been chosen, other **set** actions that changing the destination next-hop address or interface are ignored.

<b>set ip next-hop</b> {ip-addr} [... ip-addr]	Specifies a list of IP addresses of the adjacent next-hop routers to which the packets should be forwarded. It must be the IP address of an adjacent router. When multiple IP addresses are specified, the first IP address associated with an active interface is used; the other IP addresses are tried in turn. <sup>[1]</sup>
<b>set interface</b> {intf-type intf-num} [... intf-type intf-num]	Specifies a list of interfaces through which the packets can be routed. When multiple interfaces are specified, the first active interface is used to forward the packets; the other interfaces are tried in turn. <sup>[1]</sup>
<b>set ip default next-hop</b> {ip-addr} [... ip-addr]	Same logic as the <b>set ip next-hop</b> action; however, policy routing is applied only after attempted to route the packet using the normal destination-based routing.
<b>set default interface</b> {intf-type intf-num} [... intf-type intf-num]	Same logic as the <b>set interface</b> action; however, policy routing is applied only after attempted to route the packet using the normal destination-based routing.

[1] – If the PBR route does not work due to the outgoing interface is down or the next-hop address is unreachable using a directly-connected route, the packet is forwarded using the normal destination-based routing.

- The **default** keyword significantly impacts the operation of PBR. Without this keyword configured, Cisco IOS applies PBR **before** trying to use the normal destination-based routing; with this keyword configured, Cisco IOS applies PBR logic only **after** trying to use the normal destination-based routing and no explicit route for the destination existing in the routing table. A default route is not considered as an explicit route for an unknown destination address.
  - Without the **default** keyword – Try PBR first; but if the PBR route fails, route as usual.
  - With the **default** keyword – Try to route as usual while ignoring any default route; if normal routing fails, use PBR.

- The **set interface** action has no effect and is ignored if the packet is destined to a broadcast or an unknown address. This is because there is no explicit route for the destination address of the packet exists in the routing table.

- PBR also provides a mechanism to mark packets using the following commands:

<b>set ip tos</b> [ <i>number</i>   <i>name</i> ]	Sets the 8-bit IP ToS field in the IP packet header.
<b>set ip precedence</b> [ <i>number</i>   <i>name</i> ]	Sets the IP Precedence bits in the IP packet header.

- In the 8-bit IP ToS field, the 5 least-significant bits are used for setting the **Class of Service** (CoS); while the 3 most-significant bits are used for setting the **IP Precedence** – 8 possible values (0 – 7). IP Precedence is often being used when implementing QoS services, eg: **Weighted Fair Queuing** (WFQ) and **Weighted Random Early Detection** (WRED).

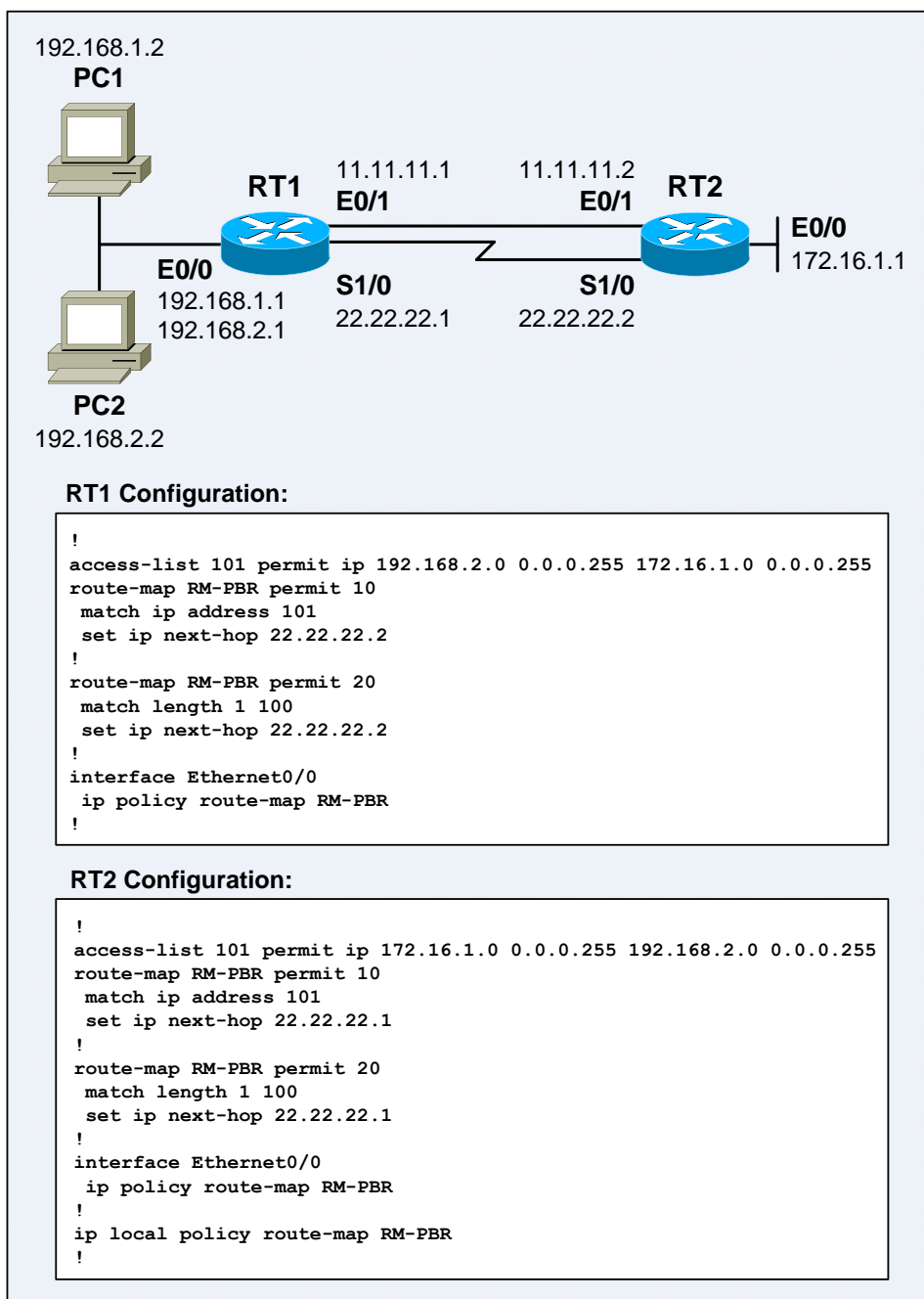
Below lists the numbers and names defined for the corresponding ToS and IP Precedence values:

<b>0   normal</b>	Sets the normal ToS.
<b>1   min-monetary-cost</b>	Sets the min-monetary-cost ToS.
<b>2   max-reliability</b>	Sets the max reliable ToS.
<b>4   max-throughput</b>	Sets the max throughput ToS.
<b>8   min-delay</b>	Sets the min delay ToS.
<b>0   routine</b>	Sets the routine precedence.
<b>1   priority</b>	Sets the priority precedence.
<b>2   immediate</b>	Sets the immediate precedence.
<b>3   flash</b>	Sets the Flash precedence.
<b>4   flash-override</b>	Sets the Flash override precedence.
<b>5   critical</b>	Sets the critical precedence.
<b>6   internet</b>	Sets the internetwork control precedence.
<b>7   network</b>	Sets the network control precedence.

The most commonly used QoS marking tool today is Class-Based Marking. PBR was one of the few tools that could be used for marking packets for this important QoS function in the past. PBR still support marking; however, most modern QoS designs ignore this capability of PBR. The IP header originally defines a ToS byte whose individual bits have been defined in various ways over the years. One such definition is the 3-bit IP Precedence field which uses the first 3 bits of the ToS byte. Back in the 1990s, the ToS byte was redefined as the Differentiated Services (DS) byte, with the first 6 bits defined as the **Differentiated Service Code Point** (DSCP). Most modern QoS implementations revolve around setting the DSCP field.

- The **ip policy route-map** {*map-tag*} interface subcommand specifies a route map to use for policy routing on an interface. Policy-based routing is configured on the **receiving** interface, not on the outgoing interface from which the packets are sent.
- Occasionally it may be necessary to use PBR to process packets originated by the router itself. However, such packets are not being processed and policy routed by **the ip policy route-map** interface subcommand, as the packets originated by the router itself do not actually enter the router through any interface. The **ip local policy route-map** {*map-tag*} global configuration command enables local policy routing to process locally originated packets using the PBR. The IP SLA section later shows an example use of this command. IP SLA causes a router to generate packets, applying PBR to such packets influence the forwarding paths of these packets.





**Figure 12-1: Network Setup for Policy-Based Routing**

- The sample network above demonstrates the PBR configuration for the following requirements:
  - EIGRP is running on RT1 and RT2. Therefore the primary path between 192.168.0.0/16 and 172.16.1.0/24 is via 11.11.11.0/30.
  - Packets sent between 192.168.2.0/24 and 172.16.1.0/24 are routed via 22.22.22.0/30.
  - Packets smaller than or equal to 100 bytes are routed via 22.22.22.0/30.
  - Other packets are routed normally via the primary path.
- Note that small ping packets sent between PC1, PC2, and RT1 are actually being policy routed! Besides that, there is a chance that the EIGRP Hello packets between 11.11.11.1 and 11.11.11.2 are also being policy routed, which will cause the flapping of EIGRP neighborship between RT1 E0/1 and RT2 E0/1. This symptom was seen on Cisco IOS Release 12.4(T) used for the IP SLA network setup later but was not seen on Cisco IOS Release 12.3 used for the network setup above. The configuration above will be enhanced to eliminate the mentioned problems.

- Below shows that PC1 sends 5 ping packets that are being policy routed via 22.22.22.0/30:

```
PC1#ping 172.16.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 64/78/88 ms
PC1#
-----
RT1#clear route-map counters RM-PBR
RT1#
RT1#debug ip policy
Policy routing debugging is on
RT1#
00:12:36: IP: s=192.168.1.2 (Ethernet0/0), d=172.16.1.1, len 100, FIB policy
match
00:12:36: IP: s=192.168.1.2 (Ethernet0/0), d=172.16.1.1, g=22.22.22.2, len
100, FIB policy routed
--- output omitted ---
RT1#
RT1#sh route-map
route-map RM-PBR, permit, sequence 10
  Match clauses:
    ip address (access-lists): 101
  Set clauses:
    ip next-hop 22.22.22.2
  Policy routing matches: 0 packets, 0 bytes
route-map RM-PBR, permit, sequence 20
  Match clauses:
    length 1 100
  Set clauses:
    ip next-hop 22.22.22.2
  Policy routing matches: 5 packets, 570 bytes
RT1#
```

The **show route-map {map-name} EXEC** command displays configured static and dynamic route maps along with their **match**, **set**, and continue clauses.

- Below shows that PC1 sends 5 ping packets that are **not** being policy routed via 22.22.22.0/30:

```
PC1#ping 172.16.1.1 size 101
Type escape sequence to abort.
Sending 5, 101-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/56/92 ms
PC1#
-----
RT1#
00:13:42: IP: s=192.168.1.2 (Ethernet0/0), d=172.16.1.1, len 101, FIB policy
rejected(no match) - normal forwarding
--- output omitted ---
```

- Below shows that all packets from PC2 are being policy routed via 22.22.22.0/30:

```

PC2#ping 172.16.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 40/61/80 ms
PC2#
-----

RT1#
00:14:18: IP: s=192.168.2.2 (Ethernet0/0), d=172.16.1.1, len 100, FIB policy match
00:14:18: IP: s=192.168.2.2 (Ethernet0/0), d=172.16.1.1, g=22.22.22.2, len 100, FIB policy routed
--- output omitted ---
=====

PC2#ping 172.16.1.1 size 101

Type escape sequence to abort.
Sending 5, 101-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 44/79/100 ms
PC2#
-----

RT1#
00:14:50: IP: s=192.168.2.2 (Ethernet0/0), d=172.16.1.1, len 101, FIB policy match
00:14:50: IP: s=192.168.2.2 (Ethernet0/0), d=172.16.1.1, g=22.22.22.2, len 101, FIB policy routed
--- output omitted ---

RT1#
RT1#sh route-map
route-map RM-PBR, permit, sequence 10
  Match clauses:
    ip address (access-lists): 101
  Set clauses:
    ip next-hop 22.22.22.2
    Policy routing matches: 10 packets, 1145 bytes
route-map RM-PBR, permit, sequence 20
  Match clauses:
    length 1 100
  Set clauses:
    ip next-hop 22.22.22.2
    Policy routing matches: 5 packets, 570 bytes
RT1#

```

- The **show ip policy EXEC** command displays the route maps used for policy routing:

```

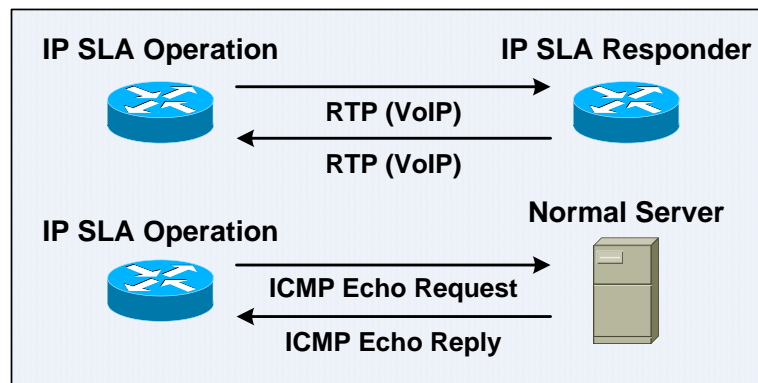
RT1#sh ip policy
Interface      Route map
Ethernet0/0    RM-PBR
RT1#
=====

RT2#sh ip policy
Interface      Route map
local          RM-PBR
Ethernet0/0    RM-PBR
RT2#

```

## IP SLA (Service-Level Agreement)

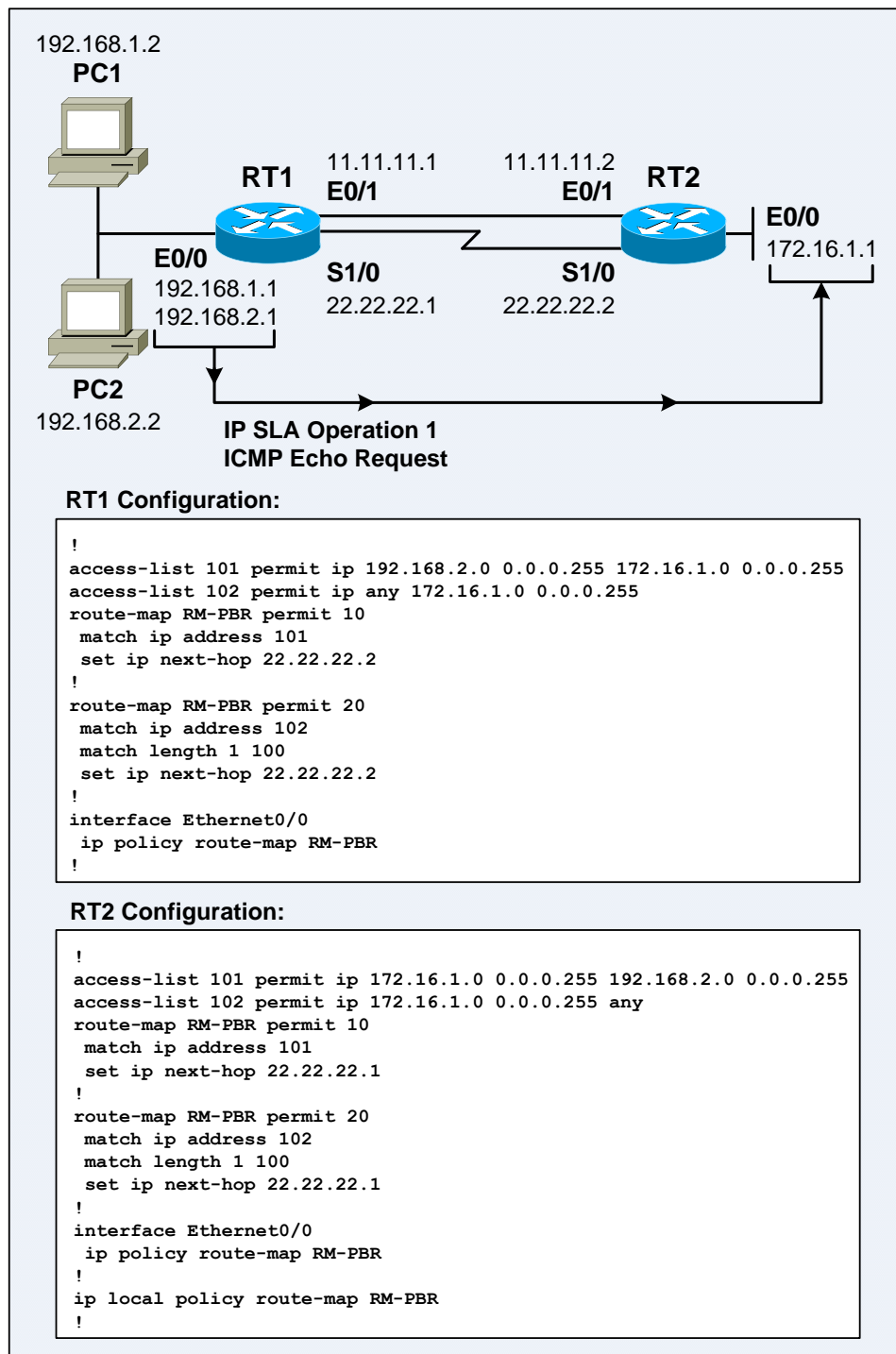
- The Cisco IOS **IP Service-Level Agreement** (IP SLA) feature allows Cisco customers to analyze and measure IP service levels for IP applications and services, to increase productivity, to lower operational costs, and to reduce network outages. The measurement can be as simple as using ICMP ping packets to determine if an IP address responds, or as sophisticated as measuring the jitter (inter-packet delay variance) of VoIP packets that flow through a particular path.
- Service provider customers can use IP SLA to measure and provide service level agreements; while enterprise customers can use IP SLA to verify the outsourced service level agreements, and understand network performance for both existing and new IP applications and services. IP SLA provides improvements over the traditional Layer 2 SLA to provide a broader scope to support end-to-end sophisticated and accurate application-aware performance measurements.
- Various IP SLA operations provide useful measurement statistics for network assessments, verifying Quality of Service (QoS), easing the deployment of new IP applications and services, designing networks, problem analysis, and network troubleshooting.
- IP SLA generates traffic in a continuous, predictable, and reliable manner to measure network performance. IP SLA performs active monitoring by generating and analyzing traffic to measure performance either between Cisco IOS devices or between a Cisco IOS device and a normal application server. IP SLA uses unique service level assurance metrics and methodology to provide highly accurate, precise service level assurance measurements.
- IP SLA sends data across the network to measure performance between multiple network locations or across multiple network paths. It simulates network data and IP services, and collects network performance information in real time. The information collected includes round-trip response time, one-way latency, jitter (inter-packet delay variance), packet loss, packet sequence, voice quality scoring, network resource availability, application performance, server response time, and download time.
- The statistics collected by IP SLA operations are stored in both CLI and SNMP MIBs that can be assessed using the CLI or SNMP through the **Cisco Round-Trip Time Monitor** (RTTMON) and SYSLOG Management Information Bases (MIBs). Network management systems such as CiscoWorks Internetwork Performance Monitor (IPM) and other 3rd party Cisco partner performance management products are able to collect the statistics data using SNMP and generate reports to justify whether a network has reached the SLAs defined for it.
- IP SLA mainly uses **operations** for its operation. Each operation defines the packet type to be generated, the source and destination addresses, and other characteristics of the packets. The configuration includes the time-of-day when the router should generate the packets, the types of statistics that should be gathered, and how often the packets should be generated. A single router can be configured with different types of IP SLA operations.  
**Note:** IP SLA has origins in earlier IOS, including the **Response Time Reporter** (RTR) feature. The RTR feature uses the term **probe** to refer to what IP SLA refers to as an **operation**.
- An IP SLA operation is able to send packets to any IP address, whether it is a router or a host. When sending to a host, no special software or configuration is required on the host; indeed the IP SLA operation send operational packets only to the native services on the host, eg: ICMP echo requests, TCP connection requests, or even HTTP GET requests to a web server. The server will try to respond to these requests.



**Figure 12-2:** The Operation of IP SLA

- An IP SLA operation can also send packets to another router that acts as an IP SLA **responder**, which provides a wider range of possible operation types. An IP SLA responder responds to IP SLA request packets that a router would not normally respond to, providing a way to monitor network behavior without introducing dedicated probes around the network to test the network. Ex: An IP SLA operation could send **Real-time Transport Protocol (RTP)** packets that have the same content as VoIP packets to an IP SLA responder. The IP SLA responder replies the packets back to the IP SLA operation as if a voice call exists between the 2 routers.
- Below list the various types of IP SLA operations:
  - Dynamic Host Configuration Protocol (DHCP)
  - Data Link Switching Plus (DLSw+)
  - Domain Name System (DNS)
  - File Transfer Protocol (FTP)
  - Hypertext Transfer Protocol (HTTP)
  - ICMP Echo and ICMP Path Echo
  - ICMP Jitter and ICMP Path Jitter
  - Transmission Control Protocol (TCP) Connect
  - UDP Echo and UDP Jitter
  - VoIP gatekeeper registration delay
  - VoIP post-dial delay
  - VoIP Real-time Transport Protocol (RTP)
- Below are the general steps to configure an IP SLA ICMP Echo operation:
  - Step 1) Create the IP SLA operation and assign it an integer operation number using the **ip sla {sla-ops-num}** global configuration command.
  - Step 2) Define the IP SLA operation type and the parameters for that operation type. For ICMP Echo, define the source (optional) and destination IP addresses or hostnames using the **icmp-echo {dest-ip-addr | dest-hostname} [source-ip {src-ip-addr | src-hostname} | source-interface {intf-type intf-num}]** IP SLA operation subcommand.
  - Step 3) (Optional) Define a (non-default) frequency that the IP SLA operation should send the packets using the **frequency {seconds}** IP SLA subcommand. The default frequency is 60 seconds.
  - Step 4) Schedule the IP SLA operation to run for a period to gather statistics using the **ip sla schedule {sla-ops-num} [life {forever | seconds}] [start-time {hh:mm[:ss] [month day | day month]} pending now | after hh:mm:ss] [ageout seconds] [recurring]** global configuration command.

- Static routes and PBR can be configured to utilize the capabilities of IP SLA operations upon controlling the usages of the static and PBR paths based upon the failure or reduced performance (based on the configured threshold) of a particular measurement.
- This section describes the concepts of IP SLA and along with its configuration in enough depth for influencing static routes and policy routing. Below implements an ICMP echo operation, which requires configuration only on one router, with no IP SLA responder. The remote router or host replies to the IP SLA ICMP Echo Requests just like any other ICMP Echo Requests.



**Figure 12-3: IP SLA Echo Operation**

- **Note:** The configuration is enhanced so that small ping packets sent between PC1, PC2, and RT1 as well as EIGRP Hellos packets between 11.11.11.1 and 11.11.11.2 are not being policy routed.

- Below shows the process of implementing an ICMP Echo operation on RT1 with the purpose to test the PBR route through the secondary path – 22.22.22.0/30. The operation is configured to:
  - Send ICMP Echo Requests to RT2 E0/0 – 172.16.1.1).
  - Use the secondary IP address of RT1 E0/0 (192.168.2.1) as the source IP address.
  - Send the packets every 10 seconds.
  - Start the operation immediately, and run it forever.
  - Enable local policy routing on RT1 for the locally originated IP SLA ICMP Echo operation packets to flow through the secondary path.
- Configuring an IP SLA ICMP Echo operation on RT1:

```

RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#ip sla 1
RT1(config-ip-sla)#icmp-echo 172.16.1.1 source-ip 192.168.2.1
RT1(config-ip-sla-echo)#frequency 10
RT1(config-ip-sla-echo)#exit
RT1(config)#ip sla schedule 1 life forever start-time now
RT1(config)#ip local policy route-map RM-PBR

```

- The parameters of the **icmp-echo** IP SLA subcommand acts as an extended ping from the CLI, which it specifies both the source and destination IP addresses.
- The **show ip sla configuration** [sla-ops-num] EXEC command displays the configuration details including all the default values for all IP SLA operations or a specified IP SLA operation:

```

RT1#sh ip sla configuration
IP SLAs Infrastructure Engine-II
Entry number: 1
Owner:
Tag:
Type of operation to perform: icmp-echo
Target address/Source address: 172.16.1.1/192.168.2.1
Operation timeout (milliseconds): 5000
Type Of Service parameters: 0x0
Vrf Name:
Request size (ARR data portion): 28
Verify data: No
Schedule:
  Operation frequency (seconds): 10 (not considered if randomly scheduled)
  Next Scheduled Start Time: Start Time already passed
  Group Scheduled : FALSE
  Randomly Scheduled : FALSE
  Life (seconds): Forever
  Entry Ageout (seconds): never
  Recurring (Starting Everyday): FALSE
  Status of entry (SNMP RowStatus): Active
Threshold (milliseconds): 5000 (not considered if react RTT is configured)
Distribution Statistics:
  Number of statistic hours kept: 2
  Number of statistic distribution buckets kept: 1
  Statistic distribution interval (milliseconds): 20
History Statistics:
  Number of history Lives kept: 0
  Number of history Buckets kept: 15
  History Filter Type: None
Enhanced History:

RT1#

```

- The **show ip sla statistics** [*sla-ops-num*] [**details**] EXEC command displays the current operational status and statistics for all IP SLA operations or a specified IP SLA operation:

```

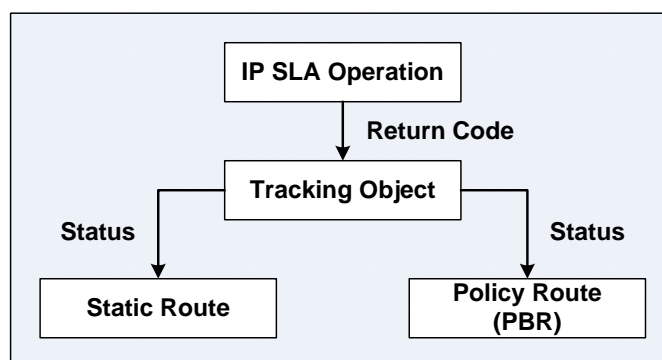
RT1#sh ip sla statistics
IPSLAs Latest Operation Statistics

IPSLA operation id: 1
  Latest RTT: 28 milliseconds
Latest operation start time: *13:44:38.003 UTC Fri May 14 2010
Latest operation return code: OK
Number of successes: 3
Number of failures: 0
Operation time to live: Forever

RT1#

```

The command output shows that 3 intervals have passed, 3 ICMP Echo Requests were success with no failures; the latest round-trip time, and the return code of the most recent operation – a key value used for **IP SLA tracking**.



**Figure 12-4: IP SLA Tracking**

- A static or policy route can be configured in a way that it will be used only when an IP SLA operation remains successful. The configuration introduces a tracking object that is being referenced by the static route, policy route, and IP SLA operation as shown in the figure above.
- The tracking object determines its tracking state as either UP or DOWN based on the most recent return code of the IP SLA operation. The IP SLA return codes vary upon different types of IP SLA operations. The return code may be an OK meaning that the last operation is successful, in which the tracking object would result in an UP state; or other return codes for other IP SLA operations that define thresholds. The tracking operation remains in an UP state as long as the result of the IP SLA operation is within the configured threshold.
- Another great usage of the tracking object is to control route flapping. **Route flap** occurs when a router adding and removing a route in the routing table in a very frequent manner. If a static route is being tracked by an IP SLA operation, the return code of the IP SLA operation could change upon every operation and results in a route flap. A delay can be set to determine how long a tracking object must wait to change its state upon a tracking state change.



- Below lists the steps to configure a static route to track an IP SLA operation:
  - Step 1) Create a tracking object to an IP SLA operation using the **track {obj-num} ip sla {sla-ops-num} {state | reachability}** global configuration command. The **state** and **reachability** keywords track the IP SLA operation return code and whether the route is reachable respectively.  
**Note:** This command was introduced on Cisco IOS Release 12.4(20)T.
  - Step 2) (Optional) Configure the delay period to regulate flapping of the tracking state using the **delay {down {seconds} | up {seconds}}**. The **down** and **up** keywords indicate the delay periods for the notifications of a down event and an UP event respectively.
  - Step 3) Configure the static route using the **ip route {dest-network} {subnet-mask} {next-hop-addr | outgoing-intf} track {obj-num}** global configuration command.
- Ex: The **delay down 30 up 10** tracking subcommand indicates that the clients that tracking the object are notified after 30 seconds and 10 seconds upon the tracking state changes from UP to DOWN and from DOWN to UP respectively.
- Below implements a static route that tracks a tracking object that tracking the existing IP SLA ICMP Echo operation on RT1:

```

RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#track 1 ip sla 1 state
RT1(config-track)#delay down 30 up 10
RT1(config-track)#exit
RT1(config)#
RT1(config)#ip route 172.16.2.0 255.255.255.0 11.11.11.2 track 1
RT1(config)#end
RT1#sh ip route

Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 2 subnets
D       172.16.1.0 [90/30720] via 11.11.11.2, 00:04:24, FastEthernet0/1
S       172.16.2.0 [1/0] via 11.11.11.2
    22.0.0.0/30 is subnetted, 1 subnets
C       22.22.22.0 is directly connected, Serial1/0
    11.0.0.0/30 is subnetted, 1 subnets
C       11.11.11.0 is directly connected, FastEthernet0/1
C       192.168.1.0/24 is directly connected, FastEthernet0/0
C       192.168.2.0/24 is directly connected, FastEthernet0/0
RT1#
RT1#sh track
Track 1
  IP SLA 1 state
  State is Up
    1 change, last change 00:00:51
  Delay up 10 secs, down 30 secs
  Latest operation return code: OK
  Latest RTT (milliseconds) 60
  Tracked by:
    STATIC-IP-ROUTING 0
RT1#

```

- Below implements an access list on RT2 to block ICMP Echo packets which results in the IP SLA operation fails, and causes the static route to be removed on RT1. RT1 will then uses the EIGRP route to reach 172.16.2.0/24.

```

RT2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT2(config)#access-list 111 deny icmp any any
RT2(config)#access-list 111 permit ip any any
RT2(config)#int s1/0
RT2(config-if)#ip access-group 111 in
RT2(config-if)#
=====
RT1#
14:13:53: %TRACKING-5-STATE: 1 ip sla 1 state Up->Down
RT1#
RT1#sh ip sla statistics
IPSLAs Latest Operation Statistics

IPSLA operation id: 1
    Latest RTT: NoConnection/Busy/Timeout
Latest operation start time: *14:13:49.927 UTC Fri May 14 2010
Latest operation return code: No connection
Number of successes: 14
Number of failures: 4
Operation time to live: Forever

RT1#
RT1#sh track
Track 1
    IP SLA 1 state
    State is Down
    2 changes, last change 00:00:07
    Delay up 10 secs, down 30 secs
    Latest operation return code: No connection
    Tracked by:
        STATIC-IP-ROUTING 0
RT1#
RT1#sh ip route

Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 2 subnets
D       172.16.1.0 [90/30720] via 11.11.11.2, 00:06:18, FastEthernet0/1
D       172.16.2.0 [90/30720] via 11.11.11.2, 00:00:12, FastEthernet0/1
    22.0.0.0/30 is subnetted, 1 subnets
C       22.22.22.0 is directly connected, Serial1/0
    11.0.0.0/30 is subnetted, 1 subnets
C       11.11.11.0 is directly connected, FastEthernet0/1
C       192.168.1.0/24 is directly connected, FastEthernet0/0
C       192.168.2.0/24 is directly connected, FastEthernet0/0
RT1#
RT1#ping 172.16.2.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.2.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/24/52 ms
RT1#

```

- The earlier **set** action in the route map used for the PBR – **set ip next-hop 22.22.22.x** can be modified to use object tracking to track the availability of the next hop – 22.22.22.x using the **set ip next-hop verify-availability 22.22.22.x {seq-num} track 1** action.
- Below shows that when the tracking object is UP, PBR works as configured. When the tracking object is DOWN, PBR acts as if the **set** action does not exist, and proceed to route the packets as per the normal destination-based routing process.  
**Note:** Similar configuration should be implemented on RT2 as well to avoid asymmetric routing.

```

RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#route-map RM-PBR permit 10
RT1(config-route-map)#no set ip next-hop 22.22.22.2
RT1(config-route-map)#set ip next-hop verify-availability 22.22.22.2 1 track 1
RT1(config-route-map)#route-map RM-PBR permit 20
RT1(config-route-map)#no set ip next-hop 22.22.22.2
RT1(config-route-map)#set ip next-hop verify-availability 22.22.22.2 1 track 1
RT1(config-route-map)#end
RT1#
RT1#sh route
route-map RM-PBR, permit, sequence 10
  Match clauses:
    ip address (access-lists): 101
  Set clauses:
    ip next-hop verify-availability 22.22.22.2 1 track 1 [up]
  Policy routing matches: 24 packets, 1536 bytes
route-map RM-PBR, permit, sequence 20
  Match clauses:
    ip address (access-lists): 102
    length 1 100
  Set clauses:
    ip next-hop verify-availability 22.22.22.2 1 track 1 [up]
  Policy routing matches: 0 packets, 0 bytes
RT1#
RT1#sh track
Track 1
  IP SLA 1 state
  State is Up
    1 change, last change 00:00:47
  Delay up 10 secs, down 30 secs
  Latest operation return code: OK  Latest RTT (milliseconds) 100
  Tracked by:
    ROUTE-MAP 0
    STATIC-IP-ROUTING 0
RT1#
! After applied the Block-ICMP-ACL on RT2 Serial1/0 and waited 30 seconds...
16:33:17: %TRACKING-5-STATE: 1 ip sla 1 state Up->Down
RT1#sh route
route-map RM-PBR, permit, sequence 10
  Match clauses:
    ip address (access-lists): 101
  Set clauses:
    ip next-hop verify-availability 22.22.22.2 1 track 1 [down]
  Policy routing matches: 26 packets, 1664 bytes
route-map RM-PBR, permit, sequence 20
  Match clauses:
    ip address (access-lists): 102
    length 1 100
  Set clauses:
    ip next-hop verify-availability 22.22.22.2 1 track 1 [down]
  Policy routing matches: 0 packets, 0 bytes
RT1#

```

## Basic BGP

- Many organizations that have redundant connections to multiple Internet Service Providers (ISPs) often implement **Border Gateway Protocol (BGP)** as the alternative to default routes for path selections and load balancing across multiple Internet connections.
- BGP is a very different and complex protocol and can be hard to understand, configure, and troubleshoot for those who have only ever dealt with interior routing protocols, eg: EIGRP and OSPF, due to the nature of the protocol is different – BGP does not deal with hops and links, but rather with autonomous systems. It is important to understand the various options required for proper BGP configuration for building scalable internetworks.
- Understanding BGP first requires an understanding of autonomous systems. The classic definition of an AS is a set of routers under a single technical administration, using an IGP and common routing strategy to route packets within the AS, and using an EGP to route packets to other ASes. Today, ASes might use multiple IGPs, and hence several sets of metrics.
- **Interior Gateway Protocols (IGPs)** are used for exchanging routing information **within** an autonomous system (AS). Ex: RIP, IGRP, EIGRP, OSPF, and IS-IS.  
**Exterior Gateway Protocols (EGPs)** are used for communication **between** autonomous systems. The main goal of EGPs is to provide an inter-domain routing system that exchanges guaranteed loop-free routing information between autonomous systems throughout the Internet and within multinational organizations. EGPs are not meant to locate a specific network, but rather provide the information to find the autonomous system in which the network resides. Ex: BGP.
- BGP is a successor to **Exterior Gateway Protocol (EGP)** that was developed at the early stages of the Internet. RFC 1771 – BGP Version 4 (BGP-4) defines the latest version of BGP. BGP-4 has many enhancements over earlier EGPs and is currently being used extensively on the Internet today to interconnect between ISPs as well as between enterprises and ISPs. RFC 2858 – Multiprotocol Extensions for BGP-4 (**BGP4+ / MBGP / MP-BGP4**) have been defined to support new protocols, eg: IPv6. BGP-4 and its extensions are the only acceptable version of BGP available for use on the public-based Internet today.
- The **Internet Assigned Numbers Authority (IANA)** is responsible for allocating AS numbers. Specifically, the **American Registry for Internet Numbers (ARIN)** for the Americas, the Caribbean, and Africa. **Réseaux IP Européens-NIC (RIPE-NIC)** for Europe, Middle East, and Central Asia, and the **Asia Pacific-NIC (AP-NIC)** for Asia-Pacific.
- The AS designator is a 16-bit number, ranging from 1 to 65535. RFC 1930 – Guidelines for Creation, Selection, and Registration of an Autonomous System (AS) provides guideless for the usages of AS numbers. The AS numbers ranging from 64512 to 65535 (inclusive, total 1023) are reserved for private use and not to be advertised on the Internet, similar to the private IP addresses. The IANA-assigned AS numbers are only required when an organization plans to connect to the Internet using BGP.
- BGP-4 advertises networks along with their subnet masks, and supports both VLSM and CIDR. These capabilities which are mandatory on the Internet were not supported by its predecessors. A router of a major ISP using CIDR has more that 120'000 CIDR blocks in its IP routing table, which contains mostly BGP routes. Not using CIDR at the Internet level will case the IP routing table to have more than 2 million entries! Therefore, using BGP-4 along with CIDR prevents the Internet routing table from becoming too large when interconnecting millions of Internet users.

- Below shows the numbers of BGP routes on a public route server (route-views3.routeviews.org) at the time of writing. Route servers are BGP routers setup by Internet service providers for managing and maintaining the BGP sessions and route advertisements throughout the Internet. Anonymous Telnet access is often granted and this is an excellent way to learn more about BGP.

```

route-views3>sh clock
05:41:25.653 UTC Wed May 19 2010
route-views3>
route-views3>sh ip bgp summary
BGP router identifier 128.223.51.108, local AS number 6447
BGP table version is 388604641, main routing table version 388604641
323834 network entries using 42746088 bytes of memory
5706068 path entries using 296715536 bytes of memory
964580/58839 BGP path/bestpath attribute entries using 162049440 bytes of memory
694842 BGP AS-PATH entries using 27109354 bytes of memory
19744 BGP community entries using 1658510 bytes of memory
24 BGP extended community entries using 2416 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 530281344 total bytes of memory
BGP activity 579299/252518 prefixes, 23678831/17967101 paths, scan interval 60
secs

--- output omitted ---
route-views3>sh ip route summary
IP routing table name is Default-IP-Routing-Table(0)
IP routing table maximum-paths is 32
Route Source      Networks      Subnets      Overhead      Memory (bytes)
connected         0             1             64            152
static            1             27            1792          4256
bgp 6447          139352       184249        20710464      49445412
  External: 323601 Internal: 0 Local: 0
internal          4341
Total             143694       184277        20712320      54537472
Removing Queue Size 0
route-views3>

```

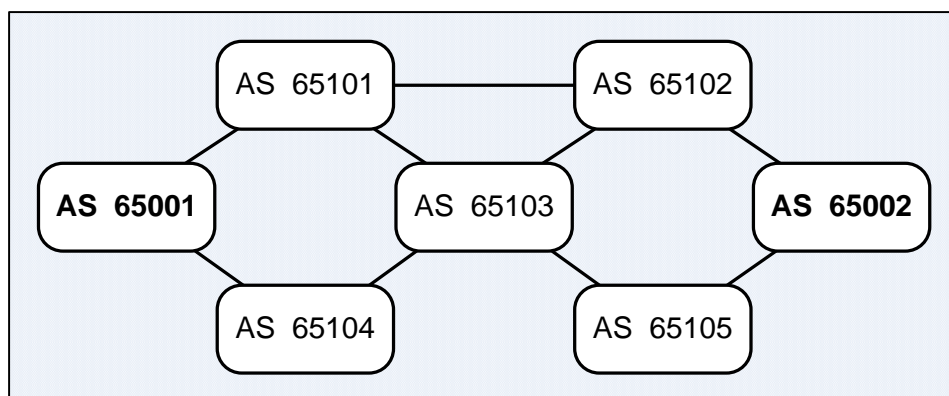
- <http://www.potaroo.net>, a website maintained by Geoff Huston, Chief Scientist at APNIC, has tracked Internet growth for many years. It is hosting many Internet-related articles and reports.

- Below shows the comparison chart of scalable IP routing protocols:

Protocol	Type	Metric	Requires Hierarchical Design?
OSPF	Interior, Link-State.	Cost	Yes
IS-IS	Interior, Link-State.	Metric	Yes
EIGRP	Interior, Advanced Distance-Vector.	Composite	No
BGP	Exterior, Path-Vector.	Attributes	No

- BGP is sometimes categorized as an advanced distance-vector protocol with many differences, features, and enhancements. It is actually known as a **Path-Vector** routing protocol.
- BGP does not communicate full knowledge of every subnet within an autonomous system, but only in conveying enough information to locate and reach an autonomous system. BGP routers perform summarization to the extreme and exchange network reachability information called **path vectors** or **path attributes**, which contains a list of the AS numbers (ASNs) to reach a destination network, the aggregated prefix addresses, and some characteristics of the network paths that can be used to implement sophisticated policy-based routing.

- The metrics used in BGP are sophisticated and are the source of its complexity and strength. Instead of simply choosing the best route by using an integer metric, the BGP metric, which is being referred to as **path attributes**, contain variety of information and provide great granularity and flexibility for the best path selection. BGP is a policy-based routing protocol that allows an autonomous system to determine the traffic flow throughout the AS using BGP attributes. Policies can be implemented for a certain CIDR block of prefixes, or for individual prefixes.
- BGP maintains its own routing table that is separate from the IP routing table for storing BGP information received from and sent to other BGP routers. A BGP router can be configured to share information between the BGP table and IP routing table. The BGP table is also known as BGP routing table, BGP topology table, BGP topology database, and BGP forwarding database.
- A BGP router also maintains a neighbor table that contains a list of its BGP neighbors. A BGP router establishes a TCP session with each configured neighbor and keeps track of the state of the relationships by sending periodic BGP keepalive messages. BGP routers do not send periodic routing updates, and BGP routers advertise only the **best path** to a destination network.
- After an adjacency is established, BGP neighbors exchange their BGP routes and place them into their BGP table. The best routes for each network are selected from the BGP table using the **BGP best path selection algorithm** and are then installed into the IP routing table based upon the administrative distance rule. EBGP routes have an administrative of 20; while IBGP routes have as administrative distance of 200.
- The **hop-by-hop routing paradigm** that is being used throughout the current Internet specifies that an AS can advertise only those routes that it itself uses to its neighboring ASes; and an AS is not allowed to influence how the neighboring AS will route its traffic, but it is allowed to determine how to route the traffic originated from the local AS to a neighboring AS. Since BGP supports any policy that conforms to that paradigm, it is being used as the inter-AS routing protocol for the current Internet. Some policies cannot be supported by the hop-by-hop routing paradigm, and therefore require techniques such as source routing.



**Figure 13-1:** The Internet Hop-by-Hop Routing Paradigm

- Below are the possible paths for AS 65001 to reach networks in AS 65002 through AS 65101:
  - 65101, 65102, 65002
  - 65101, 65102, 65103, 65105, 65002
  - 65101, 65103, 65102, 65002
  - 65101, 65103, 65105, 65002

- However, AS 65001 does not see all these possibilities. AS 65101 advertises only its best path – 65101, 65102, 65002 to AS 65001; similar to how IGP's announce their best least-cost routes. This is the only path through AS 65101 that AS 65001 sees. All packets that are originated from AS 65001 and destined for AS 65002 take this path, as it is the AS-by-AS (hop-by-hop) path that AS 65101 uses to reach the networks in AS 65002. AS 65101 does not announce the other paths, as it has chosen its best path based on the BGP routing policy in AS 65101.
- AS 65001 does learn another path to AS 65002 through AS 65104. However, even AS 65001 aware the alternative path, it has selected 65101 65102 65002 as its best path based on its own BGP routing policies. All routers in AS 65101 will use that path to forward packets to AS 65002.

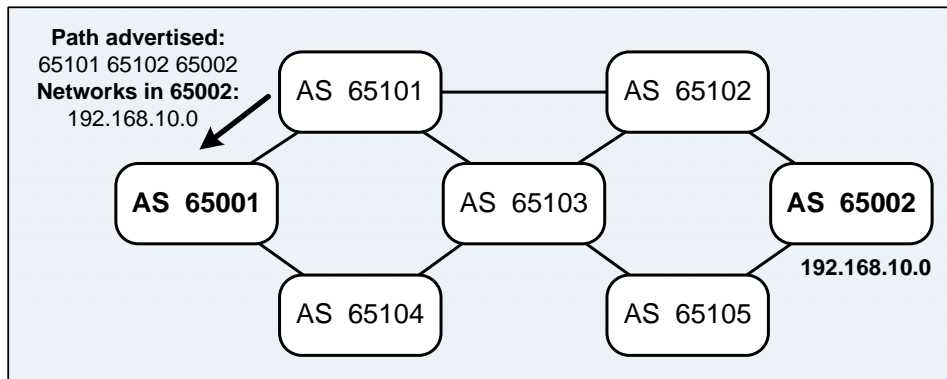


Figure 13-2: BGP Path-Vector Routing

- The BGP AS\_PATH path attribute associated with a prefix/length lists the ASNs that an end-to-end for the BGP-learned prefix, which implies that “If you use this path (route), the path will go through this list of ASNs”. The BGP AS\_PATH is guaranteed to always be loop-free. A BGP router does not accept a routing update that includes its AS number in the path list, as if an update has passed through the local AS, accepting it again may result in a routing loop.
- BGP uses the AS\_PATH attribute to choose the best route for a prefix based on the shortest AS\_PATH (least number of ASNs listed; assuming that no other path attributes are manipulated) as well as prevent routing loops.

**Note:** A **prefix** is simply a network address with the subnet mask, eg: 192.168.1.0/24;

a **route** often includes more information than just the prefix, eg: the next-hop IP address and the interface used to forward the packets to the prefix.

- Below shows the BGP table on a router resides in AS 65001. It receives 2 paths from AS 65101 and AS 65104. The > sign on the 2nd entry indicates that BGP considers it as the preferred path.

```
RT1#sh ip bgp
BGP table version is 3, local router ID is 15.15.15.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network        Next Hop        Metric LocPrf Weight Path
*   192.168.10.0   15.15.15.5      0      0 65104 65103 65105 65002 i
*>                12.12.12.2      0      0 65101 65102 65002 i
RT1#
```

- Duplicate ASN can prevent route advertisement – an AS that receives a prefix update that lists its ASN in the AS\_PATH would ignore the update as part of the BGP loop prevention process. Therefore, routers in the local AS are unable to reach the prefixes advertised by another AS.

## When to Use and Not to Use BGP

- BGP is not always the appropriate solution to interconnect ASes. It is important to understand when to use BGP and when to use other solutions, eg: static and default routes. If there is only one exit path from an AS, a static or default route is much more appropriate; implementing BGP achieve nothing except using the CPU and memory resources on the routers. 😊  
It is not necessary to implement BGP in an enterprise AS that has a local routing policy that is consistent with the routing policy implemented in the ISP AS. The only time BGP is required is when the local policy differs from the ISP policy.  
“Not as many internetworks need BGP as you might think” – Jeff Doyle.
- BGP is particularly complex when determining the path that should be taken or when used in conjunction with route maps to implement policy-based routing. Its complexity is its strength. Its brevity and path determination make it a specialized routing protocol. Implementing BGP in an AS is most appropriate when the effects of BGP are well understood and at least one of the following specific situations and conditions exists:
  - The AS has multiple connections to other autonomous systems and is actively using them for redundancy purposes. PBR decisions might need to be made on a link-by-link basis.
  - The AS needs or wants to manipulate the traffic flow that entering and leaving it.
  - The AS is an ISP and therefore conforms to both criteria above. The nature of the business requires the traffic between other autonomous systems to travel across the AS, treating it as a transit autonomous system or domain.
- A simple network is a network that is easy to manage and maintain, which is the main reason to avoid BGP configuration. **Do not use BGP** when one or more of the following conditions exist:
  - Have limited understanding of route filtering and the **BGP best path selection algorithm**.
  - A single connection to the Internet or another AS.
  - Although there are multiple links to the ISP, these links are for redundancy purpose and there are no plans to activate more than one link to the Internet.
  - Limited network resources (eg: memory and processor power) on the routers to handle constant BGP updates.
  - Low bandwidth between the autonomous systems.
- If an enterprise has a policy that requires it to differentiate between traffic from an AS and traffic from its ISP, the AS must connect to its ISP using BGP.
- BGP was designed to allow ISPs to communicate and exchange routes. These ISPs have multiple connections to one another and have agreements to exchange routing updates. BGP is the protocol that is used to implement these agreements between autonomous systems.
- If BGP is not properly controlled and filtered, it has the potential to allow an outside AS to affect the traffic flow to an AS. Ex: An enterprise connected to ISP1 and ISP2 for redundancy. A routing policy should be implemented to ensure that ISP1 does not send traffic to ISP2 via the enterprise AS. The enterprise wants to be able to receive traffic destined to the ISPs but do not want to waste valuable resources and bandwidth within its AS to route traffic for its ISPs! Therefore it is important to know and understand how BGP operates and configure it properly to prevent this from happening.



## BGP Neighbor Relationships

- There are more than 20'000 BGP routers connected to the Internet, representing more than 15'000 ASes. No single router can handle communications with all other BGP routers. A BGP router forms a direct neighbor relationship with a limited number of BGP routers. A BGP router learns the paths throughout the Internet through its BGP neighbors.
- A BGP neighbor is also known as a **BGP peer**.  
A BGP router is also known as a **BGP speaker**.
- BGP peers can be either internal or external to an AS. BGP can be implemented in either between autonomous systems or across / within an autonomous system. When BGP is running between routers reside in different ASes and used to connect the ASes, it acts as an EGP and is referred to as **External BGP (EBGP)**. EBGP routers are often directly connected to each other. BGP can also carry the external information between EBGP routers reside within the same AS, which is called **Interior BGP (IBGP)**. IBGP routers exchange BGP routing information so that all IBGP routers have the consistent BGP routing information about external ASes; and this information can be passed to other ASes.

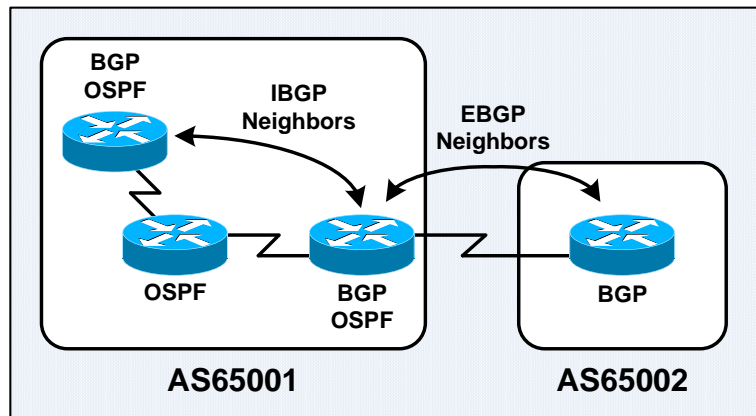
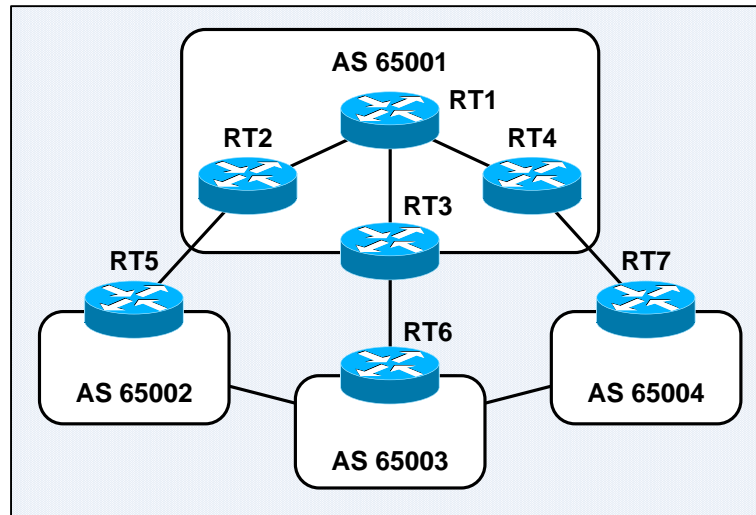


Figure 12-3: Internal and External BGP Neighbors

- IBGP routers do not have to be directly connected to each other, as long as they are able to reach each other to perform the TCP three-way handshake to establish a BGP neighbor relationship. An IBGP neighbor can be reached via a directly-connected network, static routes, or an IGP, eg: EIGRP and OSPF.
- As multiple paths often exist within an AS to reach other routers, a loopback address is usually used in conjunction with the **neighbor** BGP router subcommand to establish IBGP sessions.
- A BGP router behaves differently in several ways depending on whether the peer is an IBGP or EBGP neighbor, eg: when advertising a prefix to an EBGP neighbor, a BGP router updates the AS\_PATH path attribute; but it does not do so when advertising a prefix to an IBGP neighbor.

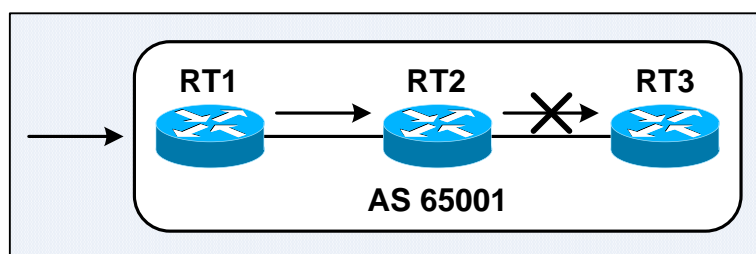


**Figure 13-4:** IBGP Neighbors Reside within the Same Autonomous System

- RT2, RT3, and RT4 learn the paths to external ASes from their corresponding EBGP neighbors – RT5, RT6, and RT7. If the link between RT3 and RT6 fails, RT3 must learn the new paths to the external ASes through its IBGP neighbors. Other IBGP routers within AS 65001 must also be informed that the path to external networks through RT3 is unavailable. IBGP sessions must be established between all BGP routers reside in AS 65001 for them to learn other paths through RT2 and RT4 to the external networks via IBGP.

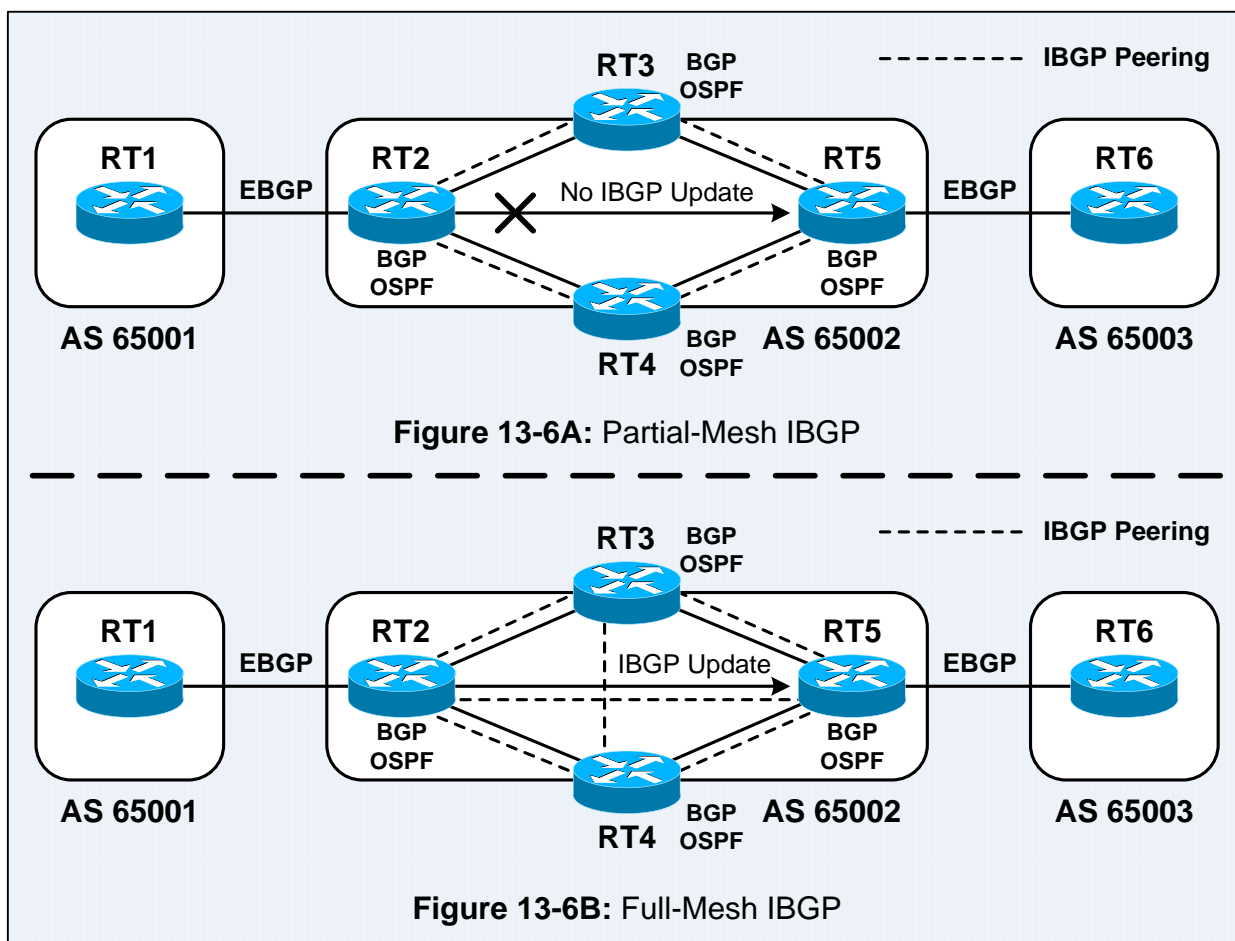
## BGP Split-Horizon and Full-Mesh IBGP Neighbors

- BGP was originally intended to run along the borders of an AS, with the routers in the middle of the AS ignore of the details of BGP – hence the name “Border Gateway Protocol”. A **transit AS** is an AS that routes traffic from an external AS to another external AS. Typically, transit ASes are ISPs. All routers in a transit AS must have complete knowledge of external routes. One way to achieve this goal is to redistribute BGP routes into an IGP at the edge routers; however, this approach introduced many problems.
- In 1994 the size of the Internet routing table was only about 4 to 8MB, so BGP could be redistributed into the local IGP, eg: EIGRP and OSPF. The edge routers running BGP would hold the full Internet routing table; and the routers in the middle that are running only the IGP, would not incur the overhead of running BGP but would still know about all the external routes.
- As the current Internet routing table is very large, redistributing all the BGP routes into an IGP is not a scalable way for the interior routers within an AS to learn about the external networks. Running full-mesh IBGP within the AS is a viable alternative.
- The **BGP split-horizon** rule governs the route advertisements between IBGP peers, which specifies that routes learn via IBGP are never propagated to other IBGP peers.  
**Note:** The BGP split-horizon rule is slightly different that the split-horizon rule as in the distance vector routing protocols.  
**Note:** Regular split-horizon rule still govern the route advertisements between EBGP peers, in which a route is not advertised back to the EBGP peer from which the route was received.



**Figure 13-5: BGP Split-Horizon Rule**

- The BGP split-horizon rule prevents RT2 from propagating routes learned from RT1 to RT3. Similar to the split-horizon rule in the distance-vector routing protocols, BGP split-horizon is necessary to ensure that routing loops are not started within an AS. As a result, full-mesh IBGP peering is required within an AS for all the routers within the AS to learn about the BGP routes.
- IGP form adjacency and exchange routing information with directly connected neighbors. IGP use broadcasts or multicasts to propagate topology changes across an AS. All IGP routers within an AS must be running the same routing protocol to handle the routing updates and maintain the same information for consistent routing operation.
- BGP does not work in the same manner as IGP. As the designers of BGP could not guarantee that an AS would run BGP on all its routers, a method had to be developed to ensure that IBGP routers could pass routing updates between them. By fully meshing all IBGP neighbors, when an update is received from an external AS, the EBGP router that is interfacing with the external AS is responsible for directly informing of all its IBGP neighbors regarding the change. IBGP neighbors that receive this update do not propagate it to any other IBGP neighbor, as they assume that the IBGP neighbor that originated the update is fully-meshed with all other IBGP neighbors and has sent the update directly to every IBGP neighbor.
- The main reason that an AS needs to fully mesh its IBGP neighbors is due to the BGP split-horizon rule that prevents routing loops or routing black holes. If the originating IBGP router is not fully-meshed with every IBGP neighbor, the IBGP neighbors that are not peering with the originating IBGP router will have different IP routing tables than the IBGP neighbors that are peering with the IBGP router that received the original BGP update from the external AS. The inconsistent routing tables can cause routing loops or routing black holes.
- TCP sessions cannot be multicast or broadcast because TCP has to ensure reliable delivery of packets to each recipient. Since TCP cannot use broadcasting, BGP cannot use it either; therefore BGP has to setup fully-meshed TCP sessions among the IBGP neighbors.



**Figure 13-6:** Partial-Mesh IBGP and Full-Mesh IBGP

- Figure 13-6A shows IBGP update behavior in a partially-meshed neighbor environment. RT2 receives a BGP update from RT1. RT2 has 2 IBGP neighbors – RT3 and RT4, but does not have an IBGP neighbor session with RT5. RT3 and RT4 are able learn about the networks that were added and withdrawn on RT1. Even if RT3 and RT4 have IBGP neighbor sessions with RT5, they assume that the AS is fully-meshed for IBGP and therefore do not propagate the update to RT5 due to the BGP split-horizon rule. Sending IBGP updates to RT5 is the responsibility of RT2, as it is the router that obtains firsthand knowledge about the networks in and beyond AS 65001. RT5 does not learn of any networks through RT2 and therefore does not use RT2 to reach any networks in AS 65001 and other ASes behind AS 65001.
- In Figure 13-6B, IBGP is fully-meshed between BGP routers in AS 65002. When an IBGP neighbor receives an update from an EBGP neighbor, the router will send the update to every IBGP neighbor in the AS. The update is sent only once to each IBGP neighbor and is not being replicated by any other IBGP neighbor.
- Each IBGP neighbor needs to know all of the other IBGP neighbors in the same AS so that it can have a complete knowledge of how to exit the AS. When all BGP routers in an AS are fully-meshed and have the same database when a consistent routing policy, they will be able to apply the same path-selection formula with the path-selection results uniform across the AS, which means no routing loops and there is a consistent policy for existing and entering the AS.

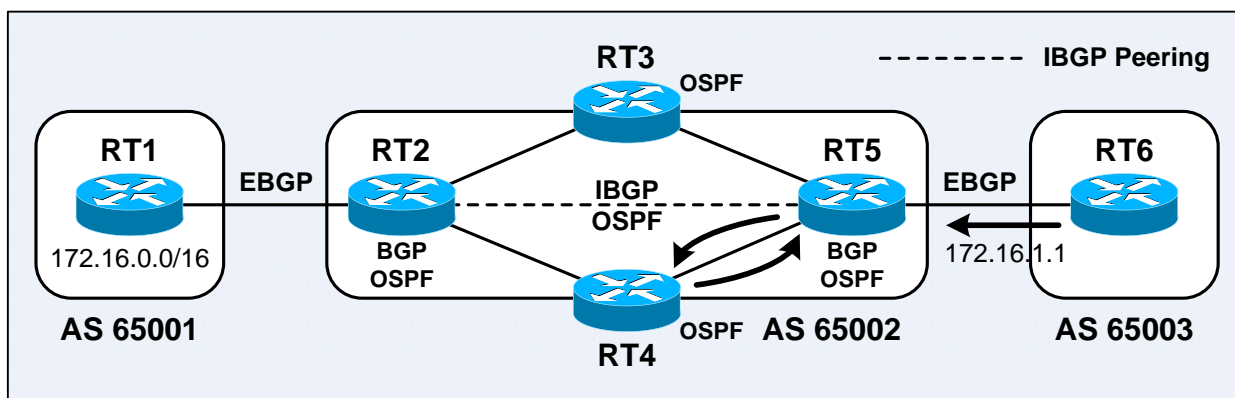


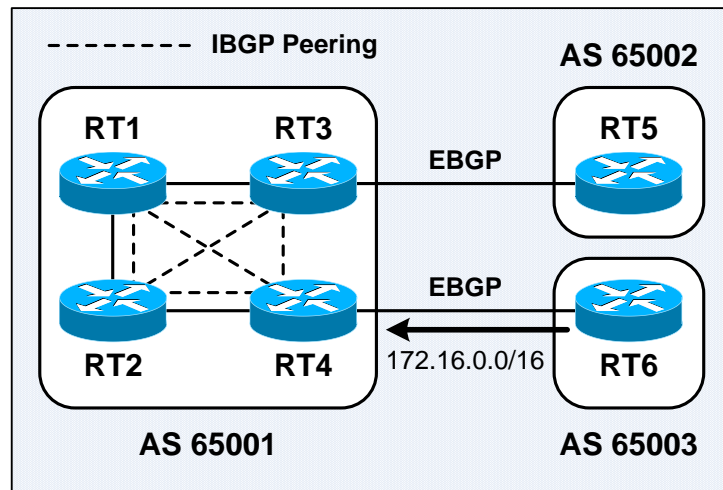
Figure 13-7: Routing Loop without Full-Mesh IBGP

- RT1, RT2, RT5, and RT6 are the only ones running BGP. An IBGP session has been established between RT2 and RT5; and EBGP sessions are established between RT1 – RT2 and RT5 – RT6. RT3 and RT4 are not running BGP. RT2, RT3, RT4, and RT5 are also running OSPF as their IGP.
- Network 172.16.0.0/16 is owned by AS 65001 and is advertised by RT1 to RT2 via EBGP. RT2 advertises it to RT5 via IBGP. RT3 and RT4 never learn about this network as it is not being redistributed into the local routing protocol – OSPF, and RT3 and RT4 are not running BGP. If RT5 advertises this network to RT6 in AS 65003, and RT6 starts forwarding packets to 172.16.0.0/16 through AS 65002, where will RT5 forwards the packets to reach RT2? If RT5 forwards packets with the destination address of 172.16.1.1 to either RT3 or RT4, they do not have an entry for 172.16.0.0/16 in their routing tables; therefore discard the packets. If RT3 and RT4 have a default route towards the exit points of the AS – RT2 and RT5, there is a high possibility that when RT5 sends a destined to 172.16.0.0/16 to RT3 or RT4, they might send it back to RT5, which forwards it again to RT3 or RT4, causing a routing loop. If BGP is fully-meshed and RT3 and RT4 are aware of network 172.16.0.0/16 from RT2, this problem does not occur.
- AS 65002 in Figure 13-7 is responsible for moving packets between AS 65001 and AS 65003, much as an ISP would. AS 65002 (and any ISP network) is a transit AS, responsible for passing packets from one AS to another. Many ASes have multiple connections to the Internet but do not use their bandwidth to transport packets of other ASes; these ASes are called **stub ASes**. Most enterprise ASes connected to the Internet are stub ASes. An ISP must be configured as a transit AS by running BGP on all of its routers and fully meshing the IBGP sessions so that packets transiting the AS can reach networks and other ASes on the other sides of the transit AS.

## BGP Synchronization

- The BGP synchronization rule states that a BGP router cannot use an IBGP-learned transit route nor advertise it to an EBGP peer; unless it is synchronized in both BGP and IGP routing tables – IGP has redistributed and propagated it across the AS and installed it in the local IP routing table. BGP running in a transit AS that is passing traffic between ASes should not advertise a route before all routers in the AS have learned about the route via IGP. The BGP synchronization rule ensures consistency information throughout an AS to avoid routing black holes within the AS, eg: advertising a destination to an EBGP neighbor when not all routers within the AS can route to the destination.

- BGP synchronization should only be used when there are routers in the AS that do not run BGP. It is safe to disable BGP synchronization only if all routers in the transit path in the AS – the path between the BGP border routers, are running BGP; or when the AS is not a transit AS. BGP synchronization is enabled by default in earlier version of Cisco IOS releases. Most current Cisco IOS releases (Cisco IOS Release 12.2(8)T and later) has disabled BGP synchronization by default, as most ISPs run BGP on all routers within their ASes.



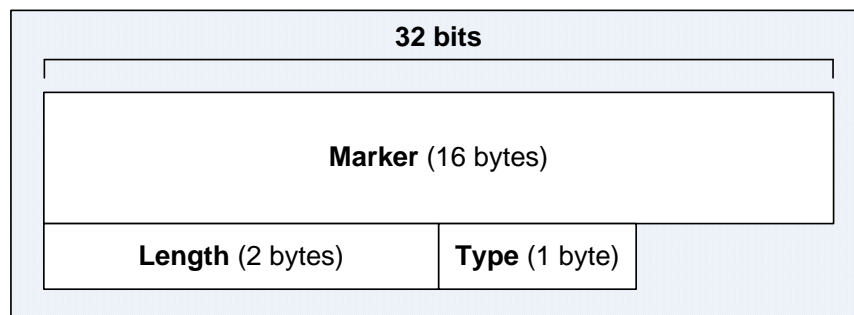
**Figure 13-8:** BGP Synchronization

- RT1, RT2, RT3, and RT4 are running BGP with each other – full-mesh IBGP. There are no matching IGP route entries for the BGP route – 172.16.0.0/16. RT3 and RT4 are not redistributing the BGP route into the IGP. RT1, RT2, RT3, and RT4 have IGP routes to the internal networks of AS 65001 but do not have routes to external networks, eg: 172.16.0.0/16.
- The following happens if synchronization is on in AS 65001:
  - RT4 advertises the route to 172.16.0.0/16 to the other routers in AS 65001 via IBGP.
  - RT4 uses the route to 172.16.0.0/16 and installs it in its IP routing table.
  - RT1, RT2, and RT3 do not use or advertise the route to 172.16.0.0/24 until they receive the matching route via an IGP. Since RT4 does not redistribute BGP routes into the IGP, these routers never use and advertise the route.
  - RT5 does not hear about 172.16.0.0/24. RT5 does not have a route to 172.16.0.0/16 and therefore unable to send traffic destined to the network.
 The solutions are **i)** redistribute the BGP routes into the IGP; **ii)** disable BGP synchronization.
- The following happens if synchronization is turned off in AS 65001:
  - RT4 advertises the route to 172.16.0.0/16 to the other routers in AS 65001 via IBGP.
  - RT1, RT2, and RT3 use and advertise the route to 172.16.0.0/16 that they receive via IBGP and install it in their IP routing tables. RT1, RT2, and RT3 must be able to reach the next-hop address for 172.16.0.0/16, via static or IGP routes.
  - RT5 hears about 172.16.0.0/16 from RT3. RT5 has a route to 172.16.0.0/16 and therefore able send traffic destined to the network.
  - When RT5 sends traffic for 172.16.0.0/16, RT1, RT2, and RT3 route the packets to RT4. The traffic flow is RT5 → RT3 → RT1 → RT2 → RT4 → RT6.
- Redistributing BGP into IGP is not scalable due to the large size of the Internet routing table. Therefore, most modern transit ASes run full-mesh IBGP and disable synchronization. Some advanced BGP configuration methods, eg: route reflection and confederation, redefine the IBGP full-mesh requirements and implementations.



## BGP Message Types

- BGP is an application layer protocol that uses TCP as its transport layer protocol – TCP Port number 179, which provides connection-oriented reliable delivery of packets. BGP assumes that its communication is reliable, and does not implement any fragmentation, retransmission, acknowledgment, sequencing, and error-recovery mechanisms as like EIGRP. BGP information is encapsulated inside TCP segments, which are then encapsulated inside IP packets. BGP is the only IP routing protocol that uses TCP as its transport layer. RIP uses UDP for its transport layer; IGRP, EIGRP, and OSPF reside directly above the IP layer; and IS-IS is resides in the network layer.
- BGP routers establish a TCP connection between them via the standard TCP three-way handshake to exchange messages to confirm the connection parameters and full routing updates. BGP routers send only changes (triggered updates) subsequently. BGP also sends periodic keepalive messages, similar to the Hello packets in EIGRP, OSPF, and IS-IS.
- EIGRP and OSPF implement a one-for-one window to ensure that update packets are explicitly acknowledged. When there are multiple packets to send, the next packet cannot be sent until an acknowledgment for the current packet is received. The acknowledgment process can be very inefficient and causes latency issues if many update packets must be exchanged over slow links. However, OSPF and EIGRP rarely have to exchange thousands of updates packets. EIGRP can hold more than 100 networks in a single EIGRP update packet; therefore 100 EIGRP update packets can hold up to 10000 networks. Most organizations do not have 10000 subnets!
- BGP, which always has more than 120'000 Internet routes to advertise, uses TCP to handle the acknowledgment process. The 16-bit TCP Window allows up to 65536 bytes to be sent without waiting for an acknowledgment. If 1000-byte packets are being sent, around 65 packets are allowed to be sent without waiting for an acknowledgment.
- With TCP sliding window, the receiver often acknowledges at halfway of the sending window, which allows any TCP applications, eg: BGP, to stream packets without having to stop and wait for acknowledgments. TCP was selected as the transport layer for BGP as TCP can transmit a large volume of data reliably. With the full Internet routing table exceeding 32MB in size and exceeding 1000 bytes of BGP changes per minute, rely upon TCP for windowing and reliability is better than implementing a BGP one-for-one windowing capability as like EIGRP and OSPF.
- BGP messages are carried within TCP segments. The minimum and maximum message sizes are 19 and 4096 octets or bytes respectively. All BGP messages share a common header. A data portion might or might not follow the header depend upon the message type.

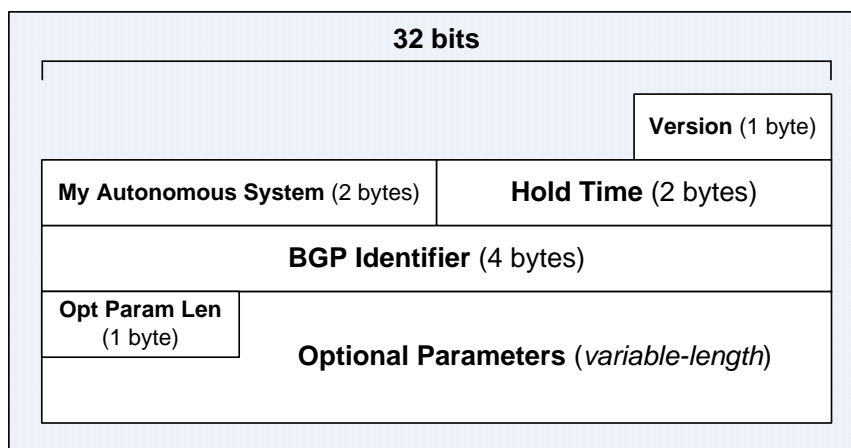


**Figure 13-9:** The BGP Message Header

- Marker is a 16-byte field that is used to detect loss of synchronization between BGP peers and to authenticate messages when authentication is supported. It is set to all 1s if the message type is Open or if the Open message does not contain authentication information; otherwise, the value of the Marker can be predicted by some computation as part of the authentication process. Length is a 2-byte field that indicates the total length of the message including the header, in octets. Type is a 1-byte field that specifies the message type.

- BGP defines the following message types:

<b>Open</b> (Type Code 1)	The 1st message sent by both peers after a TCP connection has been established. If the Open message is acceptable, a Keepalive message is sent to confirm the Open. Keepalive, Update, and Notification messages can be exchanged only after the Open has been confirmed and the BGP connection is established.
<b>Update</b> (Type Code 2)	BGP peers exchange full BGP routing tables upon the initial TCP connection. Incremental updates upon the changes in the routing table are sent subsequently.
<b>Notification</b> (Type Code 3)	Sent when an error or special condition is detected. The BGP connection is terminated immediately upon a Notification message is sent. A Notification message includes an Error Code, an Error Subcode, and the Data or reason for the error.
<b>Keepalive</b> (Type Code 4)	Consists of only the 19-byte BGP message header with no additional data. Maintains the BGP connection between 2 BGP peers. Exchanged on a period of one-third (1/3) of the hold time, but not less than 1 second; 60 seconds by default. Keepalive messages are not sent if the negotiated hold time for the BGP session is 0.

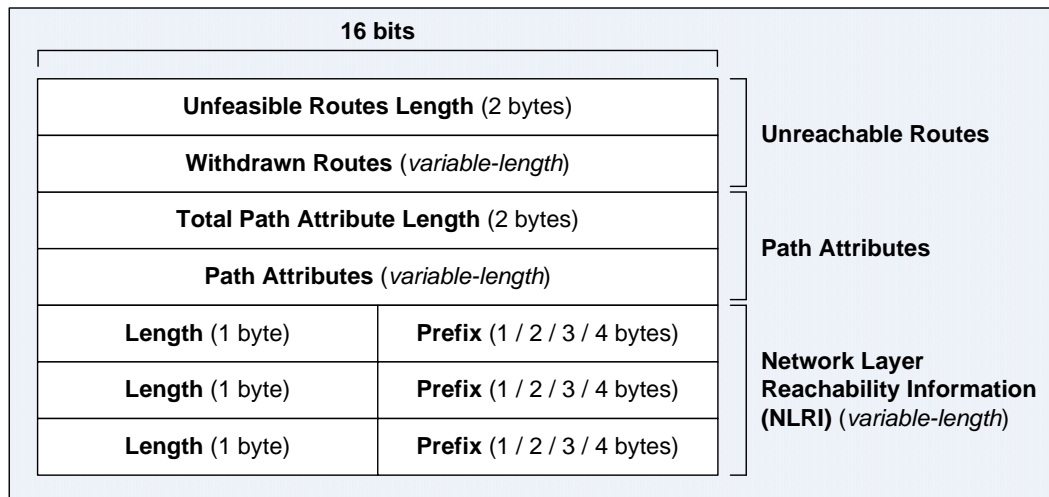


**Figure 13-10: The BGP Open Message Format**

- The BGP Open message contains the following fields and information:
  - **Version.** A 1-byte field that indicates the BGP version number running on the originator. The highest common version that both routers negotiated and support will be used. Most current BGP implementations use the current version – BGP-4.
  - **My Autonomous System.** A 2-byte field that indicates the AS number of the originator. A BGP peer uses this information to determine whether the BGP session is EBGP or IBGP and will terminate the BGP session if it is not the expected AS number.
  - **Hold Time.** A 2-byte field that indicates the number of seconds proposed by the originator for the hold time of the BGP session – the period of time that can elapse before the receiver must receive either a Keepalive or Update message from the originator. The receiver of the Open message calculates the hold timer value to use by comparing the Hold Time field specified in the Open message and its configured hold timer value; and accepts the smaller value or rejects the connection. The hold time must be either 0 or at least 3 seconds. The default hold time is 180 seconds.



- **BGP Identifier.** A 4-byte field that indicates the Router ID of the originator. The BGP identifier is an IP address assigned to a BGP router and is determined upon the startup of a BGP routing process. The BGP Router ID is chosen the same way as the OSPF Router ID is chosen. The BGP Router ID can be statically configured to override the automatic selection.
- **Optional Parameters Length.** A 1-byte field that indicates the total length of the following Optional Parameters field, in octets. A value of zero indicates that there is no Optional Parameters field is included in the Open message.
- **Optional Parameters.** A variable-length field that contains a list of optional parameters. Each parameter is specified by a 1-byte Type field, a 1-byte Length field, and a variable-length Value field that contains the parameter value itself. This field is used to advertise the support for optional capabilities, eg: multiprotocol extensions, route refresh, etc.

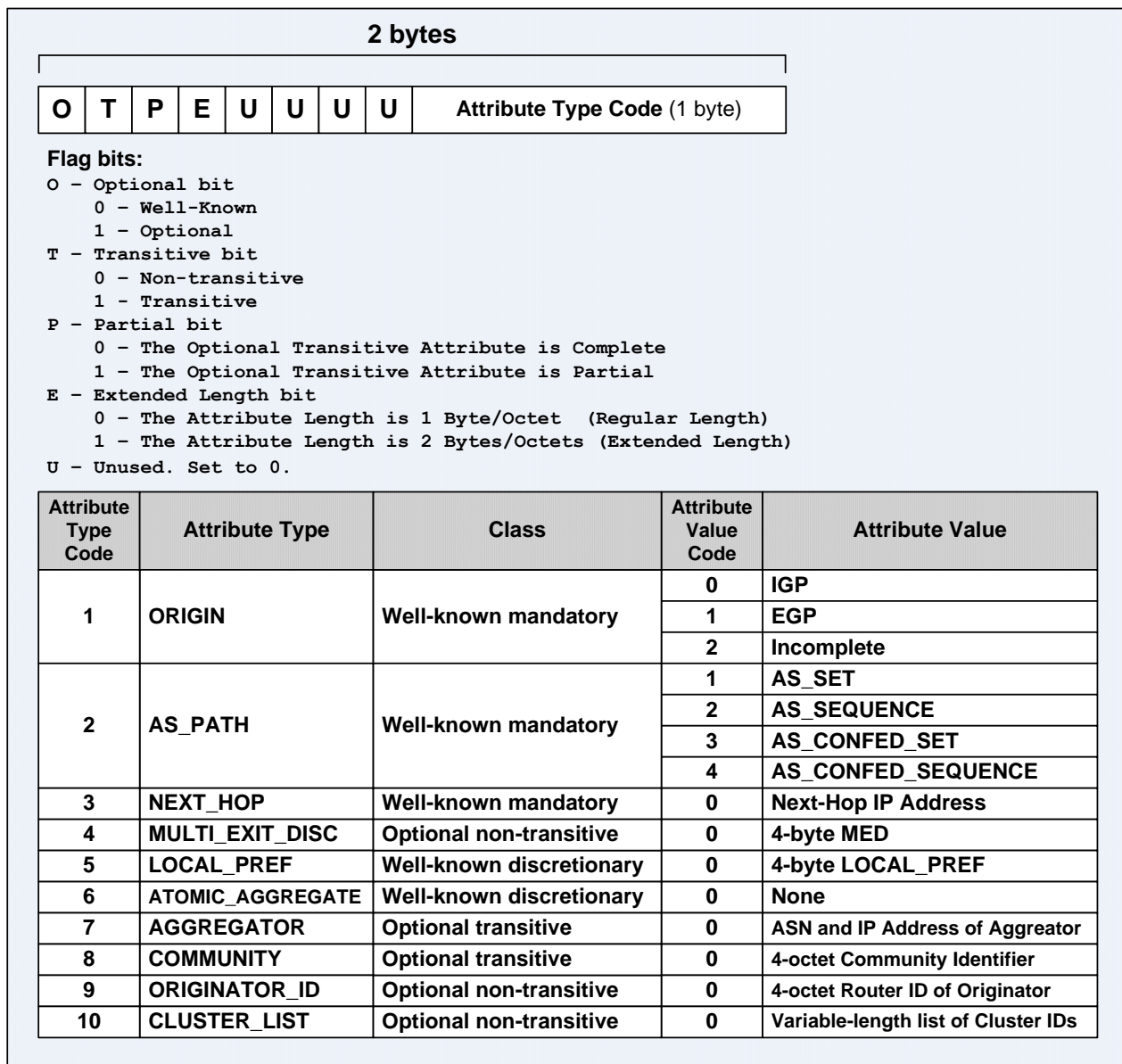


**Figure 13-11: The BGP Update Message Format**

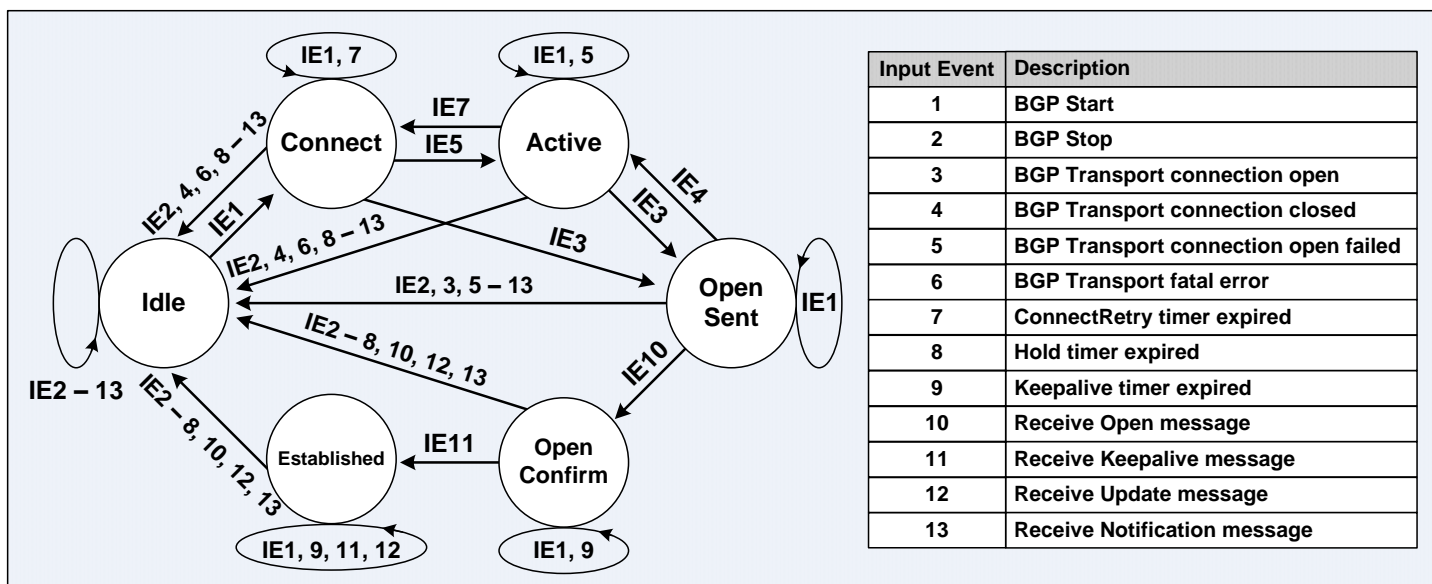
- A BGP Update message is able to advertise a single feasible route, or withdrawn multiple unfeasible routes, or both. An Update message contains information about a **single path** only; multiple paths require multiple Update messages. All the path attributes in the Update message refer to a particular path, and the prefixes that can be reached through the specified path – **update packing**. The BGP Update message contains the following fields and information:
  - **Unfeasible Routes Length.** A 2-byte field that indicates the total length of the following Withdrawn Routes field, in octets. A value of zero indicates that no routes are being withdrawn and there is no Withdrawn Routes field is included in the Update message.
  - **Withdrawn Routes.** A variable-length field that contains a list of unreachable routes that are to be withdrawn from service, if any. Each route in the list is described with a (Length – Prefix) tuple. If the Length part of the tuple is 0, the Prefix matches all routes.
  - **Total Path Attribute Length.** A 2-byte field that indicates the total length of the following Path Attributes field, in octets. A value of zero indicates that no path attributes and NLRI are included in the Update message.
  - **Path Attributes.** A variable-length field that lists the attributes associated with the NLRI in the following field. Each path attribute is a variable-length triplet of (Attribute Type, Attribute Length, Attribute Value). The 2-byte Attribute Type field of the triplet consists of the 4 Attribute Flag bits, 4 unused bits, and a 1-byte Attribute Type Code field.
  - **Network Layer Reachability Information (NLRI).** A variable-length field that contains a list of IP prefixes that can be reached via this path using the (Length – Prefix) tuples. A Length value of 0 indicates a prefix that matches all IP prefixes.

**Note:** BGP is more towards a reachability protocol than a routing protocol.

**Note:** If the path attributes associated with a route has changed, or a new path for the same prefix has been selected, a withdrawn route is not required; an Update message with the new attributes and matching NLRI will be advertised instead – **implicit withdraw**.



**Figure 13-12:** The Attribute Types and Associated Attribute Values of BGP Path Attributes



**Figure 13-13:** The BGP Finite State Machine

## The BGP Finite State Machine

- The stages of BGP connection establishment and maintenance can be described in terms of a finite state machine. Figure 13-13 shows the complete BGP finite state machine and the input events that can cause a state transition.

- Below describes the 6 BGP neighbor relationship states:

<b>Idle</b>	BGP always begins in this state, in which it refuses all incoming connections. When a BGP Start event occurs, the BGP process initializes all BGP resources, starts the ConnectRetry timer, initializes a TCP connection to the neighbor, listens for a TCP initialization from the neighbor, and changes state to Connect.
<b>Connect</b>	The BGP process is waiting for its own TCP connection attempt to complete. If the connection is successful, the BGP process resets the ConnectRetry timer, completes initialization, sends an Open message to the neighbor, and transitions to the Open Sent state. If the TCP connection is unsuccessful, the BGP process continues to listen for incoming TCP connections to be initiated by the neighbors, resets the ConnectRetry timer, and transitions to the Active state.
<b>Active</b>	The BGP process is trying to initiate a TCP connection with the neighbor. If the connection is successful, the BGP process resets the ConnectRetry timer, completes initialization, sends an Open message to the neighbor, and transitions to the Open Sent state; else if the ConnectRetry timer expires, BGP resets the ConnectRetry timer and transitions back to the Connect state.
<b>Open Sent</b>	An Open message has been sent, the BGP process is waiting for an Open message from its neighbor. When an Open is received, all its fields are checked. If an error exists, sends a Notification message and transitions to the Idle state. If no errors exist in the received Open message, sends a Keepalive message and set the Keepalive timer. Negotiate the Hold time, the smaller value is agreed upon. Do not start the Hold and Keepalive timers if the negotiated Hold time is 0. Determine the peer connection to be either internal or external based on the AS number on the peer and transitions to the Open Confirm state.
<b>Open Confirm</b>	The Open message from the neighbor has been received. Waits for the initial Keepalive message or a Notification message. If a Keepalive message is received, transitions to the Established state. If a Notification message or a TCP Reset is received, transitions to the Idle state.
<b>Established</b>	The BGP peer connection is fully established and the peers can exchange Update, Keepalive, and Notification messages. If an Update or Keepalive message is received, resets the Hold timer (if the negotiated hold time is non-zero). If a Notification message is received, transitions to the Idle state.

- The default ConnectRetry timer is 120 seconds and it cannot be changed nor tuned. The BGP process checks whether the passive TCP session is established only after the timer is expired. If the passive TCP session is not established, the BGP process initiates a new active TCP connection attempt to the remote BGP peer. During the idle period of the ConnectRetry timer, the remote BGP peer can establish a BGP session to the local BGP router.

## BGP Attributes

- BGP routers send Update messages about destination networks. BGP Update messages contain information about BGP metrics, which are called **path attributes** – the characteristics of a route. The following are some terms defining how BGP attributes are implemented:
  - An attribute is either well-known or optional, mandatory or discretionary, and transitive or non-transitive. An attribute may also be partial.
  - Not all combinations of these characteristics are valid.  
BGP path attributes fall into one of the following 4 categories:
    - **Well-known mandatory**
    - **Well-known discretionary**
    - **Optional transitive**
    - **Optional non-transitive**
  - Only Optional Transitive attributes may be marked as partial.
- A **well-known** attribute is an attribute that must be recognized by all BGP implementations. These attributes are propagated throughout BGP routers. Well-known attributes are either **mandatory**, which must be included in all BGP Update messages; or are **discretionary**, which may or may not be sent in a specific Update message. BGP routers must recognize and act upon the information in discretionary attributes when they are present in the Update messages. Well-known attributes are always transitive, therefore the Transitive bit is always set to 1.
- Attributes that are not well-known are called **optional** attributes, which are either transitive or non-transitive. An optional attribute does not need to be supported by all BGP implementations; it could be a private attribute. If it is supported, it might be propagated throughout BGP routers. An **optional transitive** attribute that is not supported by a BGP router should be propagated to other BGP routers remains unchanged (preserved), and the attribute is marked as partial <sup>[1]</sup>; an **optional non-transitive** attribute is dropped by a BGP router that does not recognize and support it, and the attribute will not be propagated to other BGP routers. New optional transitive attributes may be attached to the path by the originator or by any other AS in the path.  
[1] A router that doesn't understand and relay an optional transitive attribute sets the Partial bit to inform downstream routers that the attribute is not being processed as desired at all previous hops.
- BGP has defined the following path attributes:
  - **Well-known mandatory** attributes:
    - ORIGIN (Type Code 1)
    - AS\_PATH (Type Code 2)
    - NEXT\_HOP (Type Code 3)
  - **Well-known discretionary** attributes:
    - LOCAL\_PREF (Type Code 5)
    - ATOMIC\_AGGREGATE (Type Code 6)
  - **Optional transitive** attributes:
    - AGGREGATOR (Type Code 7)
    - COMMUNITY (Type Code 8; originally Cisco-specific, now standardized in RFC 1997)
  - **Optional non-transitive** attributes:
    - MULTI\_EXIT\_DISC – MED (Type Code 4)
    - ORIGINATOR\_ID (Type Code 9, Cisco-specific)
    - CLUSTER\_LIST (Type Code 10, Cisco-specific)

The following sections discuss all BGP attributes above, except the ATOMIC\_AGGREGATE and AGGREGATOR attributes, which will be discussed in **Chapter 15 – BGP Route Summarization**.
- Cisco has also defined the **administrative weight** BGP attribute – WEIGHT, which is a parameter that applies only upon paths within an individual router and is not communicated to other routers.

## The ORIGIN Attribute

- The BGP ORIGIN attribute is a **well-known mandatory** attribute. It specifies the origin of a routing update or path information. When BGP has multiple routes, it uses the ORIGIN attribute as a factor in determining the preferred route. The ORIGIN can be one of the following 3 values:
  - **IGP.** The NLRI was learned via an interior routing protocol in the originating AS. BGP routes that are originated within the AS and being advertised using the **network** statements are assigned an origin of IGP. It is indicated with an **i** in the BGP table.
  - **EGP.** The NLRI was learned via EGP – Exterior Gateway Protocol. It is indicated with an **e** in the BGP table. EGP is considered a historic routing protocol and is not supported on the current Internet as it performs only classful routing and does not support CIDR.
  - **Incomplete.** The origin of route is unknown or the NLRI was learned via some other means. An incomplete origin is indicated with an **?** in the BGP table. Incomplete does not imply that the route is faulty, but rather that the information for determining the origin of the route is incomplete. This usually occurs when a route is being redistributed into BGP, in which there is no way to determine the original source of the route.

## The AS\_PATH Attribute

- The BGP AS\_PATH attribute is a **well-known mandatory** attribute. It lists the ASNs of the inter-AS path to reach a destination, with the ASNs of the most recent AS and originating AS at the beginning and ending of the list. When a BGP router advertises a route to an EBGP peer, it **prepends** (put at the beginning of the list) its own ASN to the AS\_PATH for the route. No ASN is added when the route is being advertised between IBGP peers within the same AS.

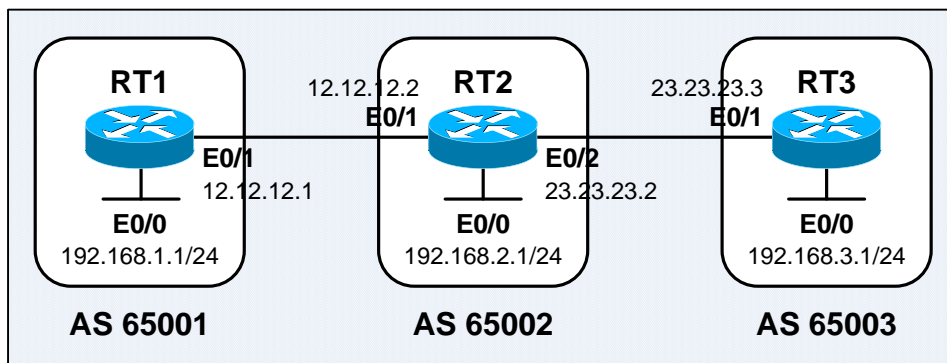


Figure 13-14: BGP AS\_PATH Attribute

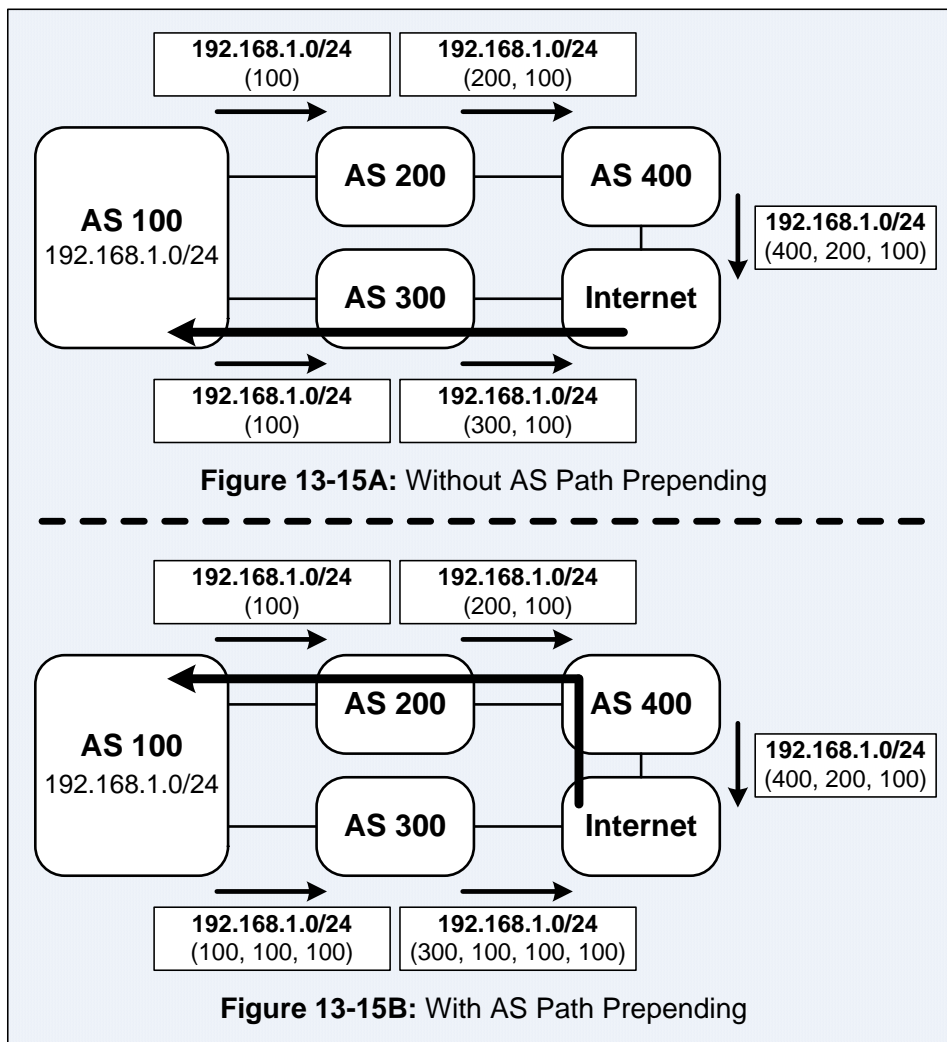
- RT1 advertises 192.168.1.0/24 in AS 65001. When the route traverses through AS 65002, RT2 prepends its own ASN to it. When the route reaches RT3, it has 2 ASNs attached to it. From the perspective of RT3, the path to reach 192.168.1.0/24 is (65002, 65001).

```

RT3#sh ip bgp
BGP table version is 4, local router ID is 192.168.3.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network        Next Hop           Metric LocPrf Weight Path
*> 192.168.1.0    23.23.23.2         0         0   65002 65001 i
*> 192.168.2.0    23.23.23.2         0         0   65002 i
*> 192.168.3.0    0.0.0.0            0         0  32768 i
RT3#
  
```

- The same concept applies for 192.168.2.0/24 and 192.168.3.0/24. The path to 192.168.3.0/24 from RT1 is (65002, 65003); RT2 traverses the path (65001) to reach 192.168.1.0/24 and the path (65003) to reach 192.168.3.0/24.
- BGP routers use the AS\_PATH attribute for **loop avoidance**. A BGP router does not accept a route received from an EBGW peer in which the AS\_PATH attribute includes its own ASN. ASNs are prepended only when routes are being advertised between EBGW peers. The AS\_PATH attribute remains when routes are being advertised between IBGW peers.
- BGP routes are considered most desirable when they traverse the least possible number of ASes. As when prefixes are being advertised from ASes to ASes, the AS\_PATH gets longer and longer. The shorter the AS\_PATH, the more desirable it is. Usually, having multiple instances of the same ASN in the list does not make sense and defeat the purpose of the AS\_PATH attribute. However, below shows a scenario that adding multiple instances of an ASN to the AS\_PATH is useful.



- Remember that outgoing BGP route advertisements directly influence incoming traffic. Normally, the traffic originated from the Internet and destined to 192.168.1.0/24 passes through AS 300 as the AS\_PATH of the route is shorter. But what if the link via AS 200 is the preferred path for incoming traffic for AS 100? Due to the bandwidth of the links along the (400, 200, 100) path is much higher than the bandwidth of the links along the (300, 100) path; or AS 200 is the primary provider and AS 300 is only the backup provider – outgoing traffic is sent via AS 200, and therefore it is desired to receive incoming traffic through the same path.

- AS 100 can influence its incoming traffic by changing the AS\_PATH of its advertised route. By adding multiple instances of its own ASN into the AS\_PATH of route advertised to AS 300, AS 100 can make routers in the Internet think that the (400, 200, 100) path is the shorter path. The procedure of adding extra ASNs into the AS\_PATH attribute is called **AS Path Prepending**. ASNs are inserted at the beginning of the AS\_PATH; just after the ASN of the originating router.

### The NEXT\_HOP Attribute

- The BGP NEXT\_HOP attribute is a **well-known mandatory** attribute. It specifies the next-hop IP address to be used to reach a destination. BGP, like IGPs, is a hop-by-hop routing protocol. However, unlike IGPs, BGP routes AS-by-AS, not router-by-router; and therefore the default next-hop always towards the next AS. The next-hop address for a network from another AS is the IP address of an entry point of the next AS along the path to that destination network.
- The IP address of the BGP NEXT\_HOP attribute is not always the address of a neighboring router. The following rules apply when determining the IP address in the BGP NEXT\_HOP attribute:
  - If the advertising router and receiving router are in different ASes (EBGP peers), the NEXT\_HOP is the **BGP Router ID of the advertising EBGP peer**.
  - If the advertising router and receiving router are in the same AS (IBGP peers), and the NLRI of the update refers to a destination within the same AS, the NEXT\_HOP is the **BGP Router ID of the advertising IBGP peer**. It is very important to understand the dependency of IBGP upon the IGP for the recursive route lookup process.
  - If the advertising router and the receiving router are in the same AS (IBGP peers), and the NLRI of the update refers to a destination in a different AS, the NEXT\_HOP is the **BGP Router ID of the advertising EBGP peer**.

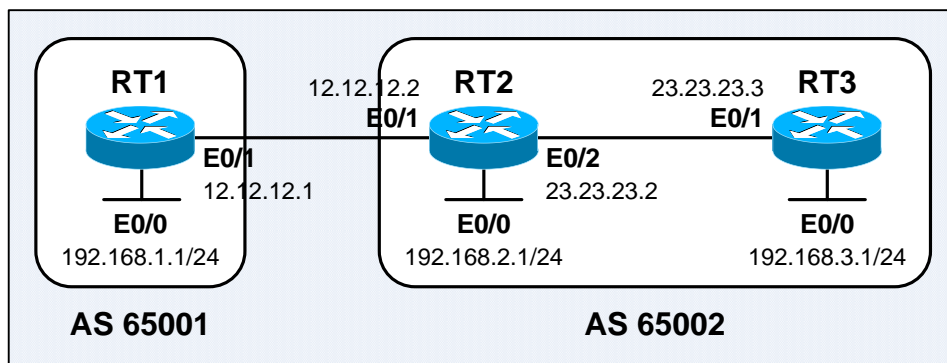


Figure 13-16: BGP NEXT\_HOP Attribute

- For EBGP, the NEXT\_HOP is the IP address of the EBGP neighbor that advertised the update. RT1 advertises 192.168.1.0/24 to RT2 with a NEXT\_HOP attribute of 12.12.12.1; and RT2 advertises 192.168.2.0/24 to RT1 with a NEXT\_HOP attribute of 12.12.12.2. Therefore, RT1 uses 12.12.12.2 as the next-hop address to reach 192.168.2.0/24, and RT2 uses 12.12.12.1 as the next-hop address to reach 192.168.1.0/24.
- For IBGP, the NEXT\_HOP attribute advertised by EBGP is preserved when carried into IBGP. Due to this IBGP NEXT\_HOP rule, RT2 advertises 192.168.1.0/24 to its IBGP peer RT3 with a NEXT\_HOP of 12.12.12.1, the IP address of RT1 E0/1; not 23.23.23.2 as we might expect. Therefore, it is very important to make sure that RT3 is able to reach the 12.12.12.0/24 subnet, either via a static route or an IGP; otherwise, it will drop packets destined to 192.168.1.0/24, as it is unable to get to the next-hop address for the destination. In fact, without an IGP running on RT2 and RT3 for RT2 to advertise the 12.12.12.0/24 to RT3, the route to 192.168.1.0/24 does exist in the BGP table but not in the IP routing table on RT3.



- IBGP routers often perform **recursive route lookup** to find out how to reach the BGP next-hop addresses using IGP entries in the IP routing table. RT3 learns a BGP update about network 192.168.1.0/24 from the route source 23.23.23.2 – RT2, with a next-hop of 12.12.12.1 – RT1. RT3 installs the route to 192.168.1.0/24 in the routing table with a next-hop of 12.12.12.1. Assuming that RT2 advertises network 12.12.12.0/24 using its IGP to RT3, RT3 then installs the route in its IP routing table with a next-hop of 23.23.23.2. Remember that IGP uses the source IP address of a routing update as the next-hop address; while BGP uses the NEXT\_HOP attribute to indicate the next-hop address of a network. When RT3 forwards a packet to 192.168.1.1, it lookups the network in the routing table and found a BGP route with a next-hop of 12.12.12.1. RT3 then performs a recursive route lookup for the route to 12.12.12.0/24 and found an IGP route to the 12.12.12.0/24 network in the routing table with a next-hop of 23.23.23.2. Eventually RT3 forwards the packet destined to 192.168.1.1 to 23.23.23.2 – RT2.
- Instead of advertising the DMZ link (the shared network between ASes) through an IGP, the **next-hop-self** configuration can be implemented on AS border BGP routers to override the NEXT\_HOP attribute with the IP addresses of the AS border routers. IBGP peers within the AS would then forward packets destined to external networks through the AS border routers.
- When BGP routers reside on a common subnet, they would set the NEXT\_HOP to an appropriate address upon route advertisement to avoid inserting additional hops into the forwarding path. This feature is known as **third-party next-hop** and does not require any special configuration; it is often utilized in the route server setups in the ISP environments. Since it is impossible for a single router from large backbone ISPs to peer with every router from different backbone ASes at the major **Internet Exchange Point (IXP)** public peering points; route servers and route server clients are setup to reside on a shared L2 broadcast medium with a common subnet, eg: Ethernet; route server clients from different ASes only require a single EBGP peering with the route server, the route server then receives the routing information from its clients and relays back to its clients. Eventually the route server clients – BGP routers from different ASes forward packets destined to different ASes using the appropriate next-hop addresses. Additionally, the BGP routers from an AS may peer with BGP routers in different ASes and then exchange external routes via IBGP.

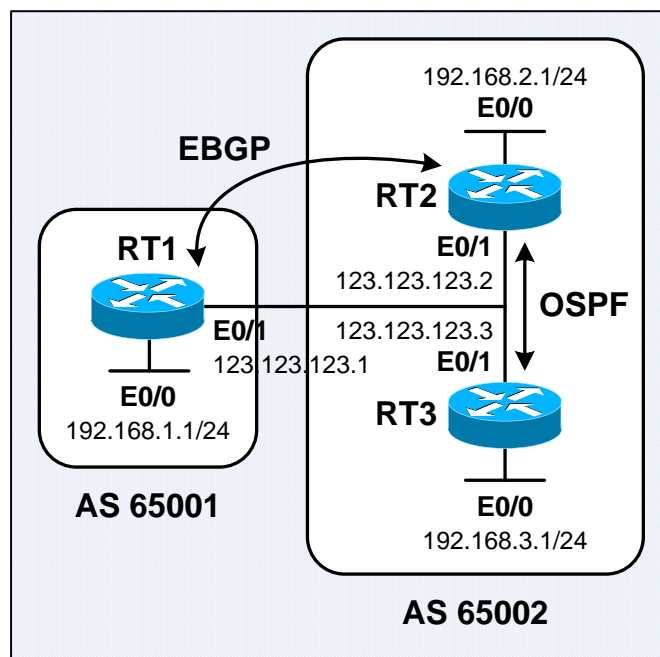


Figure 13-17: BGP NEXT\_HOP Attribute over Multi-Access Network



- RT1 and RT2 are EBGP peers; RT2 and RT3 are OSPF neighbors in AS 65002. RT2 can reach 192.168.3.0/24 via 123.123.123.3. RT2 sets 123.123.123.3 instead of its own IP address 123.123.123.2 as the NEXT\_HOP when it advertises the route to RT1 via EBGP. As the routers reside on a common subnet, it is more efficient for RT1 to use RT3 as the next-hop to reach 192.168.3.0/24, rather than having an extra hop through RT2.
- However, problems might occur when routers reside on a non-broadcast multi-access (NBMA) medium. For example, RT1, RT2, and RT3 in the figure below are connected via Frame Relay. RT2 can reach 192.168.3.0/24 via 123.123.123.3. RT2 sets 123.123.123.3 instead of its own IP address 123.123.123.2 as the NEXT\_HOP when it advertises the route to RT1 via EBGP. Routing between 192.168.1.0/24 and 192.168.3.0/24 fails if RT1 and RT3 are unable to communicate directly, eg: missing Frame Relay static mapping configuration.

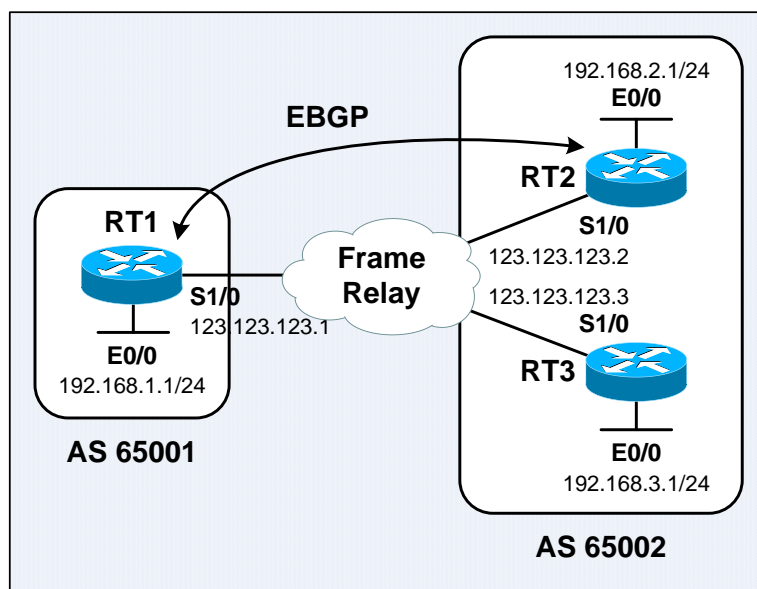


Figure 13-18: BGP NEXT\_HOP Attribute over NBMA Medium

- The solution to this problem is implements the **next-hop-self** option on RT2 to override the third-party next-hop feature and advertise itself as the next-hop for routes that advertised to RT1.

### The LOCAL\_PREF Attribute

- The BGP Local Preference (LOCAL\_PREF) attribute is a **well-known discretionary** attribute. It directs the routers within an AS towards the preferred path to exit the AS for a particular route. Local preference is an attribute that is configured and exchanged only between IBGP peers. The path with the **highest** local preference is preferred. The default local preference value is 100.

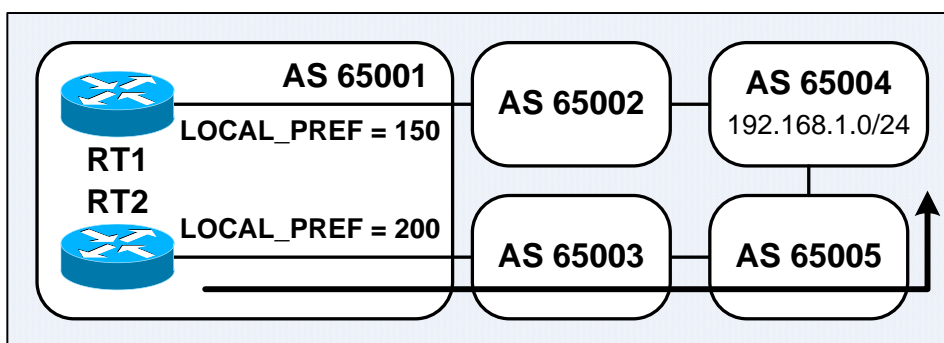


Figure 13-19: BGP LOCAL\_PREF Attribute

- RT1 and RT2 are IBGP peers. AS 65001 can reach 192.168.1.0/24 via multiple paths. The local preferences for 192.168.1.0/24 are set to 150 and 200 on RT1 and RT2 respectively. The local preference information is exchanged within AS 65001; all traffic in AS 65001 destined to 192.168.1.0/24 is forwarded through RT2 as the exit point for AS 65001.

### The MULTI\_EXIT\_DISC (MED) Attribute

- The BGP Multiple Exit Discriminator (MULTI\_EXIT\_DISC, MED) attribute is an **optional non-transitive** attribute. It is also called the **metric** in Cisco IOS. The MED attribute is known as the INTER\_AS attribute in BGP-2 and BGP-3.
- The LOCAL\_PREF attribute affects the outbound traffic leaving an AS; while the MED attribute influences inbound traffic entering an AS. The MED attribute informs EBGP peers upon the preferred path to enter an AS when the AS has multiple ingress points.  
**Note:** The MED is used as a **suggestion** to an external AS regarding the preferred entry point into the local AS, because the external AS that is receiving the MED might be using other BGP attributes for route selection.
- The **lowest** MED value is preferred, as MED is considered a metric; the lowest metric means the lowest distance, and therefore is being preferred.
- Unlike local preference, the MED is carried in EBGP updates and exchanged between ASes. MEDs are carried into an AS and used there, but are not being passed beyond the receiving AS, which means that MEDs are used only to influence traffic between 2 directly connected ASes; AS path prepending must be used to influence route preferences beyond the neighboring AS.

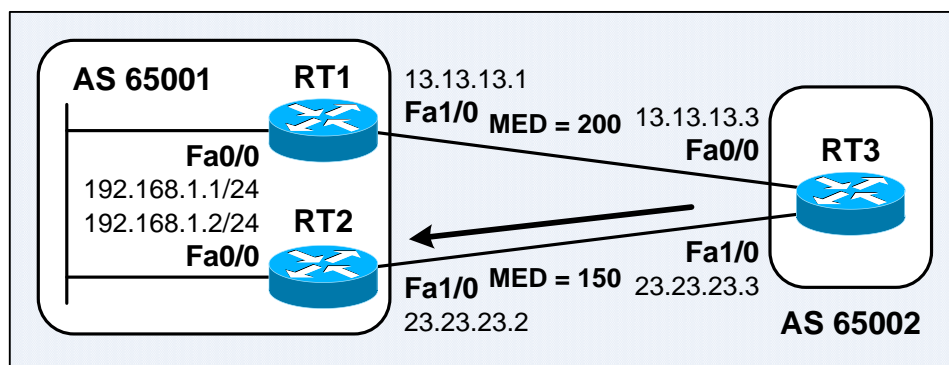


Figure 13-20: BGP MULTI\_EXIT\_DISC (MED) Attribute

- RT1 and RT2 set the MED attribute to 200 and 150 respectively. When RT3 receives the route to 192.168.1.0/24 from RT1 and RT2, it selects RT2 as the best next-hop to enter into AS 65001.
- Note that IBGP is being used between routers within an AS to exchange MEDs between IBGP peers for them to notice and select the preferred path towards the neighboring AS.
- By default, MEDs are compared only for paths through the EBGP peers in the same AS; MEDs are not compared for paths to the same destination that received from different ASes. The **bgp always-compare-med** BGP router subcommand configures a BGP router to compare the MEDs for the paths through EBGP peers from different ASes.

## The COMMUNITY Attribute

- The BGP COMMUNITY attribute is an **optional transitive** attribute of variable length. It is designed to simplify routing policy enforcements. Originally a Cisco-specific attribute, it is now standardized in RFC 1997 – BGP Communities Attribute.
- A community identifies a group of prefixes that share some common properties. BGP communities provide a way to filter incoming and outgoing routes. A BGP router can tag routes using communities, and allow other BGP routers to make decisions based upon the tag. Any BGP router can tag routes in both incoming and outgoing updates; even upon redistribution. A BGP router may also modify the COMMUNITY in a receive route according to the local policy.
- Ex: An ISP may assign a particular COMMUNITY attribute upon all its customer routes. The ISP can then set the LOCAL\_PREF and MED attributes based on the COMMUNITY value rather than on each individual prefix. This significantly simplifies the configuration of the routers.
- When a router receives updates with communities but does not understand the concept of communities, it can just forward the updates to other routers. However, if the router understands them, it must be configured to propagate them; otherwise, communities are dropped by default. Communities are not restricted within an AS; they have no boundaries and can be propagated across ASes.
- The BGP COMMUNITY attribute is a set of 4-byte (32-bit) values. RFC 1997 specifies the format as AA:NN, in which AA as the AS and NN as the administratively defined identifier. Communities are shown as numeric numbers by default. The **ip bgp-community new-format** global configuration command changes the Cisco default format to the RFC 1997 format. The best practice dictates that the AA should be the ASN of the network defining the community.
- Ex: A route from AS 123 has a COMMUNITY identifier of 321. The COMMUNITY attribute in the AA:NN format is 123:321 and is represented in hex as a concatenation of the 2 numbers – 0x007B0141, where 123 = 0x007B and 321 = 0x0141. The RFCs use the hex representation, but Cisco IOS represent COMMUNITY attribute values in decimal. In this case, 123:321 is represented as 8061249, the decimal equivalent of 0x007B0141.
- The COMMUNITY attribute values from **0** (0x00000000) through **65'535** (0x0000FFFF) and from **4'294'901'760** (0xFFFF0000) through **4'294'967'295** (0xFFFFFFFF) are reserved. Out of the reserved ranges, RFC 1997 has defined the following well-known communities that have global significance:
  - **INTERNET** (0x00000000). All routes belong to this community by default. Received routes belong to this community can be advertised freely.
  - **NO\_EXPORT** (0xFFFFF001 or 4'294'967'041). Received routes carrying this community attribute value cannot be advertised to EBGP peers – BGP speakers outside of the local AS; or outside a BGP confederation if a BGP confederation is configured.
  - **NO\_ADVERTISE** (0xFFFFF002 or 4'294'967'042). Received routes carrying this community attribute value cannot be advertised at all, to either EBGP or IBGP peers.
  - **NO\_EXPORT\_SUBCONFED** or **LOCAL\_AS** (0xFFFFF003 or 4'294'967'043). Received routes carrying this community attribute value cannot be advertised to EBGP peers, including peers in other member autonomous systems within a BGP confederation.
- A prefix can be a member of more than one community, hence have multiple community attributes. A BGP router can act based on one, some, or all of the attributes; it has the option of adding and/or modifying community attributes before propagating routes on to other internal or external peers.

## The ORIGINATOR\_ID and CLUSTER\_LIST Attributes

- The ORIGINATOR\_ID and CLUSTER\_LIST attributes are **optional non-transitive** attributes. They are used to detect and prevent routing loops when implementing route reflection.
- The 4-byte ORIGINATOR\_ID is created by a route reflector to indicate the BGP Router ID of the originating router in the local AS. When the originator sees its own Router ID as the ORIGINATOR\_ID of a received path, it knows that a loop has occurred, and ignores the route. **Note:** The originator is not necessary the route reflector! It is actually the AS edge BGP router that received the EBGP route and advertised route into the local AS through IBGP.
- A **cluster** identifies the routers involved in route reflection; and a **Cluster ID** identifies a cluster. The CLUSTER\_LIST lists a sequence of Cluster IDs which indicates the clusters that an update has passed through – the route reflection path. When a route reflector sees its local cluster ID in the CLUSTER\_LIST of a received path, it knows that a loop has occurred, and ignores the route.

## The WEIGHT Attribute

- The administrative weight attribute is a Cisco-proprietary parameter that is assigned upon paths for the path-selection process. The weight values for different paths are configured locally and effective on a router on a **per-neighbor** or **per-route** basis and are not propagated to other peers. Note that caution must be taken when manipulating weight as it only impacts path selection on the local router and hence may result in routing loops.
- The WEIGHT attribute value ranges from 0 to 65535 (inclusive). Paths that are originated by the local router have a weight of 32768 by default, and all other paths have a weight of 0 by default. Path with the **highest** weight are preferred when there are multiple paths to the same destination.
- The LOCAL\_PREF attribute is used when there are multiple routers provide multiple exit points. The WEIGHT attribute is used when a single router has multiple exit points out an AS.

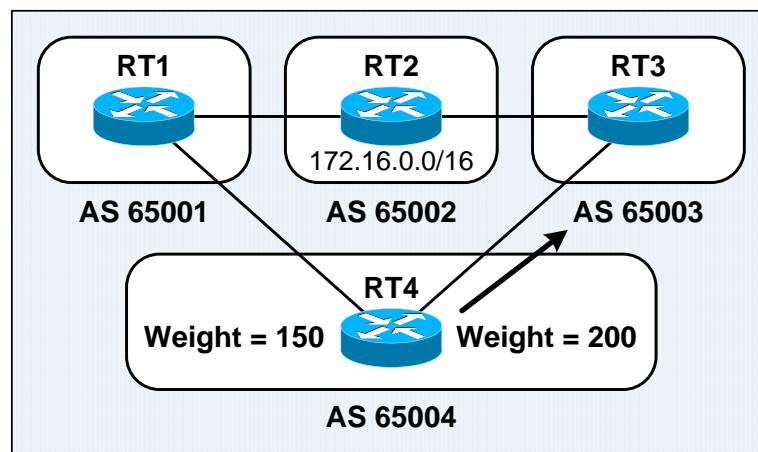


Figure 13-21: BGP WEIGHT Attribute

- RT1 and RT3 learn about 172.16.0.0/16 from AS 65002 and propagate the update to RT4. RT4 has 2 paths to reach 172.16.0.0/24 and must decide which preferred path. RT4 sets the weight for updates coming from RT1 and RT3 to 150 and 200 respectively. RT4 will then use RT3 as the next-hop to reach 172.16.0.0/16.

## The BGP Best Path Selection Algorithm

- BGP routers often receive multiple paths to the same destination. The BGP best path selection algorithm uses rules that extend far beyond than just choosing the route with the lowest metric when deciding the best path to be installed in the IP routing table for packet forwarding.
- A path must first be considered with the following criteria before it is considered valid as a candidate for the best path:
  - If BGP synchronization is enabled, an IBGP-learned path is considered as a candidate only when the IP routing table contains the route. If the matching route is learned from an OSPF neighbor, its OSPF Router ID must match the BGP Router ID of the IBGP peer. The **show ip bgp** {prefix [mask]} EXEC command is able to display paths that marked as "not synchronized". BGP synchronization is disabled by default in Cisco IOS Software Release 12.2(8)T and later.

```
RT1#sh ip bgp 172.16.0.0
BGP routing table entry for 172.16.0.0/16, version 0
Paths: (1 available, no best path)
  Not advertised to any peer
    65003
      6.6.6.6 (metric 20) from 4.4.4.4 (4.4.4.4)
        Origin IGP, metric 0, localpref 100, valid, internal, not synchronized
RT1#
```

- The next-hop address associated with the path is reachable. There is often an IGP route destined to the next-hop address associated with the path exists in the IP routing table. Next-hop addresses that are only reachable via a default route or another BGP route are not considered valid. Invalid paths due to unreachable next-hop addresses are marked as inaccessible in the output of the **show ip bgp** {prefix [mask]} EXEC command.

```
RT3#sh ip bgp 192.168.1.0
BGP routing table entry for 192.168.1.0/24, version 0
Paths: (1 available, no best path)
  Not advertised to any peer
    65001
      12.12.12.1 (inaccessible) from 23.23.23.2 (192.168.2.1)
        Origin IGP, metric 0, localpref 100, valid, internal
RT3#
```

- An EBGP-learned path is denied and not even installed in the BGP table – Routing Information Base (RIB) if the local ASN appears in the AS\_PATH attribute for the path.
- An EBGP-learned path will be denied by routing policies that implemented using access, prefix, AS\_PATH, and COMMUNITY lists. Rejects paths are still stored in the BGP table and marked as receive-only in the output of the **show ip bgp** {prefix [mask]} command if the **soft-reconfiguration inbound** keyword is configured for the EBGP peer.

```
RT1#sh ip bgp 192.168.3.0
BGP routing table entry for 192.168.3.0/24, version 0
Paths: (1 available, no best path)
  Not advertised to any peer
    65002, (received-only)
      12.12.12.2 from 12.12.12.2 (192.168.2.1)
        Origin IGP, localpref 100, valid, external
RT1#
```

- If the BGP Enforce the First Autonomous System Path feature is enabled using the **bgp enforce-first-as** BGP router subcommand, BGP discards a received EBGP-route that do not list the ASN of the EBGP peer as the first segment in the AS\_SEQUENCE-type AS\_PATH attribute for the path. This feature prevents a misconfigured or unauthorized peer from misdirecting or spoofing traffic by advertising a route as if it was sourced from the remote autonomous system. Take note that this command is enabled by default on most recent IOS releases and does not affect the operation of the BGP Local-AS feature, in which another ASN (the local ASN) that is different than the remote ASN is actually prepended to the AS\_SEQUENCE. Although the output of the **show ip bgp EXEC** command shows the manipulation of the AS\_PATH, the EBGP peer does not actually advertise the local ASN in the AS\_PATH.
- BGP is not designed to perform load balancing; BGP chooses only a single best path to reach a specific destination. The best paths are chosen because of policies, not based on bandwidth. The BGP best path selection process evaluates multiple paths until a single best path is left. The best path is then submitted to the routing table manager process and to be evaluated against other routing protocols that can also reach that network using the administrative distance rule. The routes that reside in the IP routing table (best paths) can then be advertised to other BGP peers.
- The remaining paths to reach a specific destination are still kept in the BGP table in case the best path becomes inaccessible.
- The Cisco IOS BGP implementation goes through the following process to choose the best route:
  - Prefer the route with the highest weight (Cisco-proprietary).
  - If multiple routes have the same weight, prefer the route with the highest local preference. **Note:** A path without the local preference is considered to have had the value set with the **bgp default local-preference** command, or to have a value of 100 by default, when being advertised to IBGP peers.
  - If multiple routes have the same local preference, prefer the route that was locally originated via a **network** or **aggregate-address** command, or through redistribution from an IGP. Local paths sourced by the **network** and **redistribute** commands are preferred over local aggregates sourced by the **aggregate-address** command. A locally originated route has a next-hop of 0.0.0.0 in the BGP table.
  - If the routes were not originated by the local router, prefer the route with the shortest AS path. The length of an AS\_SET is counted as 1, regardless of the number of ASes in the set; AS\_CONFED\_SEQUENCE and AS\_CONFED\_SET do not determine the AS path length. **Note:** This step is skipped if the **bgp bestpath as-path ignore** hidden BGP router subcommand is configured.
  - If the AS\_PATH length is the same, prefer the lowest ORIGIN code: **IGP < EGP < incomplete**.
  - If the routes have the same ORIGIN code, prefer the route with the lowest MED.
    - The MED comparison only occurs if the routes were received from the same AS; unless the **bgp always-compare-med** BGP router subcommand is enabled.
    - Any confederation sub-ASes are ignored – MEDs are being compared only if the first segment in the AS\_SEQUENCE is the same for multiple routes; any preceding AS\_CONFED\_SEQUENCE is ignored. \*TBC\*
    - If the **bgp bestpath med confed** is enabled, MEDs are compared for all paths that consist only of AS\_CONFED\_SEQUENCE – paths that are originated within the local confederation. \*TBC\*

- If the routes have the same MED value, prefer EBGp paths over IBGP paths.  
**Note:** Paths that contain AS\_CONFED\_SEQUENCE and AS\_CONFED\_SET are local to the confederation, and therefore are treated as internal (IBGP) paths. Additionally, there is no difference between Confederation External and Confederation Internal routes.  
**Reference:** RFC 5065 – Autonomous System Confederations for BGP – Section 5.3 – 4.
- If there is no EBGp neighbor but only IBGP neighbors and synchronization is disabled, prefer the path through the closest IBGP neighbor – the shortest path within the AS (lowest IGP metric) to reach the destination, in fact the BGP next-hop address for the route. Continue, even if the best path is already selected.
- The BGP Cost Extended Community attribute is an optional non-transitive attribute that is distributed to IBGP and confederation peers but not to EBGp peers. It provides a way to customize the local route preference and influence the best path selection process. This additional step which compares the BGP Cost Communities is added to the best path selection algorithm since the RFC is ratified. The path with the lowest cost value is preferred.
  - This step is skipped if the **bgp bestpath cost-community ignore** BGP router subcommand (available in Cisco IOS Release 12.3(2)T and later) is configured.
  - The **set extcommunity** route map clause configures Cost Community with a Cost Community ID number (0 to 255) and Cost value (0 to 4294967295). The Cost value determines the preference for a path. The path with the lowest Cost value is preferred. Paths that are not specifically configured with the Cost value are assigned with the default Cost value of 2147483647, which is the midpoint between 0 and 4294967295. These paths are then evaluated accordingly by the best path selection process. If 2 paths have the same Cost value, the path with the lower Cost Community ID is preferred.
- BGP chooses only a single best path for each destination. If the **maximum-paths {num}** BGP router subcommand is configured, and there are multiple external or confederation-external parallel paths through the same neighboring AS or sub-AS, BGP inserts the most recently received parallel paths up to the maximum of the configurable number (6) into the IP routing table for EBGp multipath load sharing. The default value when this command is not configured is 1 – no EBGp multipath load sharing. Continue, if the best path is not yet selected.
- When the routes are learned via EBGp, select the route that was received first (the oldest), as it is considered more stable and able to minimize the effect of route flapping. A newer path does not displace an older path, even if it is the preferred path based on the additional decision criteria below. It is recommended to apply the additional decision criteria below upon IBGP paths only, in order to ensure a consistent best path selection within an AS and thereby avoid routing loops.  
**Note:** This step is skipped if any of the following is true:
  - The **bgp bestpath compare-routerid** BGP router subcommand is enabled, in which the route received from the EBGp peer with the lowest BGP Router ID is selected as the best path for identical EBGp paths.
  - The BGP Router ID is the same for the routes – received from the same router.
  - There is no current best path. The current best path can be lost when the neighbor offering the path goes down. \*TBC\*
- Prefer the path received from the IBGP peer with the lowest BGP Router ID.  
**Note:** If a path contains route reflection attributes, the ORIGINATOR\_ID is used instead of the BGP Router ID.
- If the ORIGINATOR\_ID is the same for both paths, prefer the path with the shortest route reflection CLUSTER\_LIST. The length of the cluster list for routes without the CLUSTER\_LIST attribute is 0.
- If the BGP Router ID is the same for both paths, prefer the path received from the BGP peer with the lowest neighbor address – the IP address of the remote peer as configured in the **neighbor** BGP router subcommand.

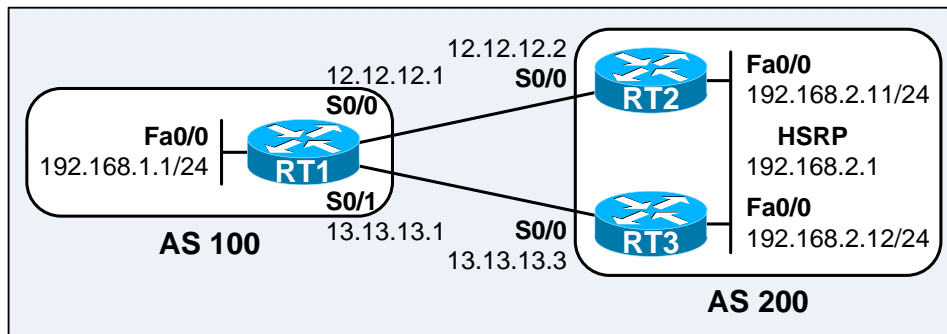


Figure 13-22: BGP Multipath Load Sharing

- Below shows the IP routing table on RT1 after implemented the **maximum-paths 2** command to perform BGP Multipath Load Sharing across the parallel EBGP paths over RT2 and RT3. Note that this command affects only the number of routes being inserted into and maintained in the IP routing table; not the number of best paths as to be selected by the BGP process. BGP still designates one of the paths as the best path based on the best path selection algorithm, as indicated by the > symbol; and this is the path that the router (RT1) advertises to both its IBGP and EBGP peers.

```

RT1#sh ip bgp
BGP table version is 4, local router ID is 192.168.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 192.168.1.0      0.0.0.0            0         32768  i
*> 192.168.2.0      13.13.13.3         0         0 200  i
*                   12.12.12.2         0         0 200  i
RT1#do sh ip route

Gateway of last resort is not set

  12.0.0.0/24 is subnetted, 1 subnets
C       12.12.12.0 is directly connected, Serial1/0
C       192.168.1.0/24 is directly connected, FastEthernet0/0
  13.0.0.0/24 is subnetted, 1 subnets
C       13.13.13.0 is directly connected, Serial1/1
B 192.168.2.0/24 [20/0] via 12.12.12.2, 00:01:32
                [20/0] via 13.13.13.3, 00:00:16
RT1#

```

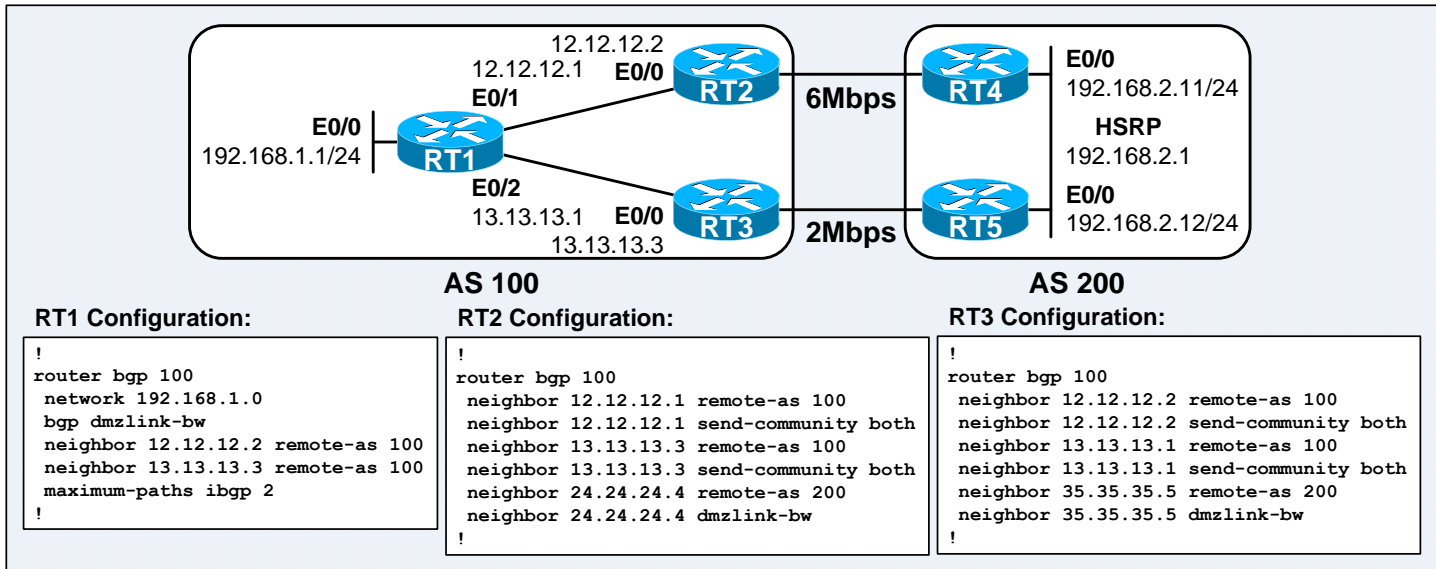
**Note:** Load balancing and load sharing are actually different concepts!

- The **maximum-paths {num}** BGP router subcommand enables EBGP multipath; while the **maximum-path ibgp {num}** BGP router subcommand enables IBGP multipath. The paths to the same destination must have the same weight, local preference, AS path length (in fact the actual value), Origin, and MED attributes in order to become multipath candidates.
- Below are some additional requirements for EBGP multipath:
  - The path should be learned from an external or confederation-external neighbor (EBGP).
- Below are some additional requirements for IBGP multipath:
  - The path should be learned from an internal neighbor (IBGP).
  - The IGP metric towards the BGP next-hops for the paths is same (equal-cost IGP paths); unless the routers are configured for Unequal-Cost IBGP Multipath Load Balancing.



- The BGP Link Bandwidth feature can be implemented for unequal-cost IBGP load balancing. It utilizes the Link Bandwidth Extended Community to advertise a received EBGP route along with the bandwidth of the link that connects to the EBGP neighbor to IBGP neighbors. The bandwidth is encoded as 4 octets in IEEE floating point format, in the unit of bytes per second. The Link Bandwidth Extended Community should not be propagated outside of an AS, similar to the Local Preference attribute.

**Note:** The draft-ramachandra-bgp-ext-communities-08 defines the Link Bandwidth Extended Community but never made it to the final **RFC 4360 – BGP Extended Communities Attribute**.



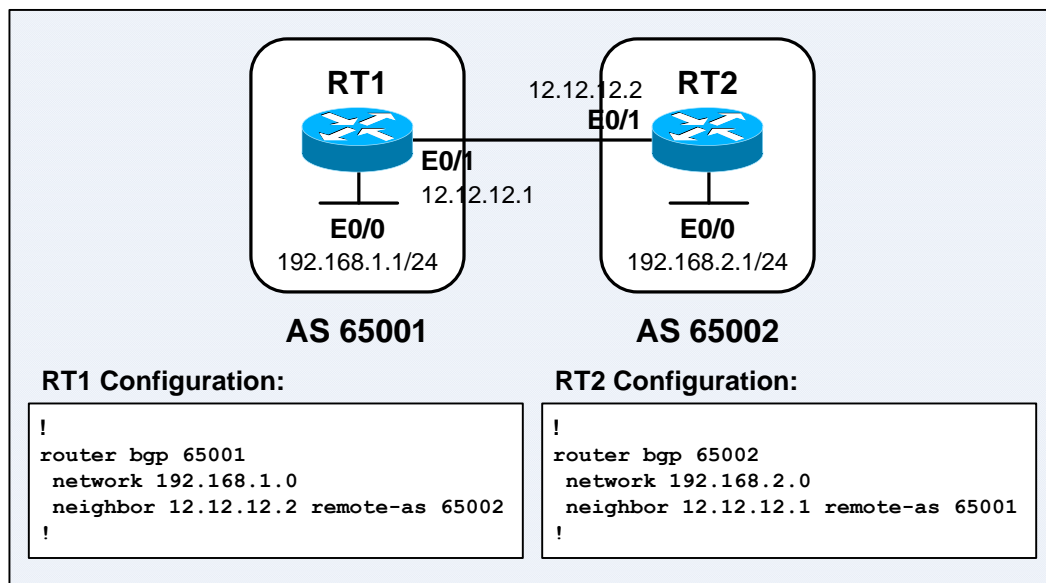
**Figure 13-23: BGP Link Bandwidth Unequal-Cost IBGP Load Balancing (Load Sharing)**

- Below shows that RT1 is able to achieve unequal-bandwidth load balancing with traffic sharing ratio proportional to the bandwidth of the inter-AS links.

```
RT1#sh ip bgp 192.168.2.0
BGP routing table entry for 192.168.2.0/24, version 5
Paths: (2 available, best #2, table Default-IP-Routing-Table)
Multipath: iBGP
Flag: 0x820
Not advertised to any peer
200
35.35.35.5 from 13.13.13.3 (35.35.35.3)
Origin IGP, metric 0, localpref 100, valid, internal, multipath
DMZ-Link Bw 250 kbytes
200
24.24.24.4 from 12.12.12.2 (24.24.24.2)
Origin IGP, metric 0, localpref 100, valid, internal, multipath, best
DMZ-Link Bw 750 kbytes
RT1#
RT1#sh ip route 192.168.2.0
Routing entry for 192.168.2.0/24
Known via "bgp 100", distance 200, metric 0
Tag 200, type internal
Last update from 24.24.24.4 00:00:13 ago
Routing Descriptor Blocks:
* 35.35.35.5, from 13.13.13.3, 00:00:13 ago
Route metric is 0, traffic share count is 1
AS Hops 1
Route tag 200
24.24.24.4, from 12.12.12.2, 00:00:13 ago
Route metric is 0, traffic share count is 3
AS Hops 1
Route tag 200
RT1#
```

## Chapter 14

# Basic BGP Lab



**Figure 14-1:** Basic BGP Network

- RT1 and RT2 in the figure above define each other as BGP neighbors, start an EBGP session, and advertise the networks 192.168.1.0/24 and 192.168.2.0/24 between them.
- The **router bgp** {local-as-num} global configuration command identifies the local AS and enters into the BGP configuration mode. The BGP process must be informed of its AS for it to differentiate between EBGP and IBGP neighbors when BGP neighbors are configured. Only a single instance of BGP process can be configured on a router at a time. Below shows the error message upon trying to initiate another instance of BGP process on RT1:

```
RT1(config)#router bgp 65002
BGP is already running; AS is 65001
RT1(config)#
```

- The **neighbor** {ip-addr | peer-group-name} **remote-as** {as-num} BGP router subcommand identifies a BGP peer router with which the local BGP router will establish a BGP session. Neighbor must be **explicitly configured** with this command for BGP to establish an adjacency. The BGP process determines whether the communication with the neighbor is an EBGP or IBGP session using the *as-num* parameter. BGP initiates an IBGP session when the *as-num* specified in the **router bgp** and the **neighbor remote-as** BGP router subcommands is the same; and initiates an EBGP session when the value is different. The **shutdown** keyword can be specified to disable (administratively shut down) an existing BGP neighbor or peer group. When implementing major policy changes and changing multiple parameters to a neighbor router, it is required to administratively shut down the neighboring router, implement the changes, and then enable the neighboring router with the **no neighbor** {ip-addr | peer-group-name} **shutdown** to bring back the session.
- The **neighbor** and **network** commands tell BGP *where* and *what* to advertise respectively. The **network** {net-num} [mask net-mask] BGP router subcommand defines a subnet or prefix in the IP routing table that is to be advertised by BGP. The *net-mask* identifies the subnet mask to be advertised by BGP. If the network mask is not specified, it is default to the classful mask.

- The **network** BGP router subcommand defines the networks that a BGP router should originate, which is a different concept from the IGP **network** commands that start up an IGP to send and receive IGP updates on the matched interfaces, as well as advertise the matched networks. Without a **network** statement, BGP passes along the received advertisements from other routers, but does not originate any network advertisements itself. In BGP, the network listed in the **network** statement does not have to be directly connected, as it does not identify interfaces on the router as like in other IGPs.
- The **mask** parameter is often being used as BGP-4 supports subnetting and supernetting. The list of network commands must include all the networks that an ASs want to advertise, not just those locally connected networks on the BGP router.
- The **network** BGP router subcommand allows classless prefixes; a BGP router can advertise individual networks, subnets, and supernets. Note that the prefix must exactly match an entry (both address and mask) in the IP routing table. A static null route is often configured to create a supernet entry in the IP routing table in order to allow BGP to advertise the supernet.
- If *net-mask* is not specified, the **network** command announces only the classful network number. At least one subnet of the specified major network must be present in the IP routing table to allow BGP to announce the classful network as a BGP router. However, when the *net-mask* is specified, an exact match to the network (both address and mask) must exist in the routing table for the network to be advertised.

**Ex #1:** The **network 192.168.1.1 mask 255.255.255.0** command is configured by mistake. BGP looks for 192.168.1.1/24 in the routing table. It might find 192.168.1.0/24, but it will never find 192.168.1.1/24. Since the routing table does not contain a specific match upon the network, BGP does not announce the 192.168.1.1/24 network to any peer.  
**Note:** This example is just for demonstration purpose and actually cannot be configured.

```
Router(config-router)#network 192.168.1.1 mask 255.255.255.0
% BGP: Incorrect network or mask configured
Router(config-router)#
```

**Ex #2:** The **network 192.168.0.0 mask 255.255.0.0** command is configured to advertise a CIDR block. BGP looks for 192.168.0.0/16 in the routing table. It might find 192.168.1.0/24, but it will never find 192.168.0.0/16. BGP does not announce the 192.168.0.0/16 to any peer. The static route **ip route 192.168.0.0 255.255.0.0 Null0** can be configured for BGP to find an exact match in the routing table and therefore announces the 192.168.0.0/16 network to any peer.

- The **show ip bgp summary** EXEC command displays the status about all BGP connections:

```
RT1#sh ip bgp summary
BGP router identifier 192.168.1.1, local AS number 65001
BGP table version is 3, main routing table version 3
2 network entries using 202 bytes of memory
2 path entries using 96 bytes of memory
2 BGP path attribute entries using 120 bytes of memory
1 BGP AS-PATH entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 442 total bytes of memory
BGP activity 2/0 prefixes, 2/0 paths, scan interval 60 secs

Neighbor      V    AS MsgRcvd MsgSent  TblVer  InQ  OutQ Up/Down  State/PfxRcd
12.12.12.2    4 65002      6      6        3    0    0 00:01:54      1
RT1#
```

- The **BGP table version** increments upon the changes in the BGP table.
  - The **TbIVer** indicates the last version of the BGP table that was sent to a neighbor.
  - The **InQ** (in queue) indicates the number of messages from a neighbor that are waiting to be processed; the **OutQ** (out queue) indicates the number of messages queued up and waiting to be sent to a neighbor. TCP flow control prevents the local router from overwhelming a neighbor with a large update.
  - The **Up/Down** indicates the length of time a neighbor has been in the current BGP state – established, active, or idle.
  - The **PfxRvd** (prefix received) indicates the number of BGP network entries that have been received from a neighbor when the session is in the established state.
- The **show ip bgp neighbors [ip-addr]** EXEC command displays detailed information about the TCP and BGP connections to all or a specified BGP peer:

```

RT1#sh ip bgp neighbors
BGP neighbor is 12.12.12.2, remote AS 65002, external link
  BGP version 4, remote router ID 192.168.2.1
  BGP state = Established, up for 00:02:00
  Last read 00:00:00, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0

                Sent          Rcvd
  Opens:                1            1
  Notifications:        0            0
  Updates:               1            1
  Keepalives:           5            5
  Route Refresh:         0            0
  Total:                 7            7
  Default minimum time between advertisement runs is 30 seconds

For address family: IPv4 Unicast
  BGP table version 3, neighbor version 3
  Index 1, Offset 0, Mask 0x2

                Sent          Rcvd
  Prefix activity:     ----          ----
  Prefixes Current:    1            1 (Consumes 48 bytes)
  Prefixes Total:      1            1
  Implicit Withdraw:   0            0
  Explicit Withdraw:   0            0
  Used as bestpath:    n/a          1
  Used as multipath:   n/a          0

                Outbound      Inbound
  Local Policy Denied Prefixes:  -----          -----
  Bestpath from this peer:        1            n/a
  Total:                           1            0
  Number of NLRI's in the update sent: max 1, min 0

  Connections established 1; dropped 0
  Last reset never
  Connection state is ESTAB, I/O status: 1, unread input bytes: 0
  Local host: 12.12.12.1, Local port: 179
  Foreign host: 12.12.12.2, Foreign port: 29301
  --- output omitted ---

```

- The **show ip bgp EXEC** command displays entries in the BGP topology database (BGP table). Specify a network number and subnet mask for more specific information about a particular network:

```

RT1#sh ip bgp
BGP table version is 3, local router ID is 192.168.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 192.168.1.0      0.0.0.0           0             32768 i
*> 192.168.2.0      12.12.12.2        0             0 65002 i
RT1#
RT1#sh ip bgp 192.168.1.0
BGP routing table entry for 192.168.1.0/24, version 2
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
   Advertised to non peer-group peers:
     12.12.12.2
   Local
     0.0.0.0 from 0.0.0.0 (192.168.1.1)
       Origin IGP, metric 0, localpref 100, weight 32768, valid, sourced, local,
best
RT1#
RT1#sh ip bgp 192.168.2.0
BGP routing table entry for 192.168.2.0/24, version 3
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
   Not advertised to any peer
     65002
     12.12.12.2 from 12.12.12.2 (192.168.2.1)
       Origin IGP, metric 0, localpref 100, valid, external, best
RT1#

```

The status codes are shown at the beginning of each entry; and the ORIGIN codes are shown at the end of each entry.

- An **s** indicates that route summarization has been performed and the route is suppressed.
- A **d** indicates that the route is being dampened for going up and down (flapping) too often. Although the route is up at the moment, it is not advertised until the penalty has expired.
- An **h** (history) indicates that the route is unavailable and is probably down; historic information about the route exists, but a best route does not exist.
- An **>** indicates the best path for a route selected by the BGP best path selection algorithm; this route is offered to the IP routing table and advertised to other BGP neighbors.
- An **i** indicates the route is learned from an IBGP peer.
- The **0.0.0.0** in the Next Hop column means that the local router has originated the route.
- The Path column contains the AS\_PATH information. The first ASN listed is the adjacent AS that the route was learned from; and the last ASN is the originating AS. If the Path column is blank, the route is originated from the local AS.
- The last column indicates how the route was entered into BGP on the originating router. An **i** means that the originating router has used a **network** or **redistribute** command to introduce the route into BGP; an **e** means that the originating router has redistributed an EGP route into BGP; an **?** means that the originating BGP process is unable to absolutely verify the availability of the route, as it is redistributed from an IGP into BGP.

## BGP Peer Groups, BGP Synchronization, Source IP Address, EBGP Multihop, and Next-Hop Attribute Configuration

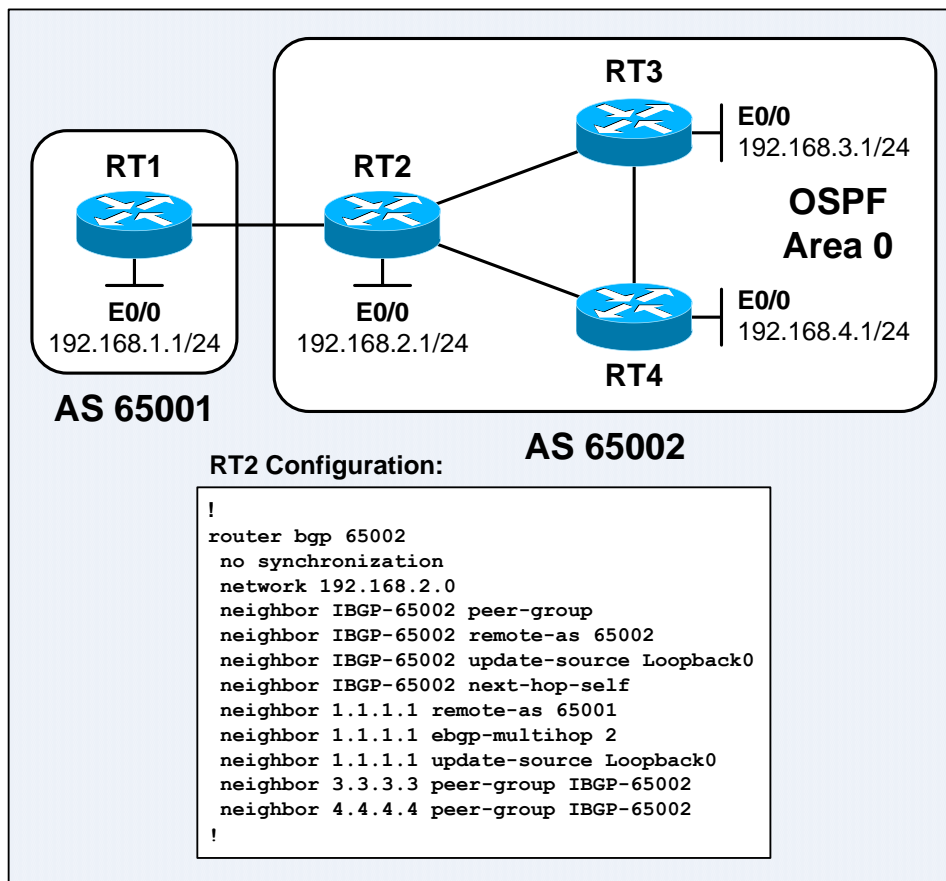


Figure 14-2: Advanced BGP Network

- In BGP, many neighbors are often configured with the same update policies, eg: they have the same route filtering applied. Neighbors with the same update policies can be grouped into **peer groups** to simplify configuration and improve route update efficiency and save resources, in which updates are generated once per peer group, rather than once for each neighbor. This approach is highly recommended when a router has are many peers.
- Instead of separately defining the same policies for each neighbor, a peer group can be defined with these policies (similar to a template) assigned to the peer group. Individual neighbors are then configured as the members of the peer group; the template is then applied upon all the members of the peer group. Note that only the options that affect the **incoming updates** can be overridden for particular members of a peer group – a peer group is a set of BGP neighbors that share the same outbound policy, but their inbound policies may be different.
- The **neighbor {peer-group-name} peer-group** BGP router subcommand creates a peer group. The peer group name is local only to the router it is configured on and not passed to other routers. The **neighbor {ip-addr} peer-group {peer-group-name}** BGP router subcommand assigns a neighbor as a member of a peer group. A neighbor can be a member of only one peer group.
- The BGP synchronization rule states that a BGP router cannot use an IBGP-learned route and advertise it to an EBGP neighbor; unless the BGP and IGP routing tables are synchronized. When BGP synchronization is disabled, BGP can use IBGP-learned routes that are not present in the local routing table. BGP synchronization can be disabled for stub ASes; or if all routers in the BGP transit path with a transit AS are running BGP – full-mesh IBGP. Disabling this feature reduced the number of routes carried in IGP and allows BGP to converge more quickly.

- The **neighbor** command tells the BGP process the destination IP address of the BGP packets; the BGP process must decide the IP address to use as the source IP address for the BGP packets. When the BGP process creates a packet, it performs a routing table lookup for the destination address and use the IP address of the outbound interface towards the destination address as the source address for BGP packets by default.
- The source IP address must match the address in the corresponding **neighbor** command on the other BGP router. Otherwise, the BGP routers will not be able to establish a BGP session. BGP does not accept unsolicited packets; it must have a **neighbor** statement for every neighbor.
- The IP addresses of loopback interfaces are often being used to establish IBGP sessions when there are multiple paths between the IBGP neighbors. The **neighbor {ip-addr | peer-group-name} update-source loopback {intf-num}** BGP router subcommand informs a BGP router to use the IP address of its loopback interface as the source for BGP connections to its neighbors.
- Peering using loopback interfaces adds resiliency to the IBGP sessions, as the IBGP sessions no longer tied to any physical interface, which might go down for many reasons. Note that the routers must have a route to the loopback address of the other neighbor in their IGP routing table.
- BGP does not require neighbors to be directly attached to the same subnet. BGP routers use TCP Port 179 connections for exchanging BGP messages; therefore allowing neighboring routers to reside on the same subnet, or separated by several routers, which is relatively common.
- When peering with an EBGP neighbor, BGP can reach the IP address of the directly connected EBGP neighbor without further configuration, and it is used as the destination address by default. Since loopback is never directly connected and EBGP neighbors never run IGP between them, static routes pointing to the physical address of the directly connected neighbor must be used, and **EBGP multihop** must also be enabled.  
**Tip:** Peer between loopback interfaces when peering between routers within the same AS – IBGP; peer to physical interfaces when peering with external ASes – EBGP, for faster failover.
- EBGP packets have a Time To Live (TTL) of 1 and hence only directly connected neighbors are allowed by default. The **neighbor {ip-addr | peer-group-name} ebgp-multihop [ttl]** BGP router subcommand configures a BGP router to accept and attempt EBGP connections to EBGP neighbors that are not directly connected. The **neighbor ebgp-multihop** command sets the TTL to 255 by default if the optional *ttl* parameter is not specified. This command must be used whenever the EBGP neighbors are more than one hop way, includes peering to loopbacks.  
**Note:** IBGP packets always have a TTL of 255 and no command is available to change it.
- When there are multiple physical connections between EBGP neighbors, using loopback interfaces and static routes to the loopback interfaces allow load balance between the connections.
- As discussed in the “**The NEXT\_HOP Attribute**” section earlier in Chapter 13, it is sometimes necessary to override the default behavior of a BGP router and configure it advertise itself as the next-hop for the routes advertised to a neighbor.
- In this sample scenario, instead of advertising the route to RT1 – 1.1.1.1 through an IGP, the **next-hop-self** configuration can be implemented on the AS border BGP routers, RT2 in this case, to override the default next-hop address – 1.1.1.1 with the IP addresses of the AS border routers, 2.2.2.2 in this case. IBGP peers (RT3 and RT4) within AS 65002 would then forward packets destined to the external networks – 192.168.1.0/24 through RT2.

## BGP Neighbor Authentication

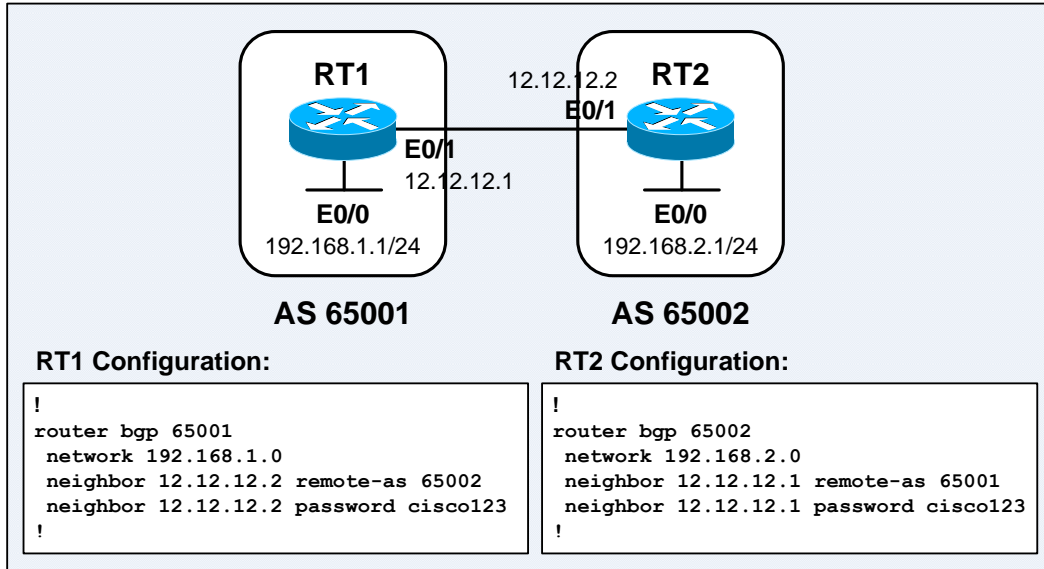


Figure 14-3: BGP Neighbor Authentication

- BGP neighbor authentication can be configured on a BGP router to authenticate the source of every BGP packets that it receives. This is accomplished exchanging a message digest or hash of the authenticating key or password that is known to both the sending and receiving routers. The authenticating key itself is not sent to prevent it from being captured by eavesdroppers.
- The **neighbor {ip-addr | peer-group-name} password {string}** BGP router subcommand enable MD5 authentication on a TCP connection between 2 BGP peers. The *string* is a case-sensitive password of up to 25 characters and 81 characters when the **service password-encryption** global configuration command is enabled and disabled respectively. The 1st character cannot be a number. The string can contain any alphanumeric characters, including spaces. However, it cannot be specified in the number-space-anything format. A space after a number can cause authentication to fail. Any combination of the following symbolic characters can be used along with the alphanumeric characters:  
`` ~ ! @ # $ % ^ & * ( ) - _ = + | \ } ] { [ " ` : ; / > < . , ?`
- Upon enabling BGP neighbor authentication, BGP routers generates and verify the MD5 digest of every TCP segment communicated across the established TCP connection between them.
- If the authentication key is configured wrongly, the BGP peering session will not be established. Always enter the authentication key carefully and verify that the peering session is established after configured authentication.
- When configuring or changing the password authentication key in an established BGP session, the local router will not tear down the existing session after configured the password. The local router will attempt to maintain the peering session using the new password until the BGP holddown timer expires. The default hold time is 180 seconds. If the password is not configured or changed on the remote router before the holddown timer expires, the session will time out and terminate.

**Note:** Configuring a new holddown timer value will only take effect after the session is reset. It is impossible to change the holddown timer value to avoid resetting the BGP session.



## Resetting BGP Sessions

- BGP often handles huge volumes of routing information. A BGP router unable to go through the huge BGP table to recalculate the invalid entries upon a BGP routing policy change occurs, eg: when distribute lists, prefix lists, and filter lists are changed. The BGP router also unable to determine the advertised routes that should be withdrawn from a neighbor. Also imagine that a configuration change after another would cause the whole process to start all over again. To avoid these problems, Cisco IOS enforces the new routing policy only upon routes that received and transmitted after the BGP routing policy configuration change has been performed.
- If the routing policy change wanted to be applied upon all routes, we must force the router to process all routes through the new filter. If the filter is applied upon incoming information, the router wants its neighbor to resend its BGP table so that it passes through the new filter; if the filter is applied upon outgoing information, the router will resend its BGP table through the new filter. Resetting the affected established BGP sessions after a BGP routing policy changes performs the mentioned actions. When the BGP sessions are reset, all information received from those sessions will be removed from the BGP table; the remote neighbor detects a BGP session DOWN state and also removes the received routes. After a while (30 – 60 seconds), the BGP sessions are reestablished automatically, and the BGP table is exchanged again and the routing information is being passed through and processed by the new filter.
- The **clear ip bgp** *{\* | ip-addr}* [*{in | out}*] | *{soft [in | out]}*] privileged command resets BGP sessions and removes entries from the BGP table. Issue this command upon a routing policy change to ensure that the change is activated and the peers are informed regarding the change. The **\*** keyword resets all current BGP sessions; while the *ip-addr* identifies a specific neighbor for which the BGP session will be reset. The optional **soft** keyword performs a **soft reset**, along with the optional **in** and **out** keywords trigger inbound or outbound soft reset; if both the keywords are not specified, both inbound and outbound soft resets will be triggered.
- The **clear ip bgp** *{ip-addr}* **out** and **clear ip bgp** *{ip-addr}* **soft out** privileged commands perform the same function – cause the router to reevaluate its existing BGP table and create a new BGP Update for the specified neighbor. Both the **clear ip bgp** *{ip-addr}* **in** and **clear ip bgp** *{ip-addr}* **soft in** privileged commands perform soft reset either through the dynamic soft reset with the Route Refresh feature or soft reconfiguration depends upon whether the **soft-reconfiguration inbound** option has been configured for the particular neighbor.
- The **clear ip bgp peer-group** *{peer-group-name}* privileged command resets the BGP sessions for all members of a BGP peer group.
- Resetting BGP sessions and clearing the BGP table disrupts routing and packet forwarding. Resetting a BGP session is a method of information the neighbor(s) regarding a policy change. The **clear ip bgp** command without the **soft** keyword performs a hard reset of BGP sessions, which means that the router on which this command is issued closes and reestablishes the corresponding TCP connections, and resends all information to the affected neighbor(s).
- Issuing the **clear ip bgp \*** command causes a very disastrous event. It completely deleted the BGP table on the router on which this command is issued, forces it to relearn all networks from every neighbor, which will take a considerable number of CPU cycles to process all the information that sent by all neighbors about the same time. If the **clear ip bgp** *{ip-addr}* command is issued, only one neighbor is reset and affected at a time, at the impact is less severe. However, this approach takes longer time for the routing policy to take effect to all neighbors, as the reset must be done individually rather than all at once using the **clear ip bgp \*** command.

- Below shows the output of the **debug ip bgp updates** and **clear ip bgp \*** commands on RT1:

```

RT1#debug ip bgp updates
BGP updates debugging is on
RT1#
RT1#clear ip bgp *
RT1#
00:01:58: %BGP-5-ADJCHANGE: neighbor 12.12.12.2 Down User reset
00:02:34: %BGP-5-ADJCHANGE: neighbor 12.12.12.2 Up
00:02:35: BGP(0): 12.12.12.2 rcvd UPDATE w/ attr: nexthop 12.12.12.2, origin
i, metric 0, path 65002
00:02:35: BGP(0): 12.12.12.2 rcvd 192.168.2.0/24
00:02:35: BGP(0): nettable_walker 192.168.1.0/24 route sourced locally
00:02:35: BGP(0): Revise route installing 1 of 1 route for 192.168.2.0/24 ->
12.12.12.2 to main IP table
00:02:35: BGP(0): 12.12.12.2 computing updates, afi 0, neighbor version 0,
table version 3, starting at 0.0.0.0
00:02:35: BGP(0): 12.12.12.2 send UPDATE (format) 192.168.1.0/24, next
12.12.12.1, metric 0, path
00:02:35: BGP(0): 12.12.12.2 1 updates enqueued (average=52, maximum=52)
00:02:35: BGP(0): 12.12.12.2 update run completed, afi 0, ran for 8ms,
neighbor version 0, start version 3, throttled to 3
00:02:35: BGP(0): 12.12.12.2 initial update completed
RT1#
=====
RT2#
00:01:59: %BGP-5-ADJCHANGE: neighbor 12.12.12.1 Down Peer closed the session
00:02:34: %BGP-5-ADJCHANGE: neighbor 12.12.12.1 Up
RT2#

```

- The **soft out** option allows BGP to perform a soft reset for outbound updates, in which it creates a new version of the BGP table and sends the new set of updates to the specified neighbors. The update includes the withdrawn routes in which the neighbors must not use anyone based on the new outbound policy. Outbound soft reconfiguration does not have any memory overhead. An outbound reconfiguration can also be triggered on the other side of the BGP session to make the new inbound policy on the local router to take effect.
- The **soft in** option generates new inbound updates without resetting the BGP sessions, but it can be memory-intensive as it requires the storing of unprocessed routing table update information. BGP does not allow a router to force its peer to send its entire table. Inbound soft reconfiguration can be configured on a router for it to process the routes in the BGP table upon changing the inbound BGP policy without performing a hard reset.
- There are 2 ways to perform an inbound soft reconfiguration – using the stored routing update information and dynamically using the Route Refresh feature. The **neighbor {ip-addr | peer-group-name} soft-reconfiguration inbound** command configures the BGP process to save all updates learned from the specified neighbor(s) in case the inbound policy is changed. The BGP process retains an unfiltered table of what the neighbor(s) is sent in the BGP table. Hence it is no need to force the other side to resend everything upon the inbound policy change. After configured the **neighbor soft-reconfiguration inbound** command for the first time, reset all the current BGP sessions so that all the routing update information will be resent by all neighbors and can then be stored in the local BGP table.  
**Note:** If the BGP routers support the BGP Route Refresh capability, it is not necessary to perform a hard reset after configured the **neighbor soft-reconfiguration inbound** command.  
**Note:** In general, don't ever implement soft reconfiguration to prevent high memory utilization and most BGP speakers nowadays already support the Route Refresh capability.

- The BGP Dynamic Soft Reset feature that available in Cisco IOS Software Release 12.1 and later utilizes the BGP Route Refresh capability to allow the dynamic requests of inbound BGP routing updates that does not require storing of unprocessed routing table update information and therefore has no memory overhead. This new method does not require any pre-configuration. Use the **show ip bgp neighbors {ip-addr}** command to determine whether a neighbor supports the Route Refresh capability. If the BGP routers support the Route Refresh capability, both the **clear ip bgp in** and **clear ip bgp soft in** commands can be issued to perform dynamic soft reconfiguration.
- The BGP Route Refresh capability provides a mechanism to request a re-advertisement of the **Adj-RIB-Out** from a BGP peer. The **Adj-RIB-In** stores the unprocessed routing information that has been learned via inbound update messages; the **Loc-RIB** contains the local routing information that has been processed by applying the local routing policies upon the routing information contained in the **Adj-RIB-In**; and the **Adj-RIB-Out** stores the routing information that the local router has selected for advertisement to its peers; in which the routing information in the Loc-RIB has been processed through the outbound policies, eg: filtering and path attributes manipulation.
- BGP routers implement RFC 2842 – Capabilities Advertisement with BGP-4 to advertise the RFC 2918 BGP Route Refresh capability between BGP peers using the Optional Parameters field in the BGP Open messages. BGP Route Refresh message is a new BGP message type defined using the BGP message Type Code 5.

## Troubleshooting BGP Neighbor States

- A BGP router goes through a finite state machine that has the **Idle**, **Connect**, **Active**, **Open Sent**, **Open Confirm**, and **Established** states with its neighbors. The **debug ip bgp** privileged command shows the various BGP states throughout the neighbor establishment process.

```

RT1#debug ip bgp
BGP debugging is on
RT1#
00:00:55: BGP: 12.12.12.2 went from Idle to Active
00:00:55: BGP: 12.12.12.2 open active, delay 26011ms
00:01:11: BGP: 12.12.12.2 passive open
00:01:11: BGP: 12.12.12.2 went from Active to Idle
00:01:11: BGP: 12.12.12.2 went from Idle to Connect
00:01:11: BGP: 12.12.12.2 rcv message type 1, length (excl. header) 26
00:01:11: BGP: 12.12.12.2 rcv OPEN, version 4
00:01:11: BGP: 12.12.12.2 went from Connect to OpenSent
00:01:11: BGP: 12.12.12.2 sending OPEN, version 4, my as: 65001
00:01:11: BGP: 12.12.12.2 went from OpenSent to OpenConfirm
00:01:11: BGP: 12.12.12.2 send message type 1, length (incl. header) 45
00:01:11: BGP: 12.12.12.2 went from OpenConfirm to Established
00:01:11: %BGP-5-ADJCHANGE: neighbor 12.12.12.2 Up
RT1#

```

- After a **neighbor** BGP router subcommand is configured, the BGP session starts in the Idle state, and the BGP process checks whether there is a route to the newly configured neighbor. The BGP session should remains in the Idle state for only a few seconds. If the BGP process is unable to locate a route to the neighbor, the BGP session remains in the Idle state. If it finds a route, it initiates a TCP connection to the neighbor and the BGP session enters into the Connect state when the TCP SYN ACK segment returns, and the TCP three-way handshake is completed.

- After the TCP connection is established, the BGP process sends a BGP Open message to the neighbor and the BGP session enters into the Open Sent state. If there is no response for 5 seconds, the state changes back to the Active state. If a response returns within 5 seconds, the BGP session enters into the Open Confirm state and the BGP process starts scanning (evaluating) the routing table for the paths to send to the neighbor. Eventually the BGP session enters into the Established state and the BGP process begins to exchange routing information with the neighbor.
- When a BGP router remains in the Idle state, it means that it is unable to reach the IP address configured in the **neighbor** statement. Verify the following to troubleshoot the BGP Idle state:
  - Ensure that the static route to the neighbor IP address is configured correctly.
  - Ensure that the IGP has announced the neighbor IP address correctly.
  - Ensure that the neighbor IP address is configured correctly.
- When a BGP router remains in the Active state, it means that it can reach the IP address configured in the **neighbor** statement and has sent a BGP Open message to the neighbor but has not received a response (the Open message from the neighbor) back from the neighbor. Verify the following to troubleshoot the BGP Active state:
  - Ensure that the neighbor has a return static or IGP route to the source IP address.
  - Ensure that the source IP address is advertised into the IGP on the neighbor.
  - Ensure that the neighbor has a **neighbor** statement peering back to the local router.
  - Ensure that the **neighbor** statement on the neighbor is configured correctly.
- If the state toggles between Idle and Active, verify that the AS numbers are configured correctly. Below shows the error messages on 2 neighboring routers in which the wrong ASN is configured in the **neighbor** statements – RT1 in AS 65001 is configured to peer with RT2 in AS 65002; while RT2 in AS 65002 is configured to peer to RT1 in AS 65002.

```

RT1#
00:01:05: %BGP-3-NOTIFICATION: received from neighbor 12.12.12.2 2/2 (peer
in wrong AS) 2 bytes FDE9
RT1#
=====
RT2#
00:01:05: %BGP-3-NOTIFICATION: sent to neighbor 12.12.12.1 2/2 (peer in
wrong AS) 2 bytes FDE9 FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF 002D 0104
FDE9 00B4 C0A8 0101 1002 0601 0400 0100 0102 0280 0002 0202 00
RT2#

```

*This page is intentionally left blank*

# BGP Route Summarization, Route Filtering, and Route Reflection

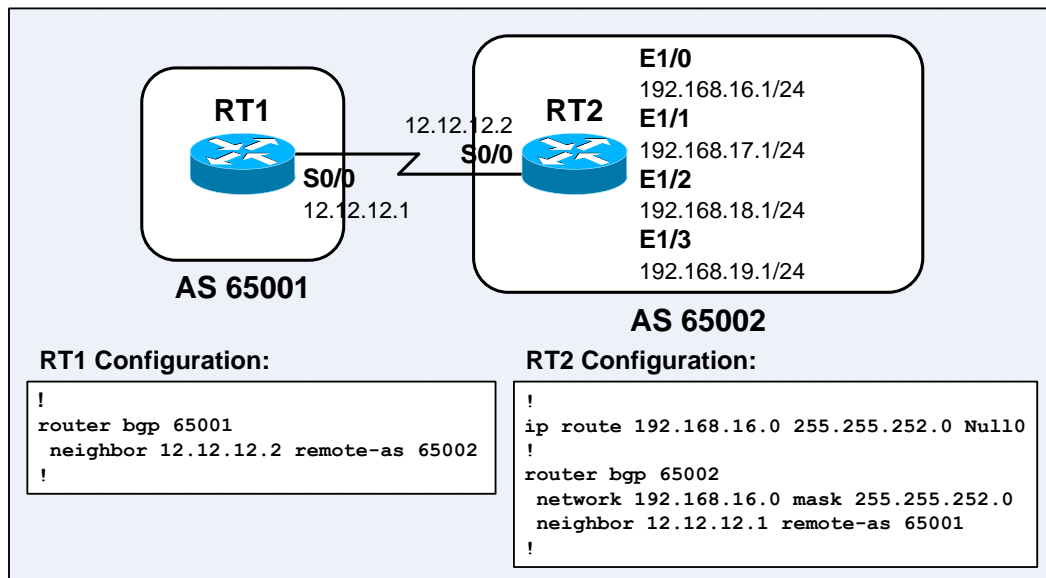
## BGP Route Summarization

- CIDR is a mechanism developed to overcome the problem of exhaustion of IP addresses and the growth of Internet routing table. CIDR allows blocks of multiple addresses to be combined, or aggregated, to create a set of classless IP addresses that summarize the routes in routing tables and resulting in fewer route advertisements.
- Earlier versions of BGP did not support CIDR until BGP-4. BGP-4 Update messages include both the prefix and prefix length; while previous version included only the prefix, and the length was assumed based on the address class.
- A BGP router can advertise overlapping routes to another BGP router. **Overlapping routes** are non-identical routes that point to the same destination, eg: 10.10.128.0/17 and 10.10.192.0/18, in which the 2nd route is actually included in the 1st route.
- **Aggregation** is the BGP term for summarization. Addresses can be aggregated when advertised by a BGP router. The BGP routers can be configured to include the combined unordered list of all ASes (AS\_SET-type AS\_PATH attribute) contained in all paths that are being summarized. 2 BGP path attributes that are related to aggregate addressing are:
  - **ATOMIC\_AGGREGATE** – A well-known discretionary attribute that is set to inform the downstream ASes that the originating router has aggregated the routes and indicate that detailed path information might be lost due to route aggregation or summarization. Any downstream BGP router that receives a route with this attribute set must remain the attribute when advertising the route to other BGP peers.
  - **AGGREGATOR** – An optional transitive attribute that specifies the BGP Router ID and AS number of the router that perform the route aggregation.
- Aggregate addresses are not used in the Internet as much as they could be because ASes that are multihomed (connected to multiple ISPs) want to make sure that their routes are advertised without being aggregated into a summarized route.
- RIPv1, RIPv2, IGRP, and EIGRP perform autosummarization by default. Autosummarization can be disabled for RIPv2 and EIGRP. Autosummarization must be disabled when an organization is assigned a portion of a Class A, B, or C networks, or else it will claim the ownership of the whole Class A, B, or C networks, which eventually causes routing problems.
- The **auto-summary** BGP router subcommand determines how BGP handles redistributed routes. The **no auto-summary** BGP router subcommand disables BGP autosummarization. When it is enabled, all redistributed subnets are summarized to their classful boundaries and stored in the BGP table. When it is disabled, all redistributed subnets are present in their original form with the original prefix and prefix length in the BGP table.

**Ex:** An ISP assigns the 11.22.33.0/24 network to an AS. If the AS uses the **redistribute connected** command to introduce this network into BGP and BGP autosummarization is enabled, BGP will announce that the AS owns the whole 11.0.0.0/8 address block, which can cause connectivity problems upon other ASes that own a portion of the 11.0.0.0/8 address space. The **network 11.22.33.0 mask 255.255.255.0** statement should be configured instead of the redistribute connected command to ensure that the assigned network is advertised correctly.

**Note:** The default behavior of the auto-summary BGP router subcommand was changed to disabled in Cisco IOS Release 12.2(8)T and later.

- The **network** BGP router subcommand with the **mask** option installs a prefix into the BGP table and advertises it to other peers when a matching IGP prefix exists in the IP routing table. If the IGP prefix flaps, the BGP prefix also flaps.
- Use the **network** {net-num} command without the **mask** option to advertise a classful network. Use the **network** {net-num} [mask net-mask] command to advertise an aggregate of prefixes that originate in the local AS. Remember that the prefix must exactly match and entry in the IP routing table (both address and prefix) for the network to be advertised. This exact match can be accomplished by using a static route with the Null0 interface, or it might already exist in the IP routing table when the IGP already performing the summarization.
- When the **network** command is configured for a classful address and there is at least one subnet of the classful address space exists in the IP routing table; BGP advertises the classful network and not the subnet. If the only single subnet for the classful network becomes unavailable, BGP will withdraw the classful network from all neighbors.  
**Note:** The BGP auto-summarization must be enabled for this scenario to work.
- The **network** command tells BGP *what* to advertise but not *how* to advertise it. We will understand this statement when learning about the **aggregate-address** command later.



**Figure 15-1:** BGP Route Summarization using the **network** Command

- Below shows that RT1 learned the summarized BGP route – 192.168.16.0/22 from RT2:

```

RT1#sh ip route

Gateway of last resort is not set

    12.0.0.0/24 is subnetted, 1 subnets
C       12.12.12.0 is directly connected, Serial0/0
B 192.168.16.0/22 [20/0] via 12.12.12.2, 00:00:17
RT1#
RT1#sh ip bgp
BGP table version is 2, local router ID is 12.12.12.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop           Metric LocPrf Weight Path
*> 192.168.16.0/22  12.12.12.2         0             0 65002 i
RT1#

```

- The **network** command was not designed to perform summarization; while the **aggregate-address** command was designed for summarization. The **aggregate-address** command aggregates only networks that are already in the BGP table; while the **network** command advertises summary routes that must exist in the IP routing table.
- A BGP router always chooses the more-specific path when making a best-path decision; however, the BGP router has several options as below when advertising overlapping routes:
  - Advertise both the more-specific and the less-specific route
  - Advertise only the more-specific route
  - Advertise only the less-specific route
  - Advertise only the non-overlapping part of the route
  - Aggregate the more-specific and less-specific routes and advertise the aggregate route
  - Advertise neither route
- The **aggregate-address** *{ip-addr net-mask}* [**summary-only**] [**as-set**] BGP router subcommand creates an aggregate (or summary) entry in the BGP table. The *ip-addr net-mask* identifies the aggregate address and the mask or the aggregate address to be created. The **aggregate-address** command automatically generates a BGP route for the summarized route with to Null0 interface and installed in the IP routing table.
- The optional **summary-only** keyword causes the router to advertise only the aggregate route; the default is advertises both the aggregate prefix and the component more-specific prefixes. Without the **summary-only** keyword, the aggregation router will still advertises the more-specific prefixes, which can be useful for redundant ISP links.  
**Ex:** ISP1 is advertising the summary and the more-specific routes, and ISP2 is advertising only the summary route. The more-specific routes through ISP1 are followed during normal operation. When ISP1 that advertising the more-specific routes becomes inaccessible, ISP2 that advertising only the summary route is followed.
- The optional **as-set** keyword generates the AS\_PATH attribute of the aggregate route to include all the ASNs listed in all the paths of the more-specific routes. The default is that the AS\_PATH attribute of the aggregate route lists only the ASN of the aggregation router, in which the aggregate route is advertised as originated from the AS of the aggregation router; and the ATOMIC\_AGGREGATE attribute is set to indicate that detailed path information might be lost due to route aggregation or summarization. The ATOMIC\_AGGREGATE attribute does not have to be included in the aggregate route when the **as-set** keyword is specified.
- If there is a route within the range indicates by the **aggregate-address** command exists in the BGP table, the aggregate route will be inserted into the BGP table and advertised to other peers. This process creates more information in the BGP table. The more-specific routes covered by the aggregate route should be suppressed from being advertised to other peers using the **summary-only** keyword. When the more-specific routes are suppressed, they are still present in the BGP table of the aggregation router.
- Both the **summary-only** and **as-set** keywords may be specified in an **aggregate-address** command, which sends only the summary address and lists all the ASes in the path information.
- The **aggregate-address** command does not replace the **network** command; at least a more-specific route that is to be summarized must reside in the BGP table. The more-specific routes may be injected into BGP by other routers, and the route aggregation is performed by another router or even in another AS – **proxy aggregation** – the aggregation router needs to configure only the **aggregate-address** command; instead of the **network** command.



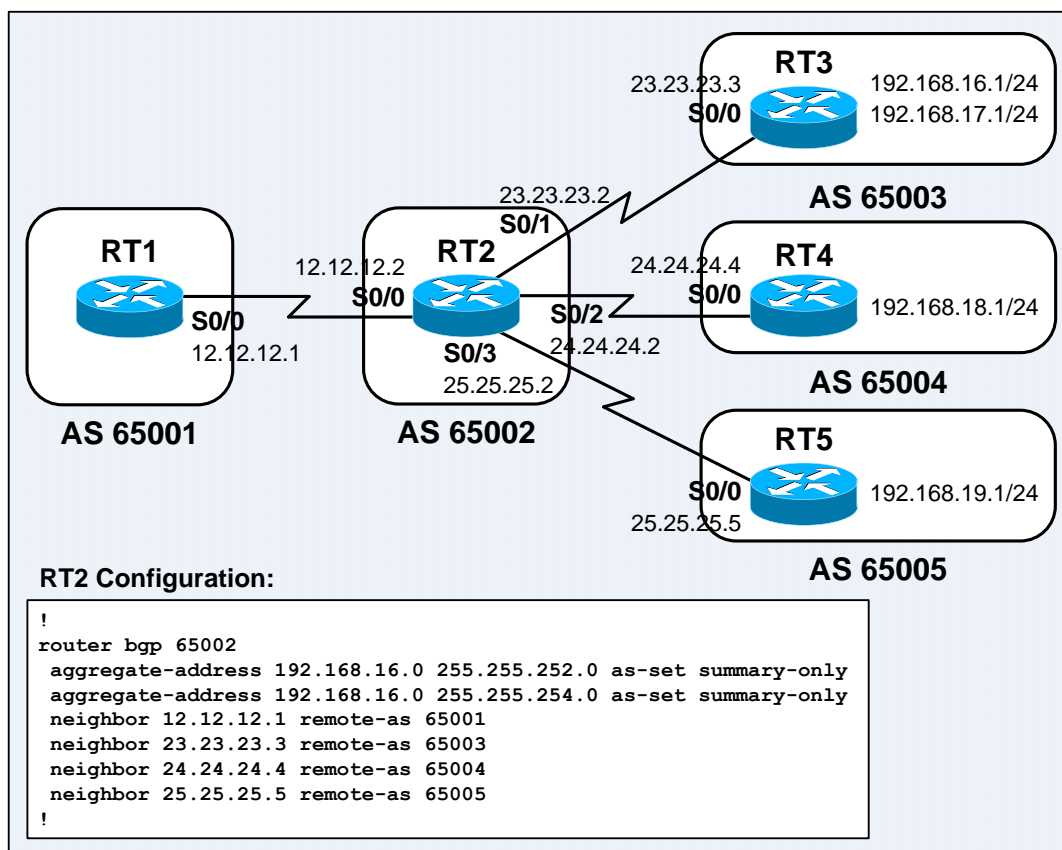


Figure 15-2: BGP Route Summarization using the **aggregate-address** Command

- Below shows the BGP tables on RT2 and RT1 when only the **summary-only** keyword is configured on RT2. Note that the more-specific routes are suppressed (the status code of **s**) and only the aggregate route is advertised to RT1.

```

RT2#sh ip bgp
BGP table version is 11, local router ID is 25.25.25.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 192.168.16.0/22  0.0.0.0                    32768 i
*> 192.168.16.0/23  0.0.0.0                    32768 i
s> 192.168.16.0     23.23.23.3           0             0 65003 i
s> 192.168.17.0     23.23.23.3           0             0 65003 i
s> 192.168.18.0     24.24.24.4           0             0 65004 i
s> 192.168.19.0     25.25.25.5           0             0 65005 i
RT2#
=====

RT1#sh ip bgp
BGP table version is 11, local router ID is 12.12.12.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 192.168.16.0/23  12.12.12.2           0             0 65002 i
*> 192.168.16.0/22  12.12.12.2           0             0 65002 i
RT1#

```

- Below shows the BGP tables on RT2 and RT1 when both the **as-set** and **summary-only** keywords are configured on RT2. Note that RT2 advertises the aggregate route 192.168.16.0/22 with the AS\_SET-type AS\_PATH attribute to include the combined unordered list of all ASes contained in all paths that are being summarized; also note that the ATOMIC\_AGGREGATE attribute is also set even the **as-set** keyword has been specified. AS\_SEQUENCE for AS 65002 is still included to allow receiving routers (RT1) to trace the path back to the aggregator (RT2).

```

RT2#sh ip bgp
BGP table version is 13, local router ID is 25.25.25.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network        Next Hop        Metric LocPrf Weight Path
*> 192.168.16.0/22 0.0.0.0          100   32768 {65003,65004,65005} i
*> 192.168.16.0/23 0.0.0.0          100   32768 65003 i
s> 192.168.16.0    23.23.23.3      0           0 65003 i
s> 192.168.17.0    23.23.23.3      0           0 65003 i
s> 192.168.18.0    24.24.24.4      0           0 65004 i
s> 192.168.19.0    25.25.25.5      0           0 65005 i
RT2#
=====
RT1#sh ip bgp
BGP table version is 13, local router ID is 12.12.12.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network        Next Hop        Metric LocPrf Weight Path
*> 192.168.16.0/23 12.12.12.2      0           0 65002 65003 i
*> 192.168.16.0/22 12.12.12.2      0           0 65002 {65003,65004,65005} i
RT1#sh ip bgp 192.168.16.0 255.255.252.0
BGP routing table entry for 192.168.16.0/22, version 13
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
   Not advertised to any peer
   65002 {65003,65004,65005}, (aggregated by 65002 25.25.25.2)
     12.12.12.2 from 12.12.12.2 (25.25.25.2)
       Origin IGP, metric 0, localpref 100, valid, external, atomic-aggregate, best
RT1#sh ip bgp 192.168.16.0 255.255.254.0
BGP routing table entry for 192.168.16.0/23, version 12
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
   Not advertised to any peer
   65002 65003, (aggregated by 65002 25.25.25.2)
     12.12.12.2 from 12.12.12.2 (25.25.25.2)
       Origin IGP, metric 0, localpref 100, valid, external, best
RT1#

```

- Below lists the 2 common types of AS\_PATH:
  - **AS\_SEQUENCE** that specifies the ordered list of ASNs towards a destination.
  - **AS\_SET** that specifies an unordered list of ASNs towards a destination.

The main purpose of AS\_PATH is used for loop prevention, in which a BGP router will discard EBGP-learned paths that list the local ASN in the AS\_PATH attribute; or else a loop may occur. When some AS\_PATH detail is lost upon aggregation, this increases the potential for a loop. Note that actually the AS\_PATH does not need to list the ASNs in any particular order; a receiving router only need to recognize whether its own ASN is included in the AS\_PATH, which is where AS\_SET comes into the picture.

## BGP Route Filtering

- The **neighbor** {ip-addr | peer-group-name} **distribute-list** {acl-num | acl-name | ip-prefix name} {in | out} BGP router subcommand filters the routing information to be advertised to a BGP neighbor or peer group based upon the specified access list.

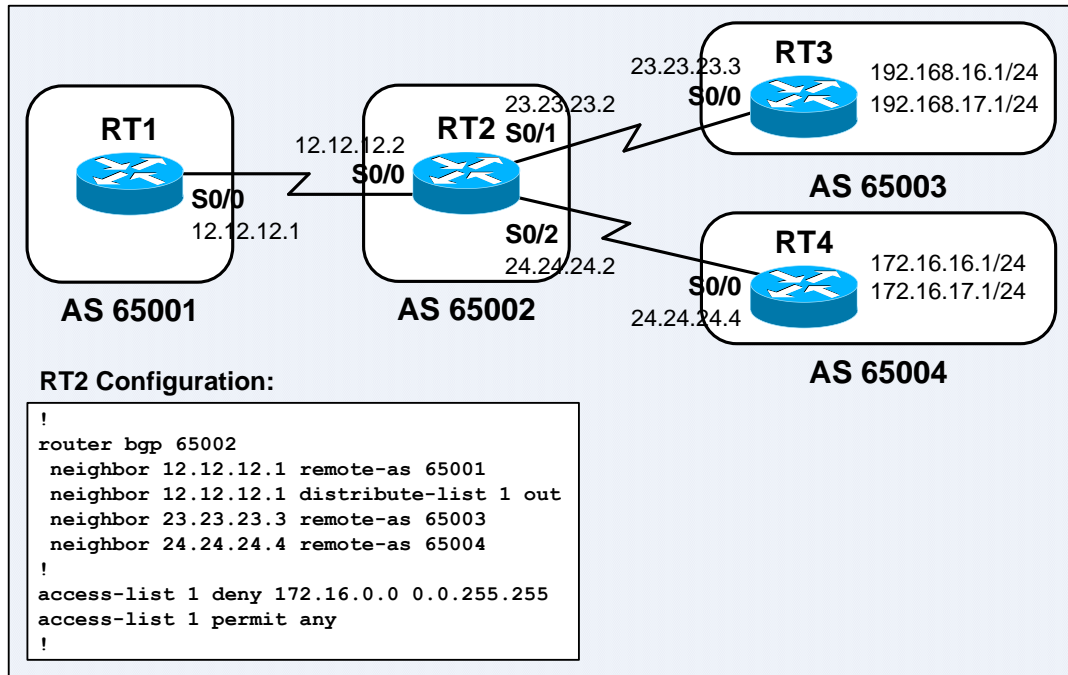


Figure 15-3: BGP Route Filtering using Distribute List

- Below shows that RT2 does not advertise the subnets that match 172.16.0.0 in the first 2 octets to RT1.

```

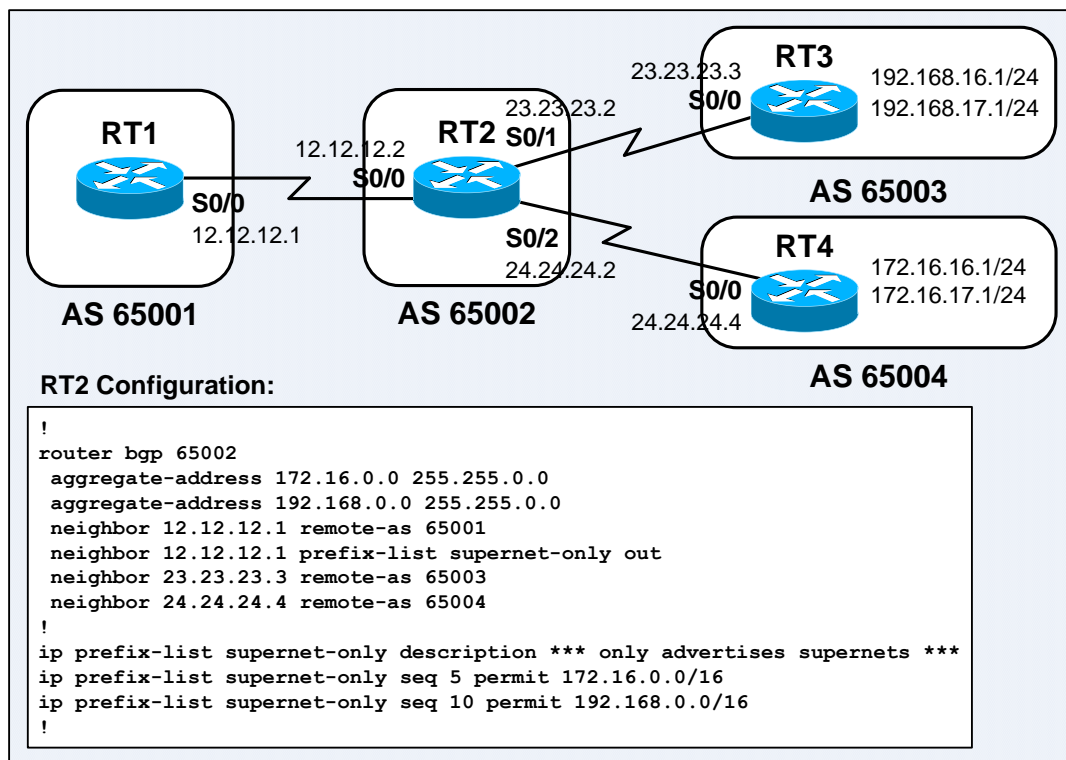
RT2#sh ip bgp
BGP table version is 5, local router ID is 24.24.24.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.16.0/24   24.24.24.4         0           0 65004 i
*> 172.16.17.0/24   24.24.24.4         0           0 65004 i
*> 192.168.16.0     23.23.23.3         0           0 65003 i
*> 192.168.17.0     23.23.23.3         0           0 65003 i
RT2#
RT2#sh access-lists
Standard IP access list 1
    10 deny    172.16.0.0, wildcard bits 0.0.255.255 (2 matches)
    20 permit any (2 matches)
RT2#
=====
RT1#sh ip bgp
BGP table version is 3, local router ID is 12.12.12.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 192.168.16.0     12.12.12.2         0           0 65002 65003 i
*> 192.168.17.0     12.12.12.2         0           0 65002 65003 i
RT1#

```

- Extended ACLs should be used to configure route filtering when using CIDR as they provide the granularity level that is necessary for advanced filtering of network addresses and masks.
- Note that distribute lists for BGP have been made obsolete by prefix lists in the Cisco IOS. Prefix lists are available in Cisco IOS Release 12.0 and later. However, note that both the **neighbor distribute-list** and the **neighbor prefix-list** BGP router subcommands cannot be applied upon a particular neighbor or peer group in any direction (inbound or outbound). They are mutually exclusive, in which only either one can be applied upon each direction.
- The **neighbor {ip-addr | peer-group-name} prefix-list {ip-prefix-name} {in | out}** BGP router subcommand filters the routing information to be advertised to a BGP neighbor or peer group based upon the specified prefix list.
- The prefix list sequence numbers are generated automatically. The **no ip prefix-list sequence-number** global configuration command disables the automatic generation of sequence numbers for entries in a prefix list.



**Figure 15-4: BGP Route Filtering using Prefix List**

- By implementing route filtering using prefix list on RT2, RT2 advertises the aggregate route to RT1 without using the **summary-only** keyword, and therefore no routes are being suppressed.

```

RT2#sh ip bgp
BGP table version is 7, local router ID is 24.24.24.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
* > 172.16.0.0      0.0.0.0           0         32768 i
* > 172.16.16.0/24  24.24.24.4        0         0 65004 i
* > 172.16.17.0/24  24.24.24.4        0         0 65004 i
* > 192.168.0.0/16  0.0.0.0           0         32768 i
* > 192.168.16.0    23.23.23.3        0         0 65003 i
* > 192.168.17.0    23.23.23.3        0         0 65003 i
RT2#

```

```

RT2#sh ip bgp neighbors 12.12.12.1 advertised-routes
BGP table version is 7, local router ID is 24.24.24.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.0.0       0.0.0.0           32768  i
*> 192.168.0.0/16  0.0.0.0           32768  i
RT2#
RT2#sh ip prefix-list
ip prefix-list supernet-only: 2 entries
   seq 5 permit 172.16.0.0/16
   seq 10 permit 192.168.0.0/16
RT2#
RT2#sh ip prefix-list detail
Prefix-list with the last deletion/insertion: supernet-only
ip prefix-list supernet-only:
  Description: *** only advertises supernets ***
  count: 2, range entries: 0, sequences: 5 - 10, refcount: 3
  seq 5 permit 172.16.0.0/16 (hit count: 1, refcount: 2)
  seq 10 permit 192.168.0.0/16 (hit count: 1, refcount: 1)
RT2#

```

- The **clear ip prefix-list** privileged command resets the hit counts shown on prefix list entries.
- An ISP always perform inbound filtering upon the routes learned from its customers or peers to restrict them to advertise only the networks that actually assigned to them in order to protect against address hijacking throughout the Internet, eg: the AS7007 incident on 25th Apr 1997.
- The earlier sections discuss BGP route filtering based upon prefixes. The following section starts with the discussion of regular expressions and followed by implementing **BGP filter list** that filter routes based upon AS paths.
- A **regular expression** is a formula for matching text strings according to a certain pattern and returns an answer of TRUE or FALSE – whether the expression describes the data, or it doesn't. Regular expressions can be used for filtering outputs of the **show** and **more** commands.
- A regular expression consists of 2 types of characters:
  - **Regular characters** that represents the characters to be matched.
  - **Control characters** (or metacharacters) that have special meanings.
 Control characters can be grouped into 3 types:
  - **Atom characters** (or atoms) – an atom is an independent control or placeholder character that defines or expands the regular characters that are before or after it. Some atoms can be standalone without regular characters.
  - **Multiplier characters** (or multipliers) – a multiplier follows an atom or a regular character to describe repetitions of the character immediately before it. Except the dot character (.), all atom characters must be grouped with regular characters before a multiplier is appended.
  - **Range characters** are brackets that specify a complete range.
- **Note:** Press **Ctrl+V** (escape sequence for Cisco IOS CLI) followed by **?** to input a question mark (**?**) into a regular expression.

- Below lists the common **atom characters** and their usages:

. (Period / Dot)	Matches any single character, including a blank space.
^ (Caret)	Matches the beginning character of the input string.
\$ (Dollar sign)	Matches the ending character of the input string.
_ (Underscore)	Matches a comma (,), underscore (_), left brace ({), right brace (}), left parenthesis (()), right parenthesis ()), the beginning of the input string (^), the end of the input string (\$), or a blank space.
(Pipe)	Operates as the Boolean OR operand that matches either of 2 strings.
\ (backslash)	Escape character that turns a succeeding control character that follows immediately into a regular character.

Below shows some simple examples of atoms:

^a.\$	Matches a string that begins with character <b>a</b> and ends with any single character. Ex: <b>ab, ax, a., a!, a0</b> , etc.
^100_	Matches <b>100, 100 200, 100 200 300, 100 300 400</b> , etc.
^100\$	Matches <b>100</b> only.
^100_200_	Matches <b>100 200, 100 200 300</b> , etc.
100\$ 200\$	Matches <b>100, 2100, 100 200, 200, 100 100, 1100 2200, 400 200</b> , etc.
^\(65000\) \$	Matches <b>(65000)</b> only.

- Below lists the common **multiplier characters** and their usages:

* (Asterisk)	Matches <b>zero or more</b> occurrences of the preceding character or pattern.
+ (Plus sign)	Matches <b>one or more</b> occurrences of the preceding character or pattern.
? (Question mark)	Matches <b>zero or one</b> occurrences of the preceding character or pattern.

A multiplier can be applied upon single- or multiple-character patterns.

To apply a multiplier upon a multi-character pattern, enclose the pattern in parenthesis.

Below shows some simple examples of multipliers:

abc*d	Matches <b>abd, abcd, abccd, abcccd, xabccde, abcde, xabcde, cabcd</b> , etc.
abc+d	Matches <b>abcd, abccd, abcccd, xabccde, abcde, xabcde, cabcd</b> , etc.
abc?d	Matches <b>abd, abcd, abcde, xabcde, cabcd</b> , etc.
a(bc)?d	Matches <b>ad, abcd, cad, cabcd, abcde, xabcde</b> , etc.
a(b.)*d	Matches <b>ad, abcd, abcde, xabcde, cad, cabcd, ab0d, ab0b0d, abxd, abxbxd</b> , etc.

- The brackets [ ] describe a range of single-character patterns; only one of the characters within the range will be matched. An exclusive match can be made specifying the caret character (^) at the start of the range to exclude (negate) all the characters within the range. The beginning and the ending characters of a range can also be specified using the dash or hyphen character (-).

Below shows some simple examples of multipliers:

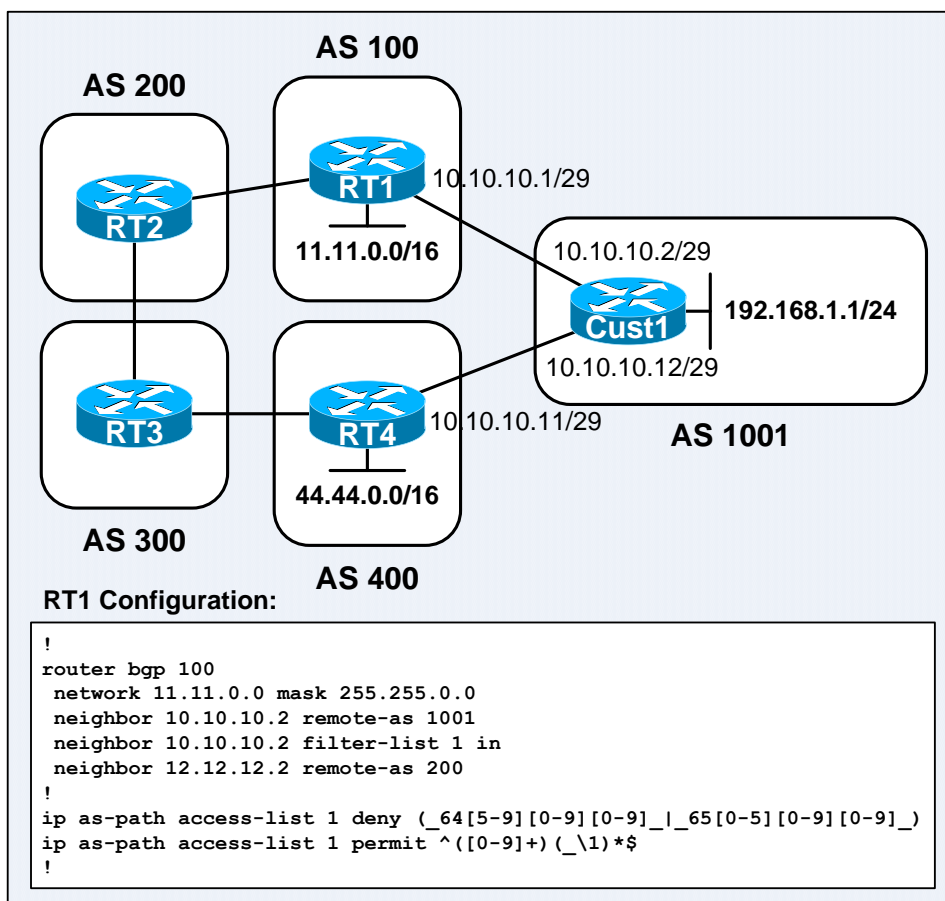
[aeiouAEIOU]	Matches any string that contains <b>a, e, i, o, u, A, E, I, O, or U</b> . Ex: <b>a, aa, Aa, eA, x2u</b> , etc.
[a-c1-2]\$	Matches any string that ends with <b>a, b, c, 1, or 2</b> . Ex: <b>a, 1b, a1, 62, xv2</b> , etc.
[^act]\$	Matches any string that does not ends with <b>a, c, and t</b> . Ex: <b>d, efg*, low2, actor, path</b> , etc.
[^a-dA-D]	Matches any single character except <b>a, b, c, d, A, B, C, and D</b> .

- Regular expressions are used extensively in pattern matching when defining BGP policies, eg: **AS Path Filtering**. The AS\_PATH attribute lists the ASNs that a prefix has traversed, in reverse order and separated by spaces. The **show ip bgp regexp {regexp} EXEC** command displays BGP routes matching the regular expression upon the AS\_PATH attribute information.

- Below shows some examples of common AS\_PATH pattern matching using regular expressions:

<b>^\$</b>	Matches an empty AS path list – all local routes that originated from the local AS.
<b>.*</b>	Matches all paths.
<b>^100\$</b>	Matches all paths that start and end with AS 100, which are updates that only originated and sent from AS 100, no AS path prepending and no intermediary. The ^ and \$ characters define the border of the input string.
<b>_100\$</b>	Matches all routes that originated from AS 100, including those prepended with 100, eg: <b>100, 100 100, 200 100</b> , etc. The underscore character limits the AS number to be exactly <b>100</b> and not <b>1100</b> or <b>2100</b> .
<b>^100_</b>	Matches all paths that received from neighboring AS 100, including those prepended with 100, eg: <b>100, 100 100, 100 200, 100 200 200, 100 300 200</b> , etc.
<b>_100_</b>	Matches all paths that passed through AS 100 but not necessarily originated from or received directly from AS 100, eg: <b>100, 100 200, 200 100, 200 100 300</b> , etc.
<b>^100(_100)*(_200)*\$</b>	Matches all paths from AS 100 and its immediate neighbor AS 200, eg: <b>100, 100 100, 100 200, 100 200 200, 100 100 100 200 200</b> , etc.

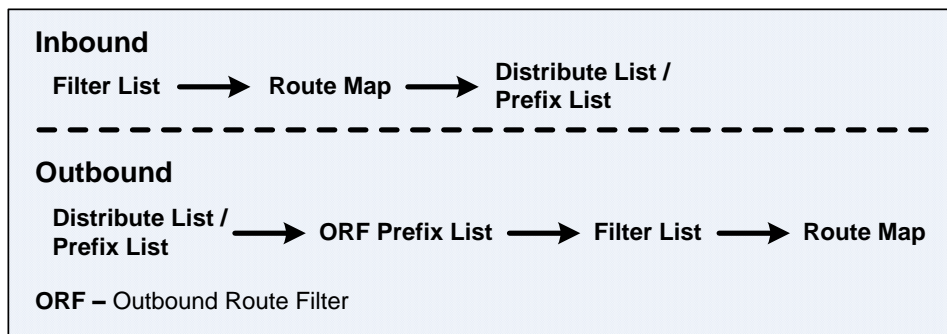
- Filtering based on the AS\_PATH information is quite effective as it filters the routing updates that contain certain AS\_PATH; compared to inefficiency and complexity of matching thousands of prefixes individually and defining the paths to become members of BGP communities.
- **AS path filter lists** are used to filter prefixes based on the BGP AS\_PATH attribute information. An AS path filter list entry matches the AS\_PATH attribute pattern using a regular expression along with a permit or deny action. The **ip as-path access-list {acl-num} {permit | deny} {as-path-regex}** global configuration command configures an autonomous system path filter. The **permit** and **deny** keywords permit and deny advertisement based on matching conditions. AS path filters can be applied upon inbound and outbound BGP route advertisements using the **neighbor {ip-addr | peer-group-name} filter-list {as-path-filter-acl-num} {in | out}** BGP router subcommand. The AS path access list number is in the range of 1 to 500, inclusive. The **show ip bgp filter-list {acl-num}** EXEC command displays BGP routes that conform to a specified filter list.
- Many disastrous Internet outages have occurred over the past due to misconfigurations and software bugs in BGP implementations; false updates, de-aggregation, and prefix hijacking in which an AS announces the routes that it does not have would eventually cause blackholing.
- Any serious and security-aware ISP should perform some sanity checks upon the BGP routes received from its customer instead of blindly accepting them. This would prevent a multi-homed customer from advertising BGP routes between its service providers and becomes a transit AS. The simple filter – all BGP routes from a customer should contain only its ASN in the AS path.



**Figure 15-5: BGP Route Filtering using AS Path Filter List**

- The initial non-scalable approach would be accept only the paths that have exactly the ASN of the customer in the AS path, eg: **ip as-path access-list 1 permit ^65001\$**.  
A more generic approach is recognize a single ASN using **^[0-9]+\$** instead.  
The **[0-9]** pattern matches a single digit.  
The **[0-9]+** pattern matches one or more digits – a number.
  - Both filters above have a problem; they fail when the customer implements AS path prepending. The filter must be able to recognize a single number that is potentially repeating multiple times. Below lists the filters that are applicable for this sample scenario to achieve the mentioned task.  
**ip as-path access-list 1 permit ^(1001\_)+\$**  
**ip as-path access-list 1 permit ^1001(\_1001)\*\$**  
**ip as-path access-list 1 permit ^(1001)(\_\1)\*\$**
- And the more generic filter is **ip as-path access-list 1 permit ^([0-9]+)(\_\1)\*\$**  
The **\1** in the filter is used for pattern recall, in which it can match a pattern recognized earlier in the regular expression – the 1st ASN in the AS path as matched by the regular expression within the 1st parenthesis.
- The 1st entry in the AS path filter list above is used to deny all paths that contain private ASNs.





**Figure 15-6: BGP Filter Processing Order**

- When multiple filters (distribute list, prefix list, filter list, route map) are configured for a neighbor, the filters are processed in a specific order as shown in the figure above. For inbound updates, the filter list is processed first, followed by the route map. The distribute list or prefix list is processed last. For outbound updates, the distribute list or prefix list is processed first, followed by the prefix list received via BGP Outbound Route Filtering, and then the filter list. The route map is processed last.
- An update has to pass through all the filters. One filter does not take precedence over another. If any filter does not match, the update is not permitted.  
**Ex:** An inbound update is permitted by a filter list and a route map but is denied by a prefix list, the update is denied. The same rule applies upon the outbound updates.
- When a policy is configured for a neighbor but the policy is not defined, the default behaviors are:
  - **For distribute lists and prefix lists, permit any.** Routes will be advertised to a neighbor configured with an undefined distribute list or prefix list.
  - **For filter lists and route maps, deny any.**  
**Caution:** No routes will be advertised to; and the advertised routes will be withdrawn from a neighbor configured with an undefined filter list or route map. Routes that are not matched by a filter list or route map will also be denied and not being advertised to; or withdrawn from a neighbor.
- BGP communities provide a way to group and tag prefixes, which then allow routers to match and followed by setting and manipulating various BGP path attributes upon them for some tasks, eg: route filtering. It is inefficient to implement distribute lists and prefix lists for large networks with complex routing policies, eg: individual neighbor statements along with the corresponding access lists and prefix lists must be configured for each neighbor on each router involved.
- Route maps provide the only mechanism to set BGP communities through the **set community** {*comm-value* [*comm-value*...] [**additive**] | **none**} route map configuration subcommand. The community value can be entered as a decimal number or in the AA:NN format. Communities are shown as numeric numbers by default. It is recommended to implement the **ip bgp-community new-format** global configuration command to display BGP communities in the AA:NN format to conform with RFC 1997. This command only affects the format in which BGP communities are displayed; it does not affect communities and community exchange.
- Note that the **set community** command removes existing communities attached to a prefix and replaces them with the new set of communities unless the optional **additive** keyword is used to specify that a community is to be added to the already existing communities. The **none** keyword removes the communities from the prefixes that are being matched and processed by the route map.

- The **neighbor** {*ip-addr* | *peer-group-name*} **route-map** {*tag*} {**in** | **out**} BGP router subcommand applies a route map upon incoming or outgoing updates. The **neighbor** {*ip-addr* | *peer-group-name*} **send-community** BGP router subcommand specifies that the BGP COMMUNITY path attribute should be send to a BGP neighbor. BGP community exchange is not enabled by default; in which communities are removed in outgoing BGP updates.
- The **ip community-list** {*comm-list-num* | {[**standard** | **expanded**] *comm-list-name*}} {**permit** | **deny**} {*comm-value* [*comm-value*...]} global configuration command creates a community list for BGP to perform policy-based access control. The community list number is in the range of 1 to 99 for standard community lists; and 100 to 500 for expanded community lists.
- Standard community lists are used to configure well-known communities and specific community numbers. A maximum of 16 communities can be configured in a standard community list. Trailing communities that exceed the limit will be omitted. Expanded community lists are used to filter communities using a regular expression.
- When multiple community values are configured in a single community list statement, a logical AND processing will be performed – all the community values must be matched. When multiple community values are configured in separate community list statements, a logical OR processing will be performed – the first list that matches a condition is processed.
- The **match community** {*comm-list-num* | *comm-list-name*} [**exact-match**] route map subcommand is used to match a BGP COMMUNITY attribute upon a value in a community list. The optional **exact-match** keyword indicates that an exact match is required, in which all of the communities and only those communities in the community list(s) must be present in the BGP COMMUNITY attribute of a path.

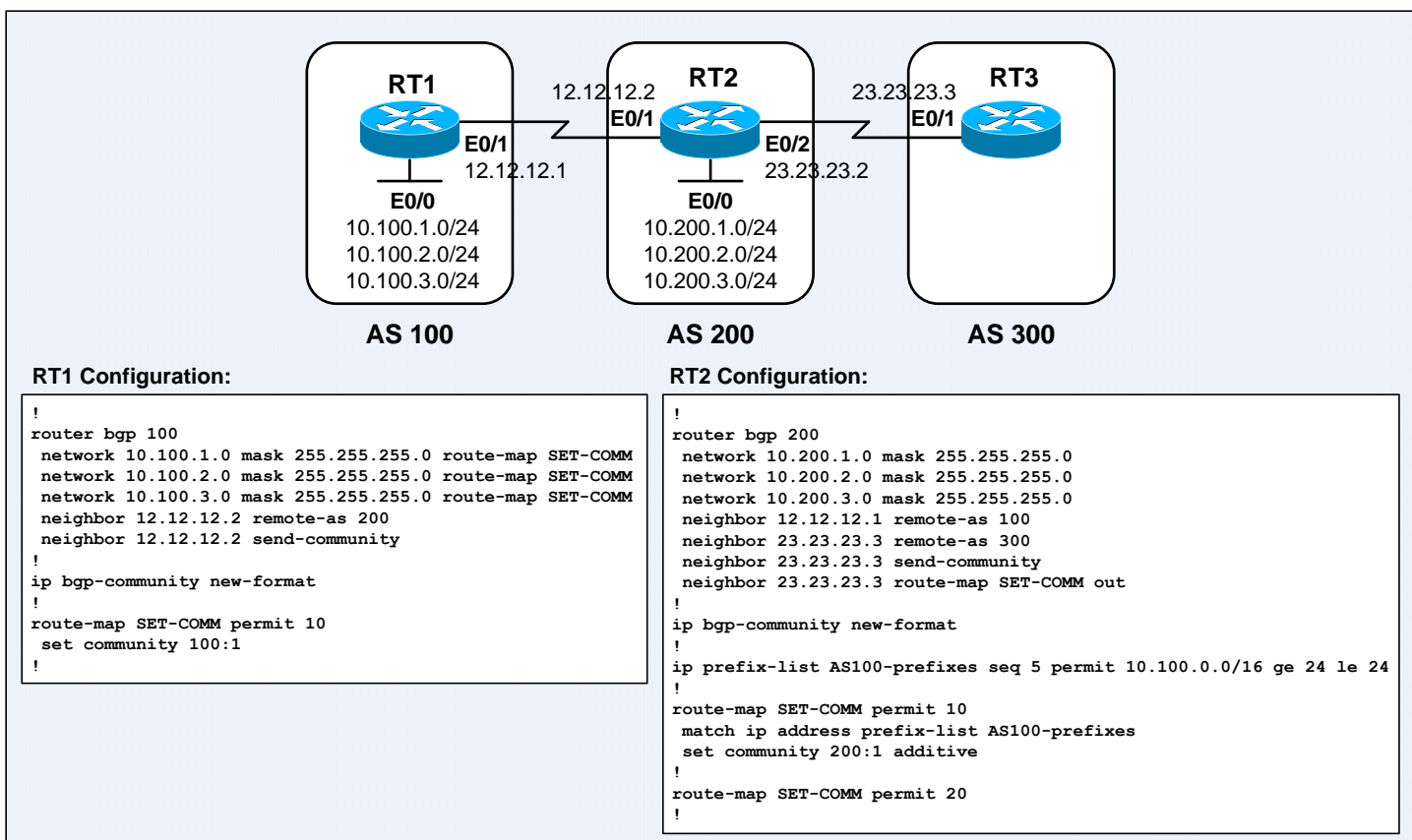


Figure 15-7: BGP Communities

- Below shows the configuration on RT3 and the communities set upon the routes originated from AS 100 and AS 200. Any route whose COMMUNITY attribute matches community list 1 has its WEIGHT attribute set to 100. Community list 1 matches routes with a community value of 100:1 – all the routes that originated from AS 100.

```

!
router bgp 300
  no synchronization
  bgp log-neighbor-changes
  neighbor 23.23.23.2 remote-as 200
  neighbor 23.23.23.2 route-map CHK-SET-COMM in
  no auto-summary
!
ip bgp-community new-format
ip community-list 1 permit 100:1
ip community-list 2 permit internet
!
route-map CHK-SET-COMM permit 10
  match community 1
  set weight 100
!
route-map CHK-SET-COMM permit 20
  match community 2
!
=====
RT3#sh ip bgp
BGP table version is 7, local router ID is 23.23.23.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 10.100.1.0/24    23.23.23.2                100  200 100 i
*> 10.100.2.0/24    23.23.23.2                100  200 100 i
*> 10.100.3.0/24    23.23.23.2                100  200 100 i
*> 10.200.1.0/24    23.23.23.2                 0     0 200 i
*> 10.200.2.0/24    23.23.23.2                 0     0 200 i
*> 10.200.3.0/24    23.23.23.2                 0     0 200 i
RT3#
RT3#sh ip bgp 10.100.1.0
BGP routing table entry for 10.100.1.0/24, version 2
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
   Not advertised to any peer
   200 100
     23.23.23.2 from 23.23.23.2 (23.23.23.2)
       Origin IGP, localpref 100, weight 100, valid, external, best
       Community: 100:1 200:1
RT3#
RT3#sh ip bgp 10.200.1.0
BGP routing table entry for 10.200.1.0/24, version 5
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
   Not advertised to any peer
   200
     23.23.23.2 from 23.23.23.2 (23.23.23.2)
       Origin IGP, metric 0, localpref 100, valid, external, best
RT3#

```

## BGP Route Reflection

- The original BGPv4 specification (RFC 1771 – A Border Gateway Protocol 4, BGP-4) does not define an intra-AS loop prevention mechanism; routers were therefore prohibited from propagating routes received from an IBGP peer to another IBGP peer; and hence requiring full-mesh IBGP sessions to be established between all BGP routers within an AS. Full-mesh IBGP is not scalable; the number of TCP sessions established for the BGP routers within an AS can be calculated using the formula  $\frac{n(n-1)}{2}$ , in which  $n$  is the number of routers. The amount of routing update traffic exchanged between IBGP peers can consume a significant amount of bandwidth and impact the availability of the network resources for applications.
- RFC 1966 – BGP Route Reflection – An alternative to full mesh IBGP was later released to introduce the implementation of route reflection as the alternative to full-mesh IBGP sessions. However, it defined that a BGP route reflector (RR) is not allowed to send an update received from a BGP route reflector client (RRC) back to the same route reflector client. As a result, the RR could not reuse the outbound BGP updates sent to a RRC for other RRCs and had to treat each RRC individually, resulting in significant performance degradation when a RR is associated with many RRCs and need to maintain different sets of outbound BGP updates for different RRCs.
- RFC 4456, which obsoletes both RFC 1966 and RFC 2796 has relaxed the update propagation rule mentioned above by defining that a RR can now sends the same update to all RRCs, even if the original route was received from an RRC. The BGP ORIGINATOR\_ID and CLUSTER\_LIST attributes that were originally defined in RFC 1966 are used by the RRCs to detect route propagation loops. This relaxed update propagation rule allows a RR to group all the RRCs into a single update group, and generates a single set outbound BGP updates for all RRCs, resulting in significant performance improvements.  
**Note:** The latest BGP standard as defined in RFC 4271 includes references to BGP route reflection.  
**Note:** Cisco IOS Release 12.3(4)T and later implemented the BGP Dynamic Update Peer-Groups feature that conforms to RFC 4456.
- BGP route reflection modifies the BGP split-horizon rule by allowing the BGP router configured as the RR to propagate or reflect routes learned by IBGP to other IBGP peers. BGP route reflection greatly reduces the number of IBGP sessions and the routing update traffic.

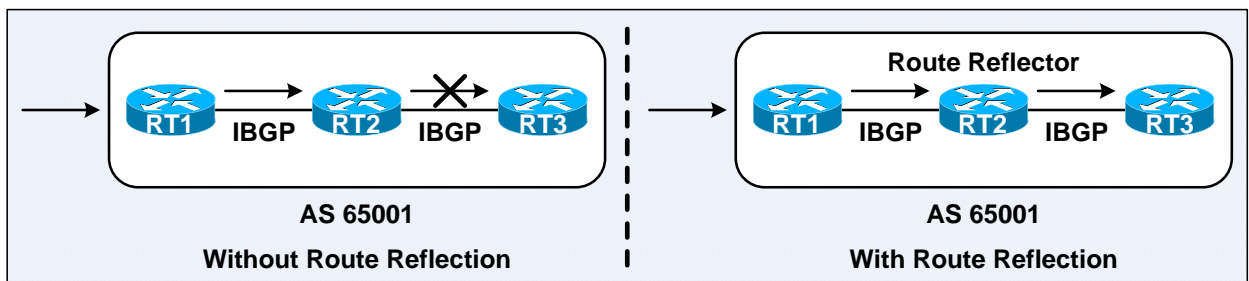


Figure 15-8: BGP Route Reflection

- Route reflection is often used by ISPs when the numbers of internal routers and the **neighbor** statements become excessive. The administrative overhead for maintaining full-mesh IBGP networks is the **neighbor** statement must be added on all routers each time a new peer is added. An AS can have multiple route reflectors, both for the redundancy purpose and grouping to further reduce the number of IBGP sessions.
- Route reflection does not affect the packet forwarding paths; but only the routing update distribution paths. However, routing loops can occur if route reflectors are configured incorrectly, eg: when 2 route reflector are configured to treat each other as route reflector clients.

- Migrating to route reflectors involves a minimal configuration and does not need to be configured on all routers at once, as normal routers can coexist with route reflectors within an AS, in which they are simply normal IBGP peers with the route reflectors. RRCs should not peer with IBGP routers outside their associated cluster.
- The AS\_PATH attribute is used for loop prevention for BGP route advertisements across ASes. The ORIGINATOR\_ID, CLUSTER\_ID, and CLUSTER\_LIST help prevent routing loops when implementing route reflection.
- A **cluster** is referred to as the combination of the RR and its clients. IBGP peering between the clients is no longer required, as the route reflector replicates routing updates between the clients. There is no defined limit for the number of RRCs that a RR may have; it is constrained by the processing capability of the RRs.
- The route reflector will create the 4-byte ORIGINATOR\_ID, an optional non-transitive BGP attribute that indicates the BGP Router ID of the originator of a route. When an update arrives back to the originator due to bad RR design and implementation, the originator discards it.
- Usually a cluster has a single RR and the cluster is identified using the ORIGINATOR\_ID. A cluster can have multiple RRs to increase redundancy and avoid single point of failure, in which all the RRs in the cluster must be assigned with a Cluster ID that allows the RRs in the cluster to recognize and discard updates originated from other RRs in the same cluster as well as reduces the number of updates that need to be stored in the BGP routing tables.
- A **route reflector cluster list** is a sequence of Cluster IDs that a route has passed within an AS. When a RR replicates an update from its client to non-clients outside the cluster, it appends the local Cluster ID upon the cluster list; it creates a cluster list if the update does not contain one. A RR is able to notice whether an update is looped back into the same cluster due to bad design. If the local cluster ID is found in the cluster list of an update, the update is ignored and discarded.
- When implementing route reflection within an AS, the IBGP routers can be divided into multiple groups that known as clusters, each having at least a RR and a few clients. Multiple RRs can exist in a cluster for redundancy purpose. The RRs from different clusters must be fully meshed through IBGP to ensure that all routes learned through EBGP are propagated throughout the AS. An IGP is still used just as it was before to advertise local routes and EBGP next-hop addresses.

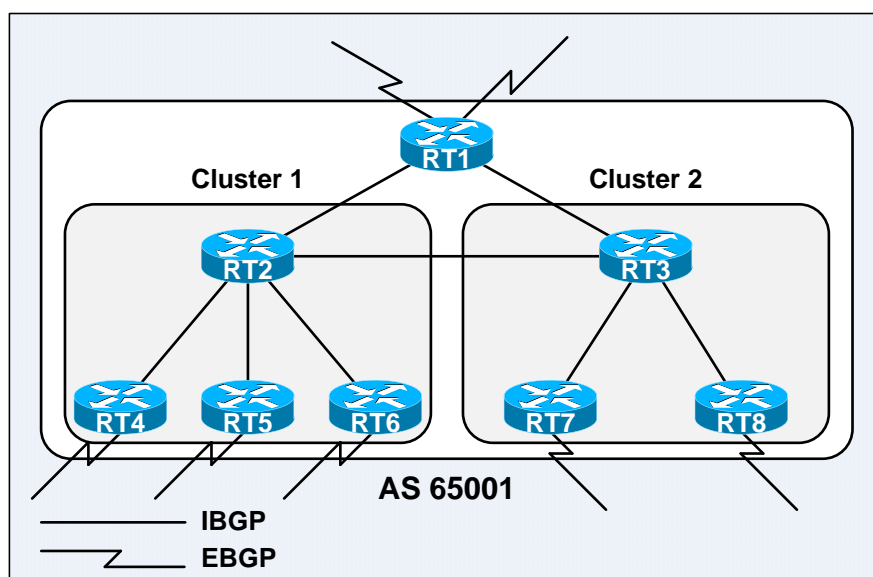


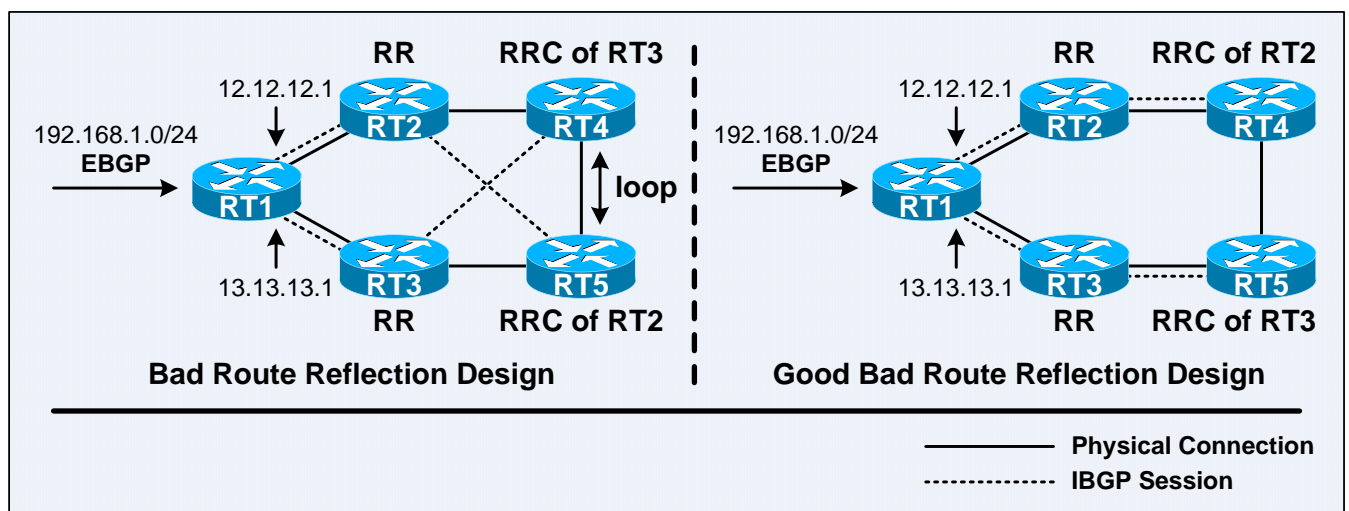
Figure 15-9: BGP Route Reflection Design

- The figure above shows an example of a BGP route reflection design. RT2, RT4, RT5, and RT6 form a cluster; RT3, RT7, and RT8 form another cluster. RT2 and RT3 are route reflectors. RT1, RT2, and RT3 are fully meshed through IBGP; while routers within a cluster are not fully meshed.
- A route reflector propagates an update depending upon the type of peer that sent the update:
  - If the update is received from a non-client peer, it sends the update to all its RRCs only.
  - If the update is received from a RRC, it sends the update to all RRCs and non-client peers. <sup>[1]</sup>
  - If the update is received from an EBGP peer, it sends the update to all RRCs and non-client peers. <sup>[1]</sup>

[1] – Non-client peers including route reflectors in other clusters.

**Note:** RRs still send updates to their EBGP peers as normal as before implementing route reflection.

- The first consideration when migrating to implement route reflection is which routers should be the RRs and which should be the RRCs. The design decision lies upon following the physical topology to ensure that both the physical and logical packet forwarding paths are the same. Route reflection design and implementation that does not follow the physical topology, in which RRCs are configured to associate with non-directly connected RRs, may result in routing loops.



**Figure 15-10:** Bad and Good Route Reflection Designs

- The left portion of the figure above demonstrates how routing loops occur when route reflection is implemented without following the physical topology. RT1 is a RRC for both RRs, RT2 and RT3. The following happens in this bad route reflection design:
  - RT4 learns that the next-hop to reach 192.168.1.0/24 is 13.13.13.1, as learnt from its RR – RT3.
  - RT5 learns that the next-hop to reach 192.168.1.0/24 is 12.12.12.1, as learnt from its RR – RT2.
  - RT4 forwards a packet destined to 192.168.1.0/24 through RT5 based on the best IGP route to the next-hop – 13.13.13.1.
  - RT5 forwards a packet destined to 192.168.1.0/24 through RT4 based on the best IGP route to the next-hop – 12.12.12.1.
  - This is a routing loop.
- The following happens in the good route reflection design as shown in the right portion of the figure:
  - RT4 learns that the next-hop to reach 192.168.1.0/24 is 12.12.12.1, as learnt from its RR – RT2.
  - RT5 learns that the next-hop to reach 192.168.1.0/24 is 13.13.13.1, as learnt from its RR – RT3.
  - RT4 forwards a packet destined to 192.168.1.0/24 through RT2 based on the best IGP route to the next-hop – 12.12.12.1. RT2 then forwards the packet to RT1.
  - RT5 forwards a packet destined to 192.168.1.0/24 through RT3 based on the best IGP route to the next-hop – 13.13.13.1. RT3 then forwards the packet to RT1.
  - There is no routing loop.

- When implementing multiple route reflectors for redundancy purpose, it is really important to complement logical redundancy with physical redundancy; and it does not make sense to build route reflector redundancy if the physical redundancy itself does not exist!
- Migrating to implement route reflection is easy, configure one RR at a time, and then remove the redundant IBGP sessions between the RRCs. The configuration only needs to be implemented on those routers intended to be route reflectors. It is recommended to configure one RR per cluster.
- Route reflection is not recommended for every topology, it is recommended only for ASes that have a large number of IBGP mesh. The route reflection concept introduces processing overhead on the route reflector server and might introduce routing loops and routing instability if configured incorrectly. Imposing complex techniques on situations that do not really need them could hurt more than help! The route reflection function is being performed only on the route reflector, all RRCs and non-clients are normal IBGP peers that have no concept and never notice that route reflection is in effect. RRCs are considered as such only because the RR treats them as RRCs.
- Note that route reflectors propagate only the best path to a particular destination to its RRCs – when a RR learns multiple paths to the same destination from multiple RRCs, only a single best path that is selected by the best path algorithm will be propagated to other RRCs. Therefore, the number of paths available to reach a particular destination after implemented route reflection will likely be lower than that of an IBGP full-mesh configuration.
- The **neighbor** {*ip-addr* | *peer-group-name*} **route-reflector-client** BGP router subcommand configures a router as a BGP route reflector and configures the specified neighbor as its client. Use the **bgp cluster-id** {*cluster-id*} BGP router subcommand to assign a Cluster ID on all RRs in a RR cluster when the RR cluster has one or more RRs. The maximum 4-byte Cluster ID can be specified in dotted or decimal format. The Cluster ID cannot be changed after the RRCs have been configured; unless the RR supports the Route Refresh capability that can readvertise it. **Note:** Cluster ID is obsolete! Redundant route reflection setups where multiple RRs serve the same set of RRCs can lead to partial connectivity problems when multiple IBGP sessions are disrupted. The revised BGP best path selection algorithm ensures that a RR in a cluster always prefers a route from a client with a shorter CLUSTER\_LIST over a reflected route; making the Cluster ID obsolete. This allows RRCs to peer with RRs or RRCs in other clusters. The Cluster ID should not be used in new network designs to increase the resilience of a network.  
The classical case for route reflector redundancy described above groups the route reflectors in the same cluster with all the RRs configured with the same Cluster ID and requires all the clients to have IBGP sessions with both RRs. The modern approach is to have only 1 route reflector per cluster, and only the RRCs that want the redundancy establish an IBGP session with all the RRs.
- **Note:** The NEXT\_HOP attribute is retained intact when a prefix is reflected by a RR. The **neighbor next-hop-self** BGP router subcommand, when configured on a route reflector, affects only the next-hop of EBGP-learned routes; the next-hop of reflected IBGP routes will be retained intact.
- Previously, route reflectors could only be used only in conjunction with peer groups when all the route reflector clients within a cluster were fully meshed due to some update exchange issues. Fortunately, this full-mesh requirement has been removed and route reflector clients of a route reflector can now be configured under a peer group and are not required to be fully meshed.



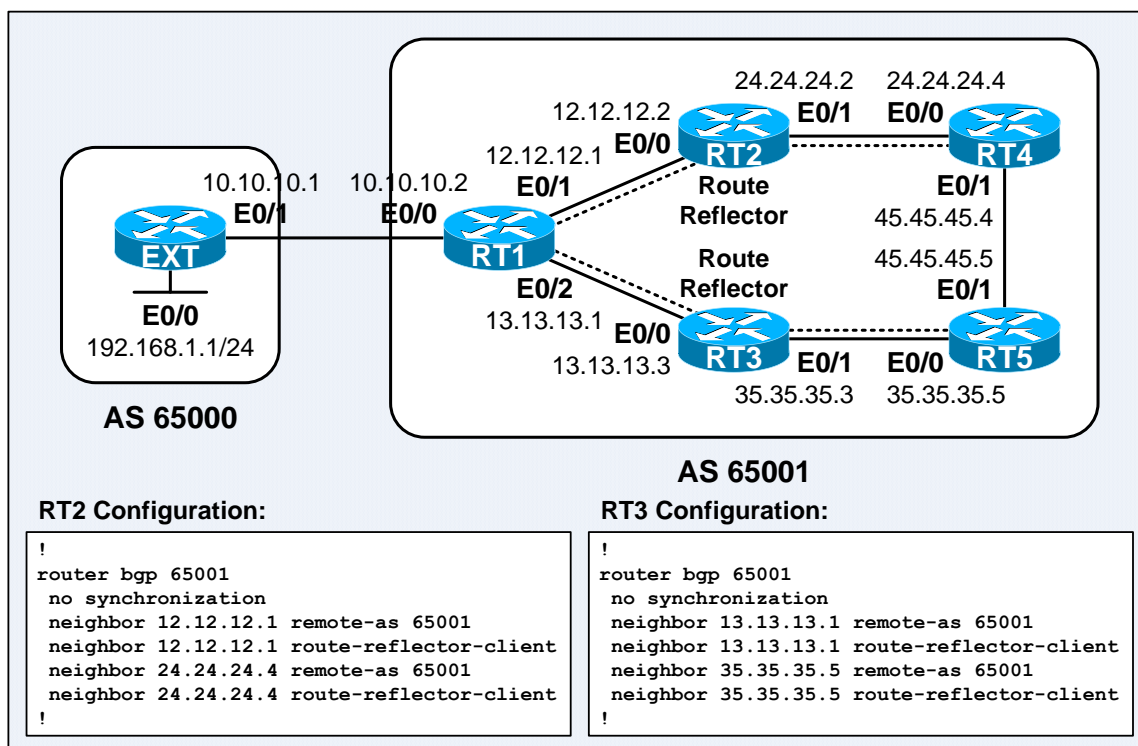


Figure 15-11: BGP Route Reflection Configuration

- Below shows the BGP and IP routing tables on RT4 and RT5 after implemented RT2 and RT3 as RRs. **Note:** RT1 has been configured with the **neighbor next-hop-self** command for RT2 and RT3.

```
RT4#sh ip bgp
BGP table version is 2, local router ID is 45.45.45.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*>i192.168.1.0      12.12.12.1              0   100    0 65000 i
RT4#
RT4#sh ip bgp 192.168.1.0
BGP routing table entry for 192.168.1.0/24, version 2
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
   Not advertised to any peer
   65000
     12.12.12.1 (metric 20) from 24.24.24.2 (24.24.24.2)
       Origin IGP, metric 0, localpref 100, valid, internal, best
       Originator: 13.13.13.1, Cluster list: 24.24.24.2
RT4#
=====
RT5#sh ip bgp
BGP table version is 2, local router ID is 45.45.45.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*>i192.168.1.0      13.13.13.1              0   100    0 65000 i
RT5#
```



*This page is intentionally left blank*

## Advanced BGP – Path Manipulation and Multihoming

- Manipulating path selection criteria affects the inbound and outbound traffic policies of an AS. Unlike IGP, BGP was never designed to choose the fastest routing path. It was designed to manipulate traffic flow to maximize or minimize the bandwidth usage. The figure below shows a common situation that can result when implementing BGP without any policy manipulation.

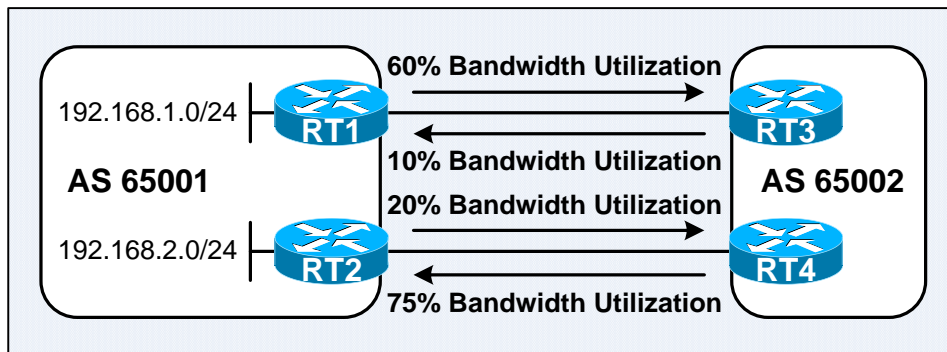


Figure 16-1: BGP Network without Policy Manipulation

- Using default configuration for BGP path selection may cause uneven bandwidth utilization. RT1 in AS 65001 is using 60% of its outbound bandwidth to RT3 in AS 65002; but RT2 is using only 20% of its outbound bandwidth to RT4. No manipulation is needed if this utilization is acceptable to the organization. However, when the load that averages at 60% is bursts above 100% of the bandwidth, it can result in packet loss, higher latency, and higher CPU utilization. It makes sense to divert some of the traffic to another path when another link is able to reach the same destinations and is not heavily utilized or underutilized. Manipulating the local preference attribute can change the outbound path selection from AS 65001.
- Determining the traffic to be manipulated first starts with performing a traffic analysis upon the outbound traffic to examine the most heavily visited IP addresses, web pages, and domain names through the network management accounting records or firewall accounting information. After this information has been determined, route maps can then be implemented to manipulate the local preference attribute for particular destinations to influence the BGP routers within an AS to forward packets destined to them through different edge routers and different paths.
- After some path manipulation has been performed, the outbound utilization for RT2 may increase from 20% to 40%, and the outbound utilization for RT1 may decrease from 60% to 40%. This change will make the outbound utilization on both links to be more balanced.
- Just as there was an outbound utilization issue from AS 65001, a similar problem can occur for inbound utilization. When the inbound utilization towards RT2 is much higher than the inbound utilization towards RT1, the BGP MED attribute can manipulate how traffic enters AS 65001. MED is considered a **recommendation**, as it is not considered until later in the BGP path selection process than the local preference. The receiving AS can override it by manipulating other attributes that are considered before the MED is evaluated.
- If an outbound or inbound load averages less than 50%, path manipulation might not be needed. However, as soon as a link starts to spike up to its capacity for an extended period of time, either more bandwidth should be needed or path manipulation should be considered.
- Without path manipulation, the most common criteria for path selection is the shortest AS path.

## Manipulating the Local Preference Attribute

- The local preference attribute is used only within an AS between IBGP routers to determine the preferred path to leave the AS to an outside destination. BGP routers will ignore the local preference attribute sent from EBGP neighbors due to software implementation bugs. The local preference for the routes advertised to IBGP neighbors is set to 100 by default; the highest value is preferred when comparing the local preferences for different routes.
- The **bgp default local-preference** BGP router subcommand defines the local preference value for all EBGP routes received by the router on which this command is configured. The value ranges from 0 to 4294967295, inclusive. The value is advertised to all IBGP neighbors.
- Manipulating the default local preference can have an immediate and dramatic effect upon the traffic flow leaving an AS. Thorough traffic analysis should be performed to understand its effects prior to implementing it, as it can cause a particular outbound link to be overutilized while leaving other outbound links to be underutilized! Route maps should often be implemented to set only certain destinations to have a higher local preference over others.

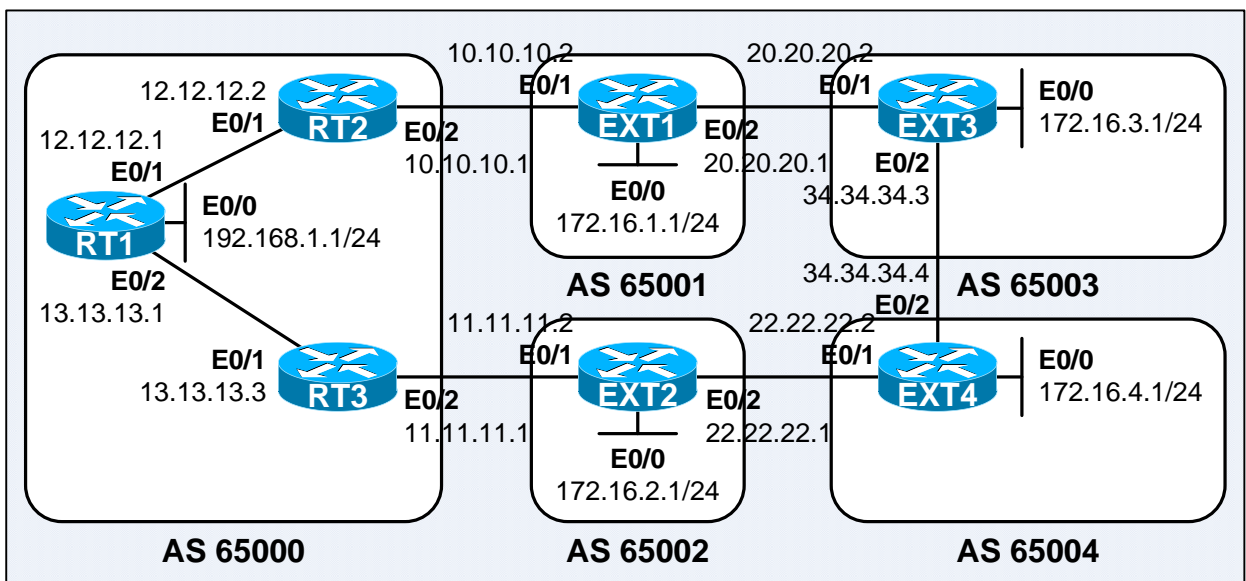


Figure 16-2: BGP Local Preference

- Below shows the BGP routing table on RT1 before manipulating the local preference attribute. RT1 acts as a route reflector. All routes have a weight of 0 and a default local preference of 100. The **neighbor next-hop-self** command is not implemented on RT2 and RT3.  
**Note:** RT2 and RT3 advertise only the best paths to destinations reside in other ASes to RT1.

```

RT1#sh ip bgp
BGP table version is 6, local router ID is 192.168.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*>i172.16.1.0/24    10.10.10.2         0      100     0 65001 i
*>i172.16.2.0/24    11.11.11.2         0      100     0 65002 i
*>i172.16.3.0/24    10.10.10.2         0      100     0 65001 65003 i
*>i172.16.4.0/24    11.11.11.2         0      100     0 65002 65004 i
*> 192.168.1.0      0.0.0.0            0                               32768 i
RT1#

```

- There are 2 loop-free paths available for AS 65000 to reach 172.16.3.0/24 and 172.16.4.0/24.

```

RT2#sh ip bgp
BGP table version is 8, local router ID is 12.12.12.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.1.0/24    10.10.10.2         0         100     0 65001 i
*>i172.16.2.0/24    11.11.11.2         0         100     0 65002 i
*> 172.16.3.0/24    10.10.10.2         0         100     0 65001 65003 i
* 172.16.4.0/24    10.10.10.2         0         100     0 65001 65003 65004 i
*>i                 11.11.11.2         0         100     0 65002 65004 i
r>i192.168.1.0      12.12.12.1         0         100     0 i
RT2#
=====
RT3#sh ip bgp
BGP table version is 8, local router ID is 13.13.13.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*>i172.16.1.0/24    10.10.10.2         0         100     0 65001 i
*> 172.16.2.0/24    11.11.11.2         0         100     0 65002 i
* 172.16.3.0/24    11.11.11.2         0         100     0 65002 65004 65003 i
*>i                 10.10.10.2         0         100     0 65001 65003 i
*> 172.16.4.0/24    11.11.11.2         0         100     0 65002 65004 i
r>i192.168.1.0      13.13.13.1         0         100     0 i
RT3#

```

- The following route map configuration is implemented on RT2 to manipulate the local preference for 172.16.4.0/24 to 200 (higher than the default of 100) for all BGP routers in AS 65000 to forward the packets destined to 172.16.4.0/24 through RT2 → EXT1 instead of RT3 → EXT2.

```

!
router bgp 65000
  no synchronization
  neighbor 10.10.10.2 remote-as 65001
  neighbor 10.10.10.2 route-map SET-LOCAL_PREF in
  neighbor 12.12.12.1 remote-as 65000
  no auto-summary
!
route-map SET-LOCAL_PREF permit 10
  match ip address 1
  set local-preference 200
!
route-map SET-LOCAL_PREF permit 20
!
access-list 1 permit 172.16.4.0 0.0.0.255
!

```

- The 1st route map **permit** statement matches and sets the local preference for 172.16.4.0/24. The 2nd route map **permit** statement that does not have any **match** or **set** clauses is similar to a **permit any** statement in an access list. Since there are no match conditions for other networks, they are permitted with their current settings without any manipulation.

- The route map is applied as an inbound route map for EXT1 to allow RT2 to process the routes received from EXT1 through the route map and manipulate the local preference accordingly. Implementing the route map as an outbound route map on RT2 for RT1 does not work if RT2 first learns the best path through RT3 and it does not have the chance to advertise the non-best path.
- Below shows the BGP routing tables on all the BGP routers in AS 65000 – RT1, RT2, and RT3. RT2 is now being selected over RT3 as the exit point of the AS towards 172.16.4.0/24.

```

RT1#sh ip bgp
BGP table version is 7, local router ID is 192.168.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*>i172.16.1.0/24    10.10.10.2         0      100      0 65001 i
*>i172.16.2.0/24    11.11.11.2         0      100      0 65002 i
*>i172.16.3.0/24    10.10.10.2         0      100      0 65001 65003 i
*>i172.16.4.0/24  10.10.10.2         0      200      0 65001 65003 65004 i
*> 192.168.1.0     0.0.0.0            0                32768 i
RT1#
=====

RT2#sh ip bgp
BGP table version is 8, local router ID is 12.12.12.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.1.0/24    10.10.10.2         0                0 65001 i
*>i172.16.2.0/24    11.11.11.2         0      100      0 65002 i
*> 172.16.3.0/24    10.10.10.2         0                0 65001 65003 i
*> 172.16.4.0/24  10.10.10.2         0      200      0 65001 65003 65004 i
r>i192.168.1.0     12.12.12.1         0      100      0 i
RT2#
=====

RT3#sh ip bgp
BGP table version is 8, local router ID is 13.13.13.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*>i172.16.1.0/24    10.10.10.2         0      100      0 65001 i
*> 172.16.2.0/24    11.11.11.2         0                0 65002 i
* 172.16.3.0/24     11.11.11.2         0                0 65002 65004 65003 i
*>i                10.10.10.2         0      100      0 65001 65003 i
*>i172.16.4.0/24  10.10.10.2         0      200      0 65001 65003 65004 i
*                  11.11.11.2         0                0 65002 65004 i
r>i192.168.1.0     13.13.13.1         0      100      0 i
RT3#

```

## Manipulating the MED Attribute

- The BGP Multiple Exit Discriminator (MULTI\_EXIT\_DISC, MED) attribute is used by an AS that tries to influence the entering point towards the AS from another AS when multiple paths exist between the 2 neighboring ASes. Since MED is being evaluated quite late in the BGP best path selection process, it can be overridden by other attributes and has no influence on the process.
- The MED for the routes originated from an AS and advertised to EBGPs is set to 0 by default; the lowest value is preferred when comparing the MEDs for different routes.
- The **default-metric** {*metric*} BGP router subcommand defines the metric value for routes redistributed into BGP through the **redistribute** router subcommand. The value ranges from 1 to 4294967295, inclusive. This value is actually the MED that is being evaluated during the BGP best path selection process. This metric is not set if the received route already has a MED value. MEDs are carried into an AS and used there, but are not being passed beyond the receiving AS, which means that MEDs are used only to influence traffic between 2 directly connected ASes.
- As like the default local preference, manipulating the default metric can have an immediate and dramatic effect upon the traffic flow entering an AS. Thorough traffic analysis should be performed to understand its effects prior to implementing it, as it can cause a particular inbound link to be overutilized while leaving other inbound links to be underutilized! Route maps should often be implemented to set only certain routes to have a higher or lower MED over others.
- When an EBGPs peer receives an update without a MED attribute, it must interpret its meaning. Cisco IOS treats routes without the MED attribute as having a MED of 0, making a route lacking the MED variable the most preferred. The IETF decided upon the missing MED should be treated as having a value of infinity, making a route lacking the MED variable the least preferred. Implement the **bgp bestpath med missing-as-worst** BGP router subcommand to configure a Cisco IOS BGP router to conform to the IETF standard – routes without the MED attribute are treated as having a MED of 4'294'967'295.

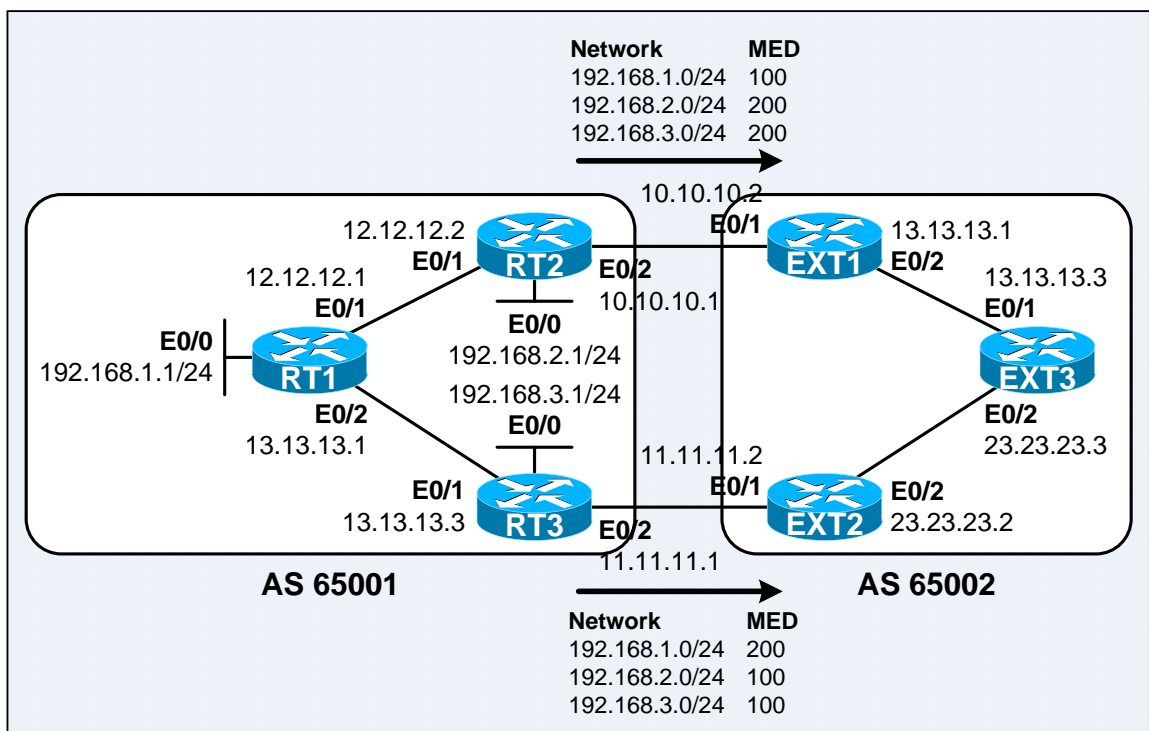


Figure 16-3: BGP MED

RT2 Configuration:	RT3 Configuration:
<pre> ! router bgp 65001 no synchronization network 192.168.2.0 neighbor 10.10.10.2 remote-as 65002 neighbor 10.10.10.2 route-map SET-MED out neighbor 12.12.12.1 remote-as 65001 ! route-map SET-MED permit 10 match ip address 1 set metric 100 ! route-map SET-MED permit 20 set metric 200 ! access-list 1 permit 192.168.1.0 0.0.0.255 ! </pre>	<pre> ! router bgp 65001 no synchronization network 192.168.3.0 neighbor 11.11.11.2 remote-as 65002 neighbor 11.11.11.2 route-map SET-MED out neighbor 13.13.13.1 remote-as 65001 ! route-map SET-MED permit 10 match ip address 1 set metric 100 ! route-map SET-MED permit 20 set metric 200 ! access-list 1 permit 192.168.2.0 0.0.0.255 access-list 1 permit 192.168.3.0 0.0.0.255 ! </pre>

- The intention of the route maps above is to designate RT2 as the preferred entry point to reach 192.168.1.0/24, and RT3 as the preferred entry point to reach 192.168.2.0/24 and 192.168.3.0/24. The networks will still be reachable through each router in case of a link or router failure.
- MEDs are being manipulated and set outbound when advertising to an EBGp neighbor. The 2nd route map **permit** statement that does not have any **match** clause but just a **set** clause is the **permit any** statement for the route map. All networks that are being processed through this section of the route map are permitted and set to a MED of 200. If the MED is not set to 200, it would have been set to a MED of 0 by default. Since 0 is less than 100, those routes would have been the preferred paths to reach the networks reside in AS 65001.
- Below shows the BGP routing table on EXT3 before manipulating the MED attribute.

```

EXT3#sh ip bgp
BGP table version is 4, local router ID is 23.23.23.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
* i192.168.1.0      11.11.11.1         0      100     0 65001 i
*>i                 10.10.10.1         0      100     0 65001 i
* i192.168.2.0      11.11.11.1         0      100     0 65001 i
*>i                 10.10.10.1         0      100     0 65001 i
*>i192.168.3.0      10.10.10.1         0      100     0 65001 i
* i                 11.11.11.1         0      100     0 65001 i
EXT3#

```

- Below shows how the BGP routing table on EXT3 has evolved to maintain only the best paths after implemented the path manipulation configuration on RT2 and RT3.

```

EXT3#sh ip bgp
BGP table version is 1, local router ID is 23.23.23.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
* i192.168.1.0      10.10.10.1         100    100    0 65001 i
* i                 11.11.11.1         200    100    0 65001 i
* i192.168.2.0      10.10.10.1         200    100    0 65001 i
* i                 11.11.11.1         100    100    0 65001 i
* i192.168.3.0      10.10.10.1         200    100    0 65001 i
* i                 11.11.11.1         100    100    0 65001 i
EXT3#
EXT3#sh ip bgp
BGP table version is 4, local router ID is 23.23.23.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*>i192.168.1.0      10.10.10.1         100    100    0 65001 i
* i                 11.11.11.1         200    100    0 65001 i
* i192.168.2.0      10.10.10.1         200    100    0 65001 i
*>i                 11.11.11.1         100    100    0 65001 i
* i192.168.3.0      10.10.10.1         200    100    0 65001 i
*>i                 11.11.11.1         100    100    0 65001 i
EXT3#
EXT3#sh ip bgp
BGP table version is 4, local router ID is 23.23.23.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*>i192.168.1.0      10.10.10.1         100    100    0 65001 i
*>i192.168.2.0      11.11.11.1         100    100    0 65001 i
*>i192.168.3.0      11.11.11.1         100    100    0 65001 i
EXT3#

```

## Manipulating the WEIGHT Attribute

- The **neighbor** *{ip-addr | peer-group-name}* **weight** *{weight}* BGP router subcommand assigns a weight upon all the routes learned through the specified neighbor. The value ranges from 0 to 65535, inclusive. Routes learned through a BGP peer have a default weight of 0; while the local routes that originated by the router itself have a default weight of 32768.
- The **set weight** *{weight}* route map action clause can also be used to manipulate the WEIGHT attribute for individual routes that matched by access lists, prefix lists, and various attributes. The weights assigned using this route map configuration command override the weights assigned using the **neighbor weight** command.



## AS Path Prepending

- AS path prepending is the manipulation of the BGP AS\_PATH attribute that inserts additional ASNs upon outgoing EBGP updates. Cisco IOS supports inbound and outbound AS path prepending on EBGP sessions. AS path prepending does not work on IBGP sessions.
- Outbound AS path prepending can be used as the last resort mechanism to influence the BGP routing policies in BGP multihoming scenarios; where all other methods, eg: manipulation upon the local preference and MED attributes through BGP communities, cannot be implemented due to the lack of support of the upstream ISPs.

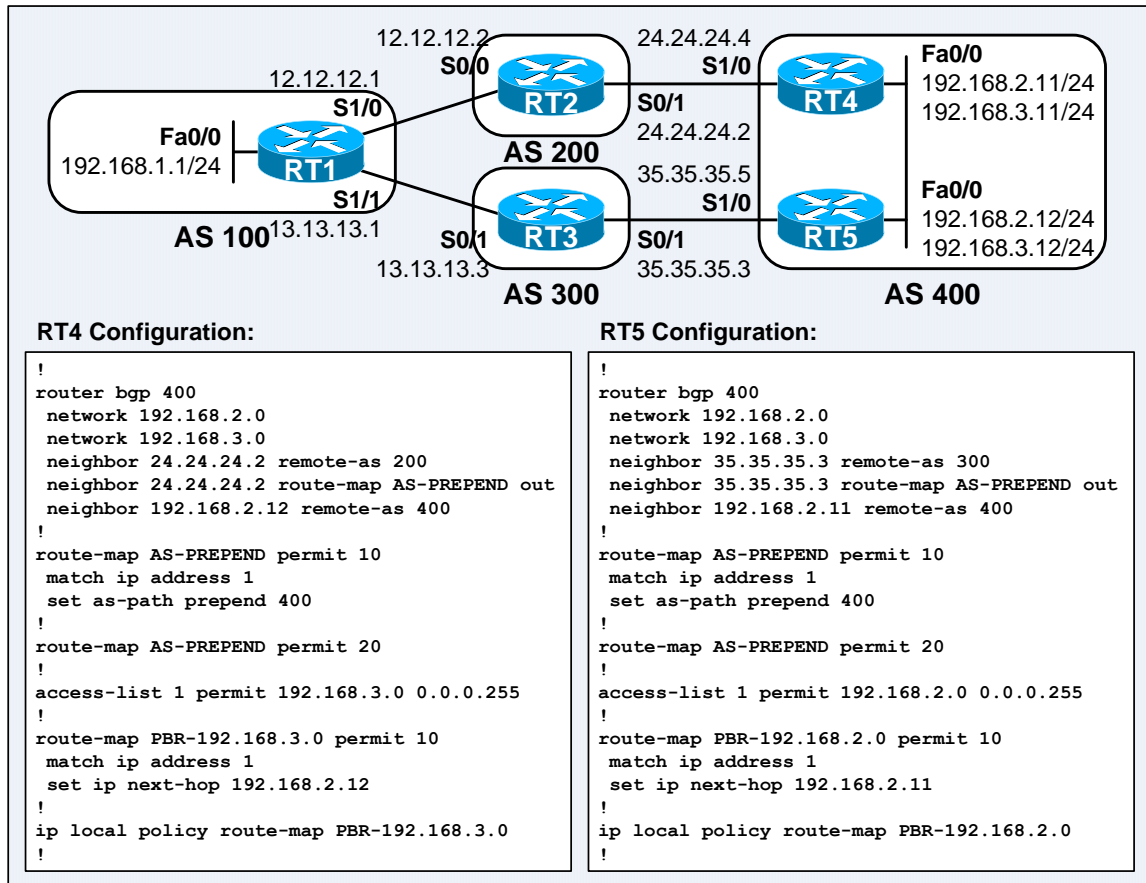


Figure 16-4: AS Path Prepending

- Below shows the BGP routing table on RT1. Since both EBGP routes learned from RT2 and RT3 have the same AS path length, whichever route that was received first (oldest path) will be selected as the best path and installed into the IP routing table.

```

RT1#sh ip bgp
BGP table version is 4, local router ID is 192.168.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network        Next Hop           Metric LocPrf Weight Path
*> 192.168.1.0    0.0.0.0             0         32768 i
*  192.168.2.0    13.13.13.3         0         0 300 400 i
*> 192.168.2.0    12.12.12.2         0         0 200 400 i
*  192.168.3.0    13.13.13.3         0         0 300 400 i
*> 192.168.3.0    12.12.12.2         0         0 200 400 i
RT1#

```

- Below shows the BGP routing table on RT1, a router in a AS behind the upstream ISP routers for AS 400 (RT2 and RT3), after AS 400 implemented AS path prepending to influence the traffic flow for inbound traffic towards its internal networks – 192.168.2.0/24 and 192.168.3.0/24. It is impractical to implement the propagation of the MED attribute across the whole network. Note that local policy routing is implemented on RT4 and RT5 mainly used to influence the traffic flow for the outbound traffic originated from AS 400 (and from the routers themselves).

```

RT1#sh ip bgp
BGP table version is 5, local router ID is 192.168.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 192.168.1.0      0.0.0.0           0         32768 i
* 192.168.2.0      13.13.13.3        0         0 300 400 400 i
*> 192.168.3.0      13.13.13.3        0         0 300 400 i
*                   12.12.12.2        0         0 200 400 400 i
RT1#

```

- By implementing AS path prepending and local policy routing in the sample network scenario, the traffic flow for inbound and outbound traffic towards 192.168.2.0/24 and 192.168.3.0/24 are being determined to flow through the RT2-RT4 and RT3-RT5 links respectively.
- Do not simply prepend any ASN when implementing AS path prepending; use only an ASN already in the AS path, eg: the ASN of the most recently added ASN or the local ASN.

## Multihoming

- Multihoming is a network setup in which a downstream AS (an enterprise) is connected to multiple upstream AS (ISPs) to achieve the following objectives:
  - Increase the reliability of the connection to the Internet; in which Internet connectivity can still be maintained if a single connection becomes unavailable.
  - Increase the performance of applications, in which certain traffic can be manipulated to load balance or load share over the Internet connections.
- BGP is inappropriate when an organization has only a single connection to the Internet; a default route is the recommended design instead. BGP provides the benefits only when an organization has multiple EBGP connections to either a single AS or multiple ASes.
- A drawback of having all EBGP connections to a single ISP is that Internet connectivity will be interrupted when there are connectivity issues in the single ISP. Below lists the advantages of having connections to multiple ISPs:
  - Redundancy with the multiple connections.
  - Does not tie to the routing policy of a single ISP.
  - Better path manipulation can be performed as there are more paths to the same networks.
- Below lists the 3 available design options for implement BGP multihoming.
  - All ISPs originate only default routes to the AS.
  - All ISPs originate default routes and selected specific routes to the AS.
  - All ISPs originate all routes to the AS.

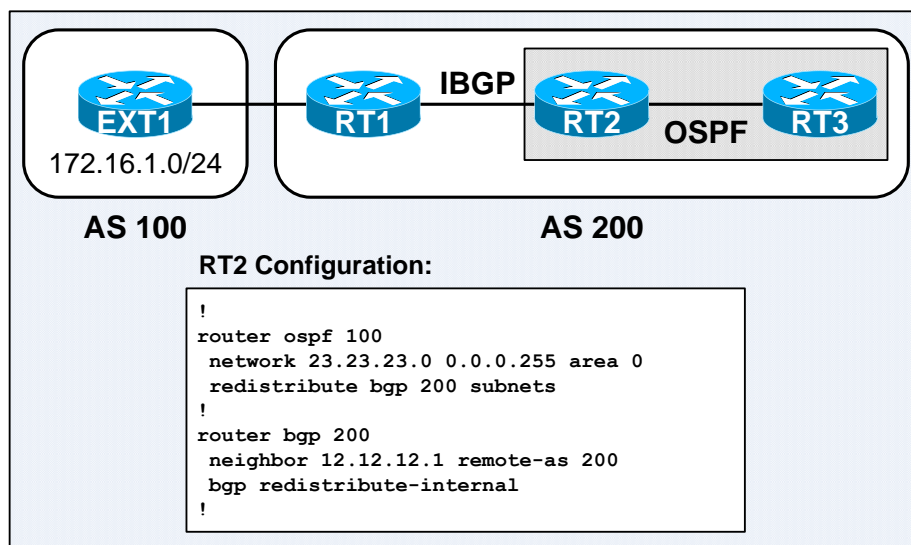
- The 1st design option requires the least router CPU and memory resources for routers within the AS as they only need to process default routes. The AS originates all its routes to the ISPs, which process them and propagate them to other ASes as appropriate.
- The upstream ISP that a router within the AS uses to reach the Internet is determined through the IGP metric towards the default route within the AS, in which the default route with the least-cost IGP metric is used. The entering point of the AS for inbound packets is decided outside the AS.
- The AS edge routers should implement discard routes that forwards traffic destined to the classful network address ranges that the AS uses, usually private addressing, eg: 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16, towards the Null interface in order to avoid wasting bandwidth to send packets unnecessarily for addresses that have not yet been deployed though the default route towards the Internet.
- Below are the limitations of this option:
  - Path manipulation cannot be performed as only a single route is received from each ISP.
  - Bandwidth manipulation is very difficult and can be accomplished only by manipulating the IGP metric of the default route.
  - Diverting outbound traffic towards different exit points is challenging as all destinations are being matched by the same default route.
- Regional ISPs that have multiple connections to national or international ISPs often implement this option. They do not use BGP for path manipulation but to ease the process of adding new customers and new networks of their customers. By running EBGP with the upstream ISPs, a regional ISP can simply adds the new networks of its customers into the BGP process and propagate across the Internet through the upstream ISPs in minutes. The operational issue without such setup is that an ISP must wait until the upstream ISPs add the new networks into their BGP process and configure static routes pointing to it every time it adds new networks.
- The 2nd design option is implemented when an enterprise AS requested its ISPs to originate some specific routes that it exchanges a lot of traffic in order for it to perform path manipulation to choose an outbound path over another path for certain destinations in the Internet. The customer can requests a partial routing table for the networks of the ISP and its customers; or the routes from any other ASes that the customer wants – customer-specified routes.
- This design option requires more resources within the AS than just originating the default route to the AS, as its routers must process the default and all external routes provided by the ISPs. However, this option is less CPU- and bandwidth-intensive than receiving the full BGP table.
- An AS that is receiving a partial Internet routing table has 2 options for the internal routers: redistribute BGP into the IGP; or run BGP on at least the core routers, if not all routers, in the AS. With the 1st option, the internal routers use the IGP metric of the redistributed routes to determine the exit path of the AS. The paths to all other external destinations that are not explicitly known through redistribution are decided by the IGP metric towards the default route within the AS.
- Running BGP on the internal routers allows better path manipulation for individual networks through BGP attributes than only relying upon the IGP metric. Manipulating individual routes can be difficult for IGPs, in which the best paths are selected based on the least-cost bandwidth for an interface that also applies to all networks via that path. Manipulating the interface bandwidth or cost to determine the best path for a specific network will also change the best paths for multiple networks, which eventually defeats the purpose of path manipulation.

- The 3rd design option requires high-end IBGP routers as the core routers to handle the full Internet routing table. Usually the edge routers of the AS reach the external networks based on the shortest AS path. The AS is unable to decide the paths that inbound packets take to reach it, but it can influence them using the MED attribute.
- Filtering BGP advertisements to upstream ISPs are important to prevent an enterprise AS from becoming a transit AS for its ISPs in multihoming scenarios. An AS edge router can tag the routes received from an ISP with the NO\_EXPORT community to inform other edge routers not to advertise the routes to other ISPs in order to prevent the enterprise AS from becoming a transit AS.
- Below lists the terms often used in the discussions of BGP multihoming:
  - Single homed (1 link per ISP; 1 ISP)
  - Dual homed (2+ links per ISP; 1 ISP)
  - Single multihomed (1 link per ISP; 2+ ISPs)
  - Dual multihomed (2+ links per ISP; 2+ ISPs)

## Redistribution between IGP and BGP

- A BGP router separately maintains the BGP table and the IP routing table. The router can be configured to share information between the BGP table and the IP routing table via redistribution. **Ships in the night (SIN) routing** is the approach in which the IGP and BGP are not communicating with each other via redistribution; both the IGP and BGP routing updates share the same routers and network bandwidth resources, performing different and unrelated functions.
- The IGP routing information can be advertised into BGP through one of the following ways:
  - Using the **network** BGP router subcommand.
  - Redistribute discard route (static route to the Null0) into BGP upon route summarization.
  - Redistribute dynamically learned IGP routes into BGP. This approach is not recommended as it can cause instability and even routing loops if implemented inappropriately. Note that any change upon the IGP routes (eg: route flapping) will originate a BGP Update and results in high rate of BGP exchanges throughout an internetwork.  
**Note:** Redistribute only local routes upon route redistribution; redistributing IGP routes that were actually learned through BGP redistribution earlier back into the BGP itself can result in routing loops. Implementing route filtering for such setups can be complex. The **redistribute** BGP router subcommand results in an incomplete Origin attribute for the redistributed route, as indicated by the **?** in the **show ip bgp** command output.
- **Note:** OSPF does not inject external OSPF routes into BGP by default as a measurement to prevent routing loop in case the external OSPF routes were actually originated from BGP. The **match external 1 external 2** keywords must be specified for the **redistribute** router subcommand under the BGP routing process in order to achieve the mentioned objective.
- The BGP routing information may be sent into an AS by redistributing the BGP routes into IGP. Redistributing BGP routes into IGP may overwhelm the IP routing tables on routers with limited resources. Implement route filtering to control the prefixes to be redistributed into the IGP. **Note:** RFC 1364 – BGP OSPF Interaction specified that the BGP Identifier must be the same as the OSPF Router ID when redistributing routes between BGP and OSPF.

- Redistribution from BGP into IGP normally is not required for ISP ASes, which typically have all routers, or at least the routers in the transit path within the AS running BGP, with full-meshed IBGP and exchanging EBGP routes across the AS. Synchronization would be disabled on all the BGP routers as synchronization between IGP and BGP is not required. The BGP routes are not need to be redistributed into the IGP. The IGP would need to route only information local to the AS and routes to the next-hop addresses of the EBGP routes.
- Redistributing BGP routing information into an AS may be necessary when an AS is running BGP only on its border routers and has other routers in the AS that do not run BGP but require knowledge of external routes.
- IBGP routes are not redistributed into IGP by default. The **bgp redistribute-internal** BGP router subcommand enables the redistribution of IBGP routes into IGPs, eg: OSPF and IS-IS. Issue the **clear ip bgp** command to reset BGP connections after this command is configured. Implement route filtering to control the prefixes to be redistributed into the IGP.  
**Note:** The **redistribute bgp {as-num} [subnets]** router subcommand is still required under the IGP process. The **subnets** keyword is applicable only for OSPF.



**Figure 16-5: IBGP-IGP Redistribution**

- Below shows the routing table on RT3:

```

RT3#sh ip route

Gateway of last resort is not set

 3.0.0.0/32 is subnetted, 1 subnets
C       3.3.3.3 is directly connected, Loopback0
 23.0.0.0/24 is subnetted, 1 subnets
C       23.23.23.0 is directly connected, FastEthernet0/0
 172.16.0.0/24 is subnetted, 1 subnets
O E2   172.16.1.0 [110/1] via 23.23.23.2, 00:00:20, FastEthernet0/0
RT3#

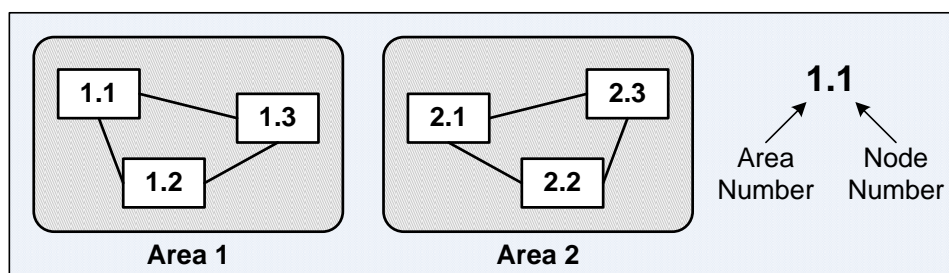
```

- **Note:** Redistributing IBGP routes into IGP can be dangerous – it can lead to routing loops as BGP cannot detect routing loops within an AS – it can only detect loops in EBGP routes; and the IGP metrics will be inconsistent from router to router.

## Integrated IS-IS

### DECnet and ISO CLNS

- **Intermediate System to Intermediate System (IS-IS)**; pronounced as “*i-sys*”) is a hierarchical link-state routing protocol for the **Open System Interconnection (OSI)** protocol suite. The OSI suite uses **ConnectionLess Network Service (CLNS)** to describe a connectionless data delivery service to the transport layer; whereas **ConnectionLess Network Protocol (CLNP)** defines the actual network layer protocol. CLNS and CLNP which are used for unreliable connectionless data delivery is similar and equivalent to **Internet Protocol (IP)** in TCP/IP suite. IS-IS uses CLNS addresses extensively for its operations, eg: identifying routers and building the link-state database (LSDB).  
**Note:** ISO makes a fine semantic distinction between the service offered to higher layers (CLNS) and the protocol used to implement it (CLNP). There is not such distinction in the TCP/IP world.
- IS-IS is the most popular and stable IP routing protocol in the Internet Service Provider industry. IS-IS is robust in some of the world’s largest internetworks due to its simplicity and stability.
- ISO has developed a suite of routing protocols – **End System to Intermediate System (ES-IS)**, **Intermediate System to Intermediate System (IS-IS)**, and **Inter-Domain Routing Protocol (IDRP)** for the OSI protocol suite. **Note:** Cisco IOS does not support IDRP.
- What is the difference between ISO and OSI? **International Organization for Standardization (ISO)** is the largest standards organization in the world that develops standards for networking; whereas **Open System Interconnection (OSI)** Reference Model represents an international standardization program that facilitates multivendor networking equipment interoperability. In the networking world, the ISO is best known for its OSI Reference Model.
- ES-IS and IS-IS were developed by **Digital Equipment Corporation (DEC)** for DECnet Phase V. DEC designed the DECnet protocol stack as part of its **Digital Network Architecture (DNA)**. DECnet supports both connection-oriented and connectionless OSI network layers. DECnet Phase V is equivalent to ISO CLNS. DECnet Phase V implements full OSI routing, which includes support for ES-IS and IS-IS. IS-IS was submitted by DEC to the ISO as the routing protocol for OSI.
- DECnet addresses are not associated with the physical networks to which the nodes are connected. DECnet locates hosts using **area.node** address pairs. The value of an area ranges from 1 to 63 (inclusive), and a node address between 1 and 1023 (inclusive). Therefore, each area can have 1023 nodes; and approximately 65000 nodes can be addressed in a single DECnet network. Areas can span across many routers, and a single cable can support many areas. Therefore, if a node has multiple network interfaces, it uses the same area.node address for all interfaces.



**Figure 17-1:** DECnet Locates Hosts using Area.Node Address Pairs

- Cisco IOS supports packet forwarding and routing for ISO CLNS on networks with different data link layer technologies, including Ethernet, Token Ring, FDDI, and serial. CLNS routing can be implemented on serial interfaces with HDLC, PPP, Link Access Procedure, Balanced (LAPB), X.25, SMDS, or Frame Relay encapsulation.
- The Cisco CLNS implementation is also compliant with the **Government OSI Profile (GOSIP) Version 2**. GOSIP is a specification that profiles networking products for procurement by the Federal Governments of the United States. This specification was first published as FIPS (Federal Information Processing Standard) 146-1 in 1990, and required OSI protocols to be used. FIPS 146-2 which was published on 1995 removed the procurement requirement for the OSI protocols, allows the procurement of products that implement ISO, ITU-T, or IETF standards. As the interest in OSI implementations declined, subsequent deployments of networking services for the civilian (non-military) government agencies are predominantly based on the TCP/IP suite.
- Routers are being referred to as **intermediate systems (ISs)**; while non-routing hosts are being referred to as **end systems (ESs)** in the OSI specification and terminology. IS-IS is a dynamic routing protocol for the OSI suite that allows the communication between routers to exchange and distribute routing information for routing CLNP packets in the ISO CLNS environment
- Cisco IOS fully support the following ISO and **American National Standards Institute (ANSI)** standards as part of its CLNS support:
  - i) **ISO 8473** – ISO ConnectionLess Network Protocol (CLNP)
  - ii) **ISO 8348/AD2** – Network Service Access Point (NSAP) addresses
  - iii) **ISO 9542** – End System to Intermediate System Routeing Information Exchange Protocol
  - iv) **ISO 10589** – IS-IS Intra-domain Routeing Information Exchange Protocol

**Note:** The spelling of **Routeing** is not a typographic error but the British spelling adopted by the ISO committee.
- The CLNP specification defines the format of packets. A data packet is generally referred to as **Protocol Data Unit (PDU)**.

- The Q3-Stack is a standardized management protocol which provides routing facilities (Layer 3) within a Telco Data Communication Network (DCN). The Q3 management traffic can be carried in a timeslot  $n$  of a 2Mbps signal. Below describes the general layer architecture of the Q3-Stack:

<b>Application</b>	Provides 2 application specific services – CMISE for management and FTAM for file transfer (similar to FTP). FTAM uses services provided by ACSE (Application Control Service Element) while CMISE uses both ACSE and ROSE (Remote Operations Service Element).
<b>Presentation</b>	Provides data representation and conversion, eg: ASN.1 encoding and BER decoding. Standards used in this layer are: <ul style="list-style-type: none"> <li>- <b>ISO 8822</b> – Presentation Service Definition</li> <li>- <b>ISO 8823</b> – OSI Presentation Protocol (PRES)</li> <li>- <b>ISO 8825</b> – Specification of Basic Encoding Rules (BER) for Abstract Syntax Notation (ASN)</li> </ul>
<b>Session</b>	Provides the management (opening and closing) of sessions. When a connection loss it will try to recover the connection; when a connection is idle for a long period, it may close it and reopen it for next use. These processes occur transparently to the higher layers. Standards used in this layer are: <ul style="list-style-type: none"> <li>- <b>ISO 8326</b> – Session Service Definition</li> <li>- <b>ISO 8327</b> – OSI Session Protocol (SES)</li> </ul>
<b>Transport</b>	Provides flow control, error detection and correction at the packet level, segmentation, and reassembly of packets. ISO has defined different classes of transport protocols. Standards used in this layer are: <ul style="list-style-type: none"> <li>- <b>ISO 8072</b> – Transport Service Definition</li> <li>- <b>ISO 8073</b> – Connection-Oriented Transport Protocol (COTP)</li> <li>- <b>ISO 8602</b> – ConnectionLess Transport Protocol (CLTP)</li> </ul>
<b>Network</b>	Provides routing and delivery of packets between nodes in a network. Standards used in this layer are: <ul style="list-style-type: none"> <li>- <b>ISO 8348</b> – Network Service Definition. Defines the connection-mode (CONS) and connectionless-mode (CLNS) services as the network layer for the interface specification between the network and transport layers.</li> <li>- <b>ISO 8473-1</b> defines the ConnectionLess Network Service (CLNS).</li> <li>- <b>ISO 8473-2</b> defines CLNS on ISO 8802 subnetworks.</li> <li>- <b>ISO 8473-3</b> defines CLNS on X.25 subnetworks.</li> <li>- <b>ISO 8473-4</b> defines CLNS on subnetworks provide OSI data link service.</li> <li>- <b>ISO 8473-5</b> defines CLNS on ISDN circuit switched B channels.</li> <li>- <b>ISO 9542</b> defines the ES-IS protocol for CLNS.</li> <li>- <b>ISO 10589</b> defines the IS-IS Intra-domain (Level 1) for CLNS.</li> <li>- <b>ISO 10747</b> defines the IS-IS Inter-domain (Level 2) for CLNS.</li> <li>- <b>ISO 8208</b> defines the X.25 Packet Layer Protocol for DTE.</li> <li>- <b>ISO 8878</b> defines how to use X.25 to provide the OSI Connection-Mode Network Service (CONS).</li> </ul>
<b>Data Link</b>	Provides error detection and correction at the bit level when transmitting data over a network media; and identifies the upper network layer protocols.
<b>Physical</b>	Provides the bit transfer over the physical medium. The X.25 protocol suite references X.21 and X.21bis as the standards for the physical layer; however, 2M TS $n$ or G.703 may also be used. The most frequently used standard for LAN is Ethernet, which may be used for both OSI and TCP/IP.



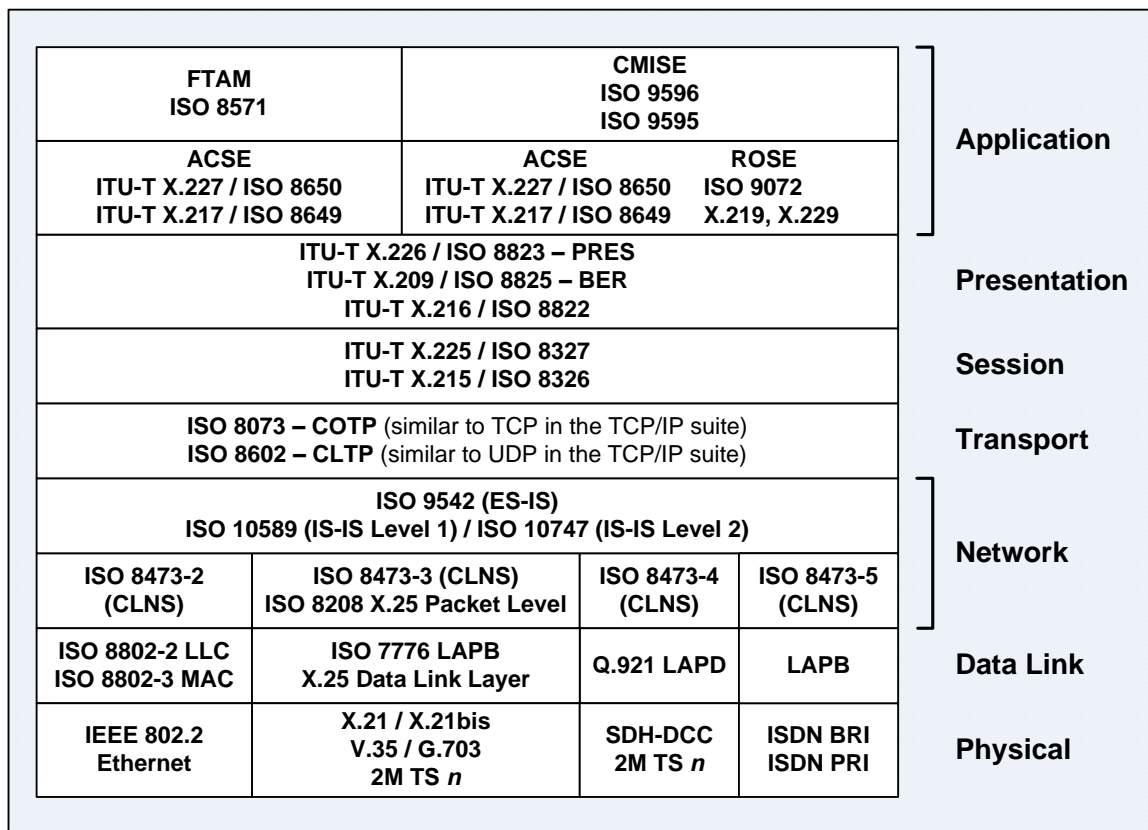


Figure 17-2: OSI Management Stack

- The OSI network layer can be divided into 2 main parts:
  - i) **Basic** – Forwarding packets from a node to another node based on the information in the local forwarding database. This part is specified by the CLNP protocol.
  - ii) **Advanced** – Automatically creating and updating the local forwarding database used by the first part. This part is specified by the ES-IS and IS-IS protocols.

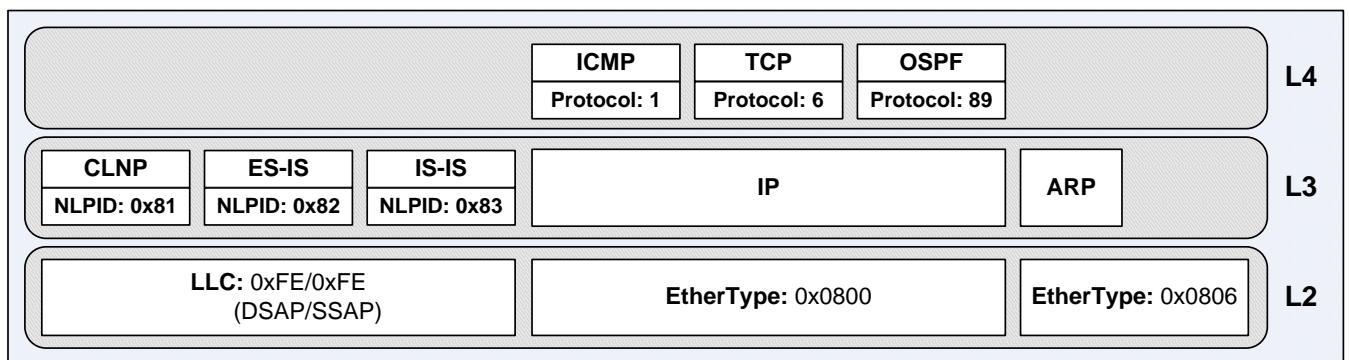
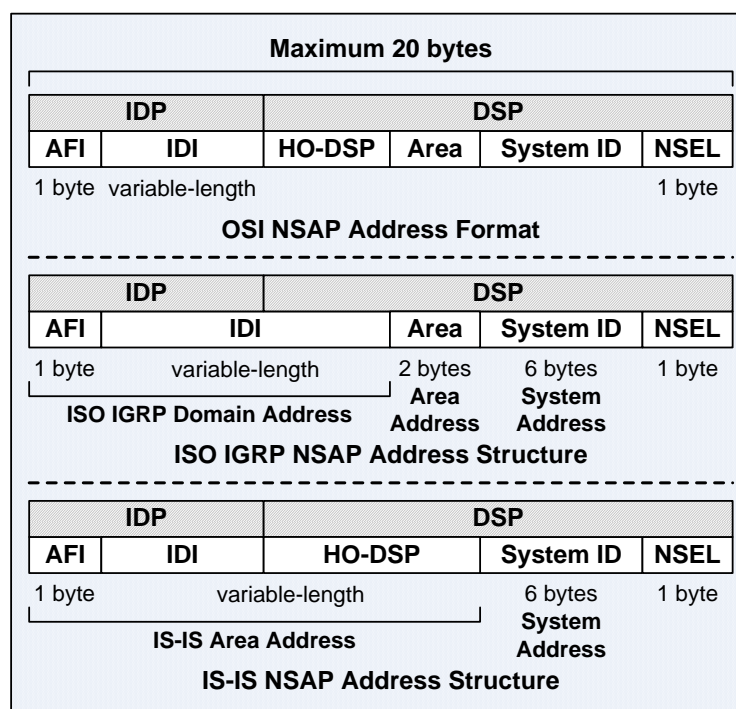


Figure 17-3: OSI and TCP/IP Network Layer Protocols

- Many sources claim that IS-IS runs on top of CLNP, but it is not the case! The identification of a L3 OSI protocol is based on the L2 info (DSAP = 0xFE) and the 1st byte of the L3 header – **Network Layer Protocol Identifier** (NLPID). IS-IS is therefore a separate network layer protocol and does not rely upon CLNP for datagram delivery; whereas IP routing protocols encapsulate their packets into IP, TCP, or UDP datagrams.

## Network Service Access Point (NSAP) Addressing

- Cisco IOS supports both the ISO-developed IS-IS routing protocol and the Cisco IOS Interior Gateway Routing Protocol (IGRP) for dynamic routing of ISO CLNS, as well as static routing for ISO CLNS. ISO CLNS is a standard for the network layer of the OSI reference model. Understanding the NSAP addresses is important for learning and implementing CLNS and IS-IS. Cisco IOS supports all NSAP address formats defined by ISO 8348/AD2; however, Cisco IOS supports the ISO IGRP and IS-IS dynamic routing protocols only for NSAP addresses that conform to the ISO 10589 standard for IS-IS.
- CLNS addresses in the ISO network layer are called **Network Service Access Points (NSAPs)** and **Network Entity Titles (NETs)**. Each node in an OSI network often has many NSAP and NET addresses. Each NSAP differs from one of the NETs for that node in only the last byte – the **NSAP-selector (NSEL)**, which identifies a process or upper-layer service on a device; similar to the transport layer port number and network layer Protocol field in the TCP/IP suite. When an NSAP is specified with an NSEL of 0, the NSAP is called the NET. The NET refers to the device itself, which is equivalent to the Layer 3 OSI address of the device. Routers use NETs similar to OSPF Router IDs to identify themselves in the IS-IS **Link-State PDUs (LSPs)** used to distribute link-state information – the basis for the OSI routing calculation; and the SPF calculations rely on NET addresses to identify routers for its operation.
- IS-IS provides a large and hierarchical addressing scheme. Its addressing scheme is designed for global addressing instead of local addressing. CLNS uses long variable-length NSAP addresses up to a maximum of 20 bytes, making it a viable successor to IPv4. At the time when the IETF community started to design the next-generation IP (before IPv6 appeared on the drawing board), the proposals to use CLNS were taken pretty seriously. However in the end, IETF decided to invent another protocol – IPv6, which effectively quadrupling the IPv4 address space while retaining most of the benefits and drawbacks of IPv4. The technical explanation for this decision was the variable-length CLNS addresses make the hardware implementation of layer 3 forwarding pretty complex; while some of the real reasons are probably the *not-invented-here* syndrome, and the lack of total control over a new protocol inherited from another organization.
- An NSAP address consists of the OSI network address and the info of the upper-layer protocols; which is equivalent to the combination of the IP address and the Protocol field that identifies upper-layer protocols in an IP header.
- NSAP addresses have a maximum size of 20 bytes. Various uses require definition of different address structure. The high-order bits identify the inter-area structure, while the low-order bits identify the unique systems within an area (intra-area).



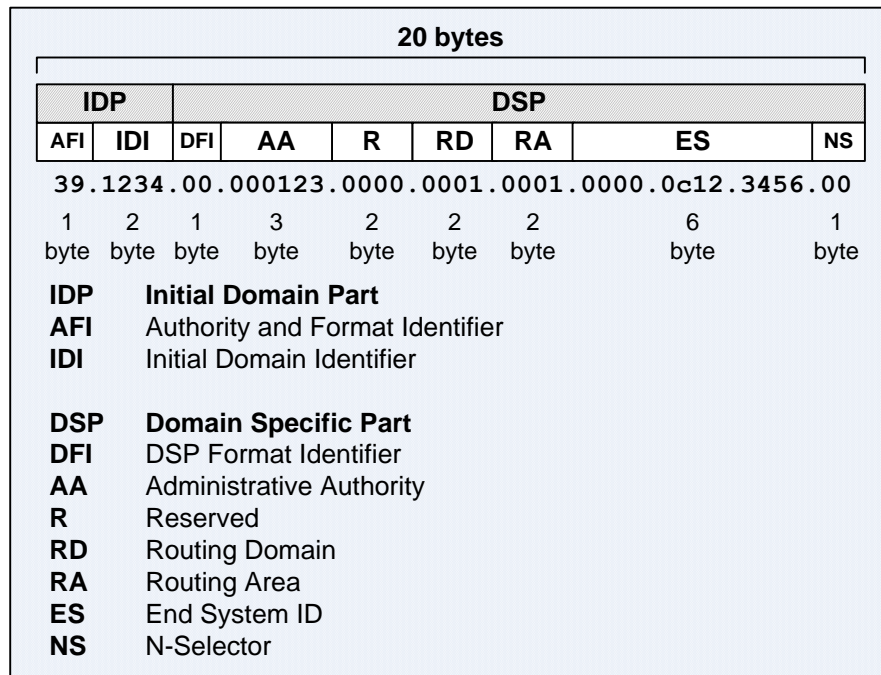
**Figure 17-4: OSI, ISO IGRP, and IS-IS NSAP Address Structures**

- The NSAP address for IS-IS and Integrated IS-IS consists of 3 fields – Area Address, System Address, and NSEL. The length of the area address can range from 1 to 13 bytes; therefore an NSAP address of an OSI device can be as little as 8 bytes in length. However, the NSAP is usually longer for finer granularity of the allocation of areas.
- The ISO 10589 NSAP addresses consist of the following:
  - i) The **Initial Domain Part (IDP)** of a NSAP address consists of the 1-byte **Authority and Format Identifier (AFI)** and the variable-length **Initial Domain Identifier (IDI)**. The IDP is similar to an IP classful major network. The AFI byte specifies the authority that assigned the address and the format of the address. The length of the IDI and the encoding format of the DSP of an NSAP address are based on the value of the AFI. Below lists some valid AFI values:

AFI	Address Domain
<b>39</b>	ISO Data Country Code (DCC)
<b>45</b>	E.164
<b>47</b>	ISO 6523 International Code Designation (ICD)
<b>49</b>	Locally administered (private). Similar to private addresses defined in RFC 1918. IS-IS routes these addresses. However, these addresses should not be advertised to other CLNS networks because they are ad-hoc addresses. Organizations which implemented private addressing schemes may face issues upon merger.

- The IDI identifies a sub-domain under the AFI, eg: 47.0005 is assigned to civilian departments of the US Government; while 47.0006 to the US Department of Defense.
- ii) The **Domain-Specific Part (DSP)** is used for routing within an IS-IS routing domain. The DSP of a NSAP address is comprised of the High-Order Domain-Specific Part (**HO-DSP**), system identifier (**System ID**), and NSAP selector (**NSEL**). The HO-DSP subdivides the domain into areas. The HO-DSP is the OSI equivalent of a subnet in IP. The **System ID** identifies an individual OSI device. In OSI, each device has an address; whereas in IP, each interface has an address that belongs to a different subnet. The **NSEL** identifies a process or service on the device and is not being used for routing.

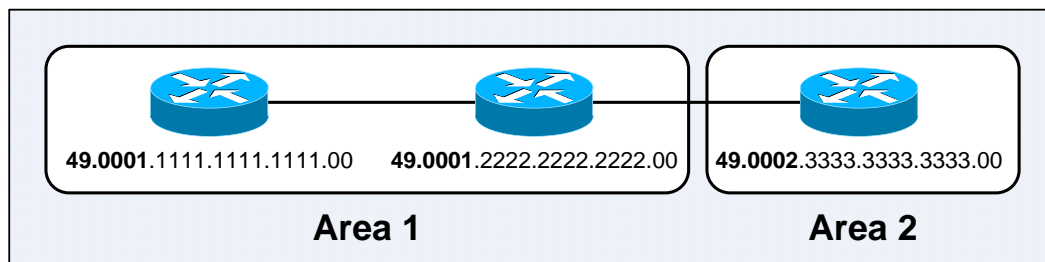
- The IDP is used for external routing between autonomous systems. It is assigned and standardized by ISO to identify organizations, and the organizations are responsible for assigning the format for the rest of the address by defining the DSP structure – the DSP is not standardized. The fact that ISO CLNS NSAP address can take so many forms often causes confusion. There is an address format recommended by the US GOSIP, ANSI, and UK GOSIP.



**Figure 17-5:** US GOSIP, ANSI, and UK GOSIP NSAP Address Structure

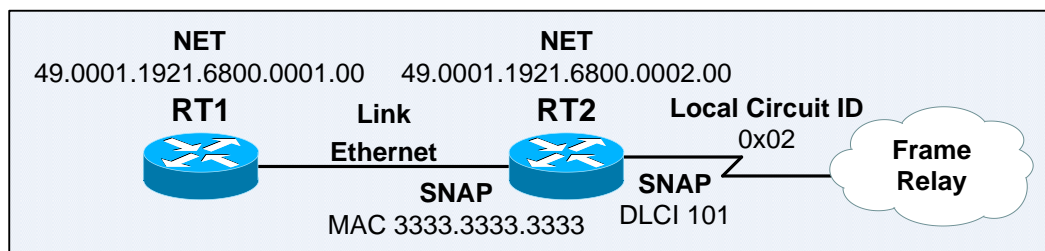
- The simplest NSAP format when running IS-IS as an IGP, comprises the following:
  - i) The **Area Address**, which must be at least 1 byte, separated into the 2 following parts:
    - The **AFI** set to 49, which designates locally administered and therefore individual addresses can be assigned by the organization.
    - The **Area ID**, the octets of the area address after the AFI, must be at least 1 byte.
  - ii) The **System ID**. All System IDs within an IS-IS routing domain must be the same length. Cisco enforces this OSI directive by fixing the length of the System ID at 6 bytes. Cisco IOS is compliant with the US Government OSI Profile (GOSIP) Version 2.0 standard which requires a 6-byte System ID.
  - iii) The **NSEL**, which must always be set to 0 for a router. The NSAP is called the **NET** when it has an NSEL of 0. Routers use NETs to identify themselves in the IS-IS PDUs.
- The Area Address is like an IP subnet; the System ID is like the host portion of an IP address. The ISO CLNS NSAP address is assigned to system instead of interface as with TCP/IP.
- A sample CLNS NSAP address is 49.0001.0000.0c12.3456.00, which represents the following:
  - AFI of 49.
  - Area ID of 0001.
  - System ID of 0000.0c12.3456, the MAC address of a LAN interface on the device.
  - NSEL of 0. The CLNS NSAP address is a NET which identifies a device.
 A Cisco router interprets the first byte of an NSAP address as the AFI, the last byte as the NSEL, the preceding 6 bytes are System ID, and anything else as the area address.
- **Note:** Sometimes the area address is also being referred to as the **prefix**. Many IS-IS documentation uses the terms **Area ID** and **Area Address** interchangeability.

- The 1st part of an NSAP is the **Area Address** and is associated with the IS-IS routing process. All devices within an area must have the same area address, which actually defines the area. The area address is used in L2 routing. Unlike OSPF which a router can reside in multiple areas, an IS-IS router can be a member of only one area.



**Figure 17-6:** IS-IS Routers are Members of Only One Area

- ESs recognize only other ESs and ISs on the same subnetwork that share the same area address.
- The 6-byte NSAP **System ID** must be unique within an area. The MAC address of a router is often being used as the System ID for the router; while for Integrated IS-IS, an IP address of a router is often being encoded as the System ID, eg: 192.168.0.1 → 1921.6800.0001.
- L1 intra-area routing is based on System IDs; therefore, each ES and IS must have a unique System ID within the area. All L2 and L1/L2 ISs recognize themselves in the IS-IS backbone; therefore, they must also have unique System IDs. As a conclusion, the System IDs should remain unique throughout an IS-IS routing domain. When the System IDs remain unique, there can never be a conflict at L1 or L2 if when a device moves between different areas.
- The 1-byte **NSEL** field in of an NSAP identifies a process or upper-layer service on a device; similar to the transport layer port number and network layer Protocol field in the TCP/IP suite. NET addresses are NSAP addresses with an NSEL value of 0. A NET address is actually the NSAP address is used to uniquely identify an OSI host within an IS-IS routing domain. Since IS-IS is originated from the OSI world and routing information is carried in IS-IS updates, NET addresses are still required even if the only routed protocol is IP.
- 3 additional IS-IS terms related to NET addresses – **Subnetwork Point of Attachment (SNAP)**, **Circuit ID**, and **Link** are introduced in the figure below.



**Figure 17-7:** SNAP, Circuit ID, and Link

- The **Subnetwork Point of Attachment (SNPA)** provides subnetwork (data link layer) services, which are the physical connection to a medium and the services offered to the physical and network layers. SNPA address is same as the Layer 2 address and is assigned using:
  - The **MAC address** on a LAN interface.
  - The **Virtual Circuit ID** from X.25 or ATM connection, or the **Data-Link Connection Identifier (DLCI)** from Frame Relay connection.
  - SNPA is simply set to \*HDLC\* for **High-Level Data Link Control (HDLC)** interfaces.

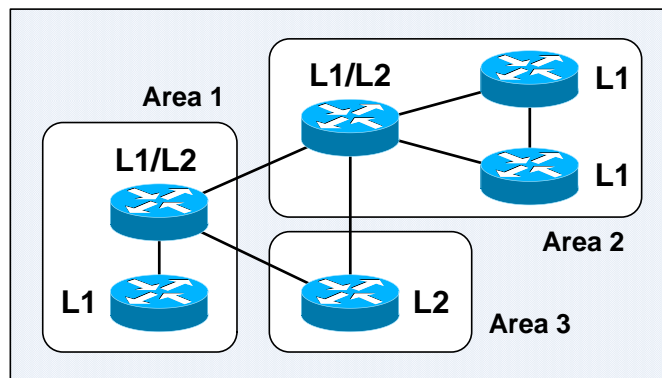
- A **Circuit** is the IS-IS term for an interface. Since the NSAP and NET refer to the entire device, therefore a Circuit ID is used to distinguish a particular interface. A router assigns a 1-byte Circuit ID to each of its interfaces as follows:
  - i) In the case of point-to-point interfaces, the SNPA is the sole identifier for the circuit. Ex: For an HDLC point-to-point link, the Circuit ID is 0x00.
  - ii) In the case of LAN interfaces, the Circuit ID is appended to the end of the System ID of the **Designated IS** (DIS) to form a 7-byte LAN ID, eg: 1921.6800.0001.01. On Cisco routers, the router hostname is used instead of the System ID; therefore the Circuit ID of a LAN interface may look like P1R1.01.  
**Note:** The Designated IS (DIS) will be discussed in more detail later.
  
- A **Link** is the path between 2 neighboring ISs. It is defined as being up when communication is possible between 2 neighboring SNPAs. It is transmitted to all other ISs within an area via LSPs.

## Integrated IS-IS

- IS-IS was developed for routing ISO CLNP networks and operates in strictly ISO CLNS terms; while **Integrated IS-IS** is an implementation of IS-IS that supports both ISO CLNS and IP in a single protocol. Integrated IS-IS tags CLNP routes with information about IP subnets. Integrated IS-IS provides an alternative to the dominant OSPF. It can be used for CLNS routing, IP routing, or a combination of both. CLNP is the first routed protocol for which IS-IS has provided routing services; the support for IP was added later.
  
- IS-IS is a public standard that was originally published as **ISO 10589** and republished as **RFC 1142 – OSI IS-IS Intra-domain Routing Protocol**; while Integrated IS-IS (or Dual IS-IS) is published as **RFC 1195 – Use of OSI IS-IS for Routing in TCP/IP and Dual Environments**.
  
- CLNS addresses apply to entire nodes and not to interfaces. Since IS-IS was originally designed for CLNS and IS-IS **Link-State PDUs** (LSPs) use NSAP addresses to identify the router, build the link-state topology table and the underlying IS-IS routing tree; therefore Integrated IS-IS still requires CLNS NSAP node addresses to function properly, even when it is being implemented for IP routing only on a router that is forwarding only IP packets. OSPF runs on top of IP; IS-IS runs directly on top of the data link layer – **protocol independent**.
  
- Integrated IS-IS supports VLSM and provides fast convergence ability as like OSPF and EIGRP. Each of them has its advantages and disadvantages, but this communality makes any of them scalable and appropriate for supporting today's large-scale networks.
  
- IS-IS operates similarly to OSPF. IS-IS allows a routing domain to be partitioned into areas. IS-IS routers establish adjacencies using a Hello mechanism and exchange link-state information using Link-State PDUs (LSPs) through an area to build the link-state topology database (LSDB). Every IS-IS router then applies Dijkstra's SPF algorithm upon its LSDB to build the SPF tree – **SPT** and select the best paths to be installed into the routing table. There is a minimal amount of info communicated between areas compared to OSPF, which reduces the burden of IS-IS routers. IS-IS could be described as **OSPF using only totally-stubby areas**. 🙌👍
  
- As with other LS routing protocols, IS-IS routers have the full picture of the network topology, and can independently make routing decisions based on the accurate picture of the network.



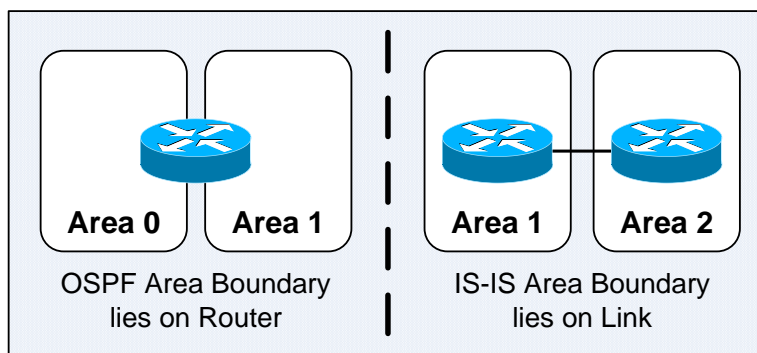
- IS-IS takes place at 2 levels within a routing domain – **Level 1 (L1)** and **Level 2 (L2)**.  
**L1 routing** occurs within an IS-IS area and is responsible for routing ESs and ISs inside an area (intra-area routing). All devices in an L1 routing area have the same area address. Routing within an area is accomplished using the locally significant address portion known as the System ID, and choosing the lowest-cost path. L1 builds a topology of System IDs in the local area and routes traffic **within the area** using the lowest-cost paths.  
**L2 routing** occurs between IS-IS areas (inter-area routing). L2 routers learn the locations of L1 routing areas to build the inter-area topology and routing table. L2 routers use the destination area address to route traffic using the lowest-cost paths. L2 exchanges prefix information (area addresses) between areas and routes traffic **between areas** using the lowest-cost paths.  
**Note:** Level 0 (L0) routing occurs between ESs and ISs on the same subnet. The OSI routing process begins at this level – end system and intermediate system. Level 3 (L3) routing occurs between separate OSI routing domains using IDRP; similar to BGP in IP inter-domain routing.



**Figure 17-8: 3 Types of IS-IS Routers**

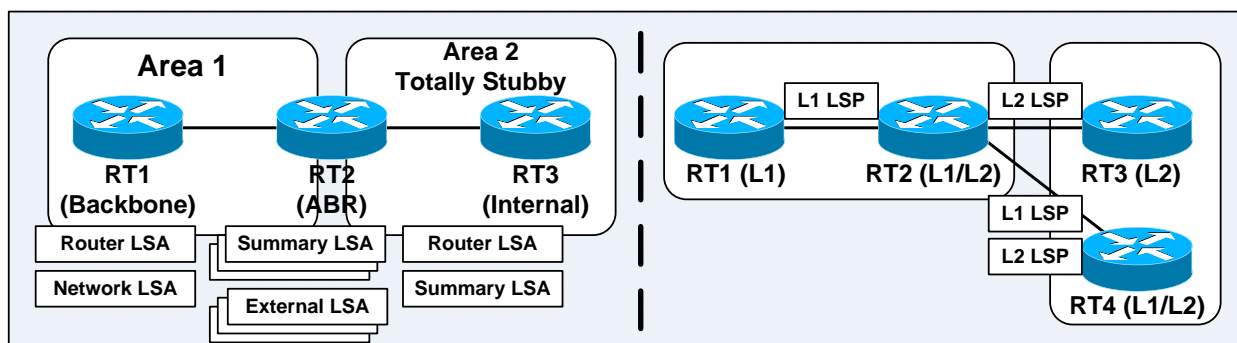
- IS-IS defines 3 types of routers to support the 2-level L1 and L2 hierarchical routing:
  - i) L1 routers use LSPs to learn about paths within the areas they reside in (intra-area). They form adjacencies and exchange routing info amongst themselves within an L1 area. It is similar to an OSPF totally stub router, which has only the topological info of an area, and uses a default route to the nearest L1/L2 router to route traffic to other areas.
  - ii) L2 routers use LSPs to learn about paths between areas they reside in (inter-area). They are considered backbone routers that only form adjacencies and exchange routing info amongst themselves within the backbone and route traffic between areas.
  - iii) Level 1 / Level 2 (L1/L2) combo routers learn about paths both within and between areas. They allow the L1- and L2-only routers to exchange routing information between areas. L1/L2 routers are equivalent to area border routers (ABRs) in OSPF. An L1/L2 router maintains a L1 LSDB for the area that it resides in and a L2 LSDB for inter-area routing.
- IP subnets are treated as leafs in the IS-IS SPT. Areas as recognized by the format of their NETs produce a summary into L2, and the L1/L2 router inject a default route back into L1 areas.
- Designing a totally flat IS-IS network with all L1/L2 routers provides an advantage of easy migration to multiple areas.
- An IS-IS router may reside in a L1 area, in the L2 backbone, or both. L1/L2 routers connect L1 areas to the L2 backbone. An L1/L2 router will maintain a L1 routing table to route to ES and IS within its own area using System IDs, and a L2 prefix table to route to other areas. When the only L1/L2 router in an area failed, the area would be unreachable throughout the routing domain!

- An L2 router is similar to an OSPF backbone router. The paths between the L2 and L1/L2 routers are called the backbone. All L2 and L1/L2 routers (the path of the backbone) must be contiguous – they cannot be separated by a L1 router somewhere in the middle.
- An IS-IS router normally has 1 NET address. The limit is 3 NETs for conventional IS-IS; and 3 NETs per area for multiarea Integrated IS-IS. Configuring multiple NETs on routers is a useful technique for merging 2 domains or transitioning from one addressing scheme to another. If multiple NETs are configured on the same router, they all must have the same System ID. The default can be changed using the **max-area-addresses** router subcommand.  
**Note:** The wording of *area* in this context is more appropriate being referred to as *process*.



**Figure 17-9:** Area Boundaries of OSPF and IS-IS

- IS-IS area boundaries lie on links instead of routers as with OSPF. Each IS-IS router belongs to exactly one area. Neighboring routers are able to learn that they are in the same or different areas and negotiate the appropriate adjacencies – L1, L2, or both.
- [Integrated] IS-IS has its own **Protocol Data Units (PDUs)** to transport information between ISs. Conventional IS-IS and Integrated IS-IS routing information is not carried within a network layer routed protocol (eg: CLNP) but is instead encapsulated directly within data link layer frames.
- With OSPF, network design is constrained because OSPF is based on a central backbone area 0, with the restriction that all other areas must be physically or logically connected to backbone. In comparison, IS-IS has a hierarchy of L1, L1/L2, and L2 routers. IS-IS permits a more flexible approach upon extending the backbone. The backbone can be extended by simply adding additional L1/L2 or L2 routers, a much less-complex process than with OSPF. 😊
- OSPF generates many types of small LSAs; whereas IS-IS groups IS-IS updates and sends them in a single LSP. As the network complexity increases, flooding of IS-IS updates is not an issue. Since each packet must be routed though, and routing takes network resources, so more packets represent a larger impact on the network. Since IS-IS uses significantly fewer LSPs, more routers, at least 1000, can reside in a single area, making IS-IS more scalable than OSPF.



**Figure 17-10:** IS-IS and OSPF Routing Updates

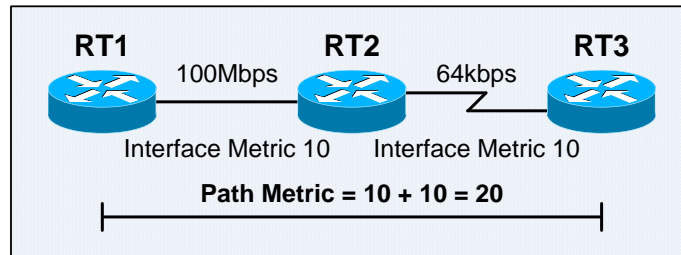


- IS-IS is also more efficient than OSPF in terms of CPU usages and processing routing updates. There are fewer LSPs (LSAs in OSPF terminology) to process. Besides that, the mechanism by which IS-IS installs and withdraws prefixes is less resource intensive as it uses NET addresses, which are already summarized.
- The convergence time depends on various factors, eg: timers, number of nodes, and router type. Based on the default times, IS-IS detects a failure faster than OSPF; therefore faster convergence. IS-IS has more timers than OSPF for fine-tuning and achieve finer granularity. Convergence time can be decreased significantly by fine-tuning those timers.
- L1/L2 routers should implement route summarization. Route summarization has many benefits. It saves CPU and memory resources since every router no longer responsible for the LSPs of the entire routing domain, and topology changes can be isolated to a small portion of the network instead of be propagating throughout the entire domain; therefore routers in other portions of the network can spend lesser time and resources for routing convergence upon topology changes.
- Below lists the rules for implementing route summarization on IS-IS:
  - All L2 routers can summarize routes at the area boundary.
  - When an L1/L2 router is summarizing routes sent to an L2 router, all L1/L2 routers must summarize in the same way.
  - All L1 routers cannot summarize routes.
- Older implementations of IS-IS use **narrow metrics**, which limits the maximum interface metric to 63 (6-bit) and the maximum total path metric to 1023 (10-bit); hence provides little space to distinguish between paths. Cisco IOS Software Release 12.0 and later support **wide metrics**, which allows a 24-bit interface metric and a 32-bit path metric. However, Cisco IOS uses narrow metrics by default. Mixing routers using narrow and wide metrics would increase complexities. The **metric-style narrow**, **metric-style wide**, and **metric-style transition** router subcommands instruct an IS-IS router to generate and accept old-style, new-style, and both styles of TLVs respectively.
- Narrow- and wide-style metrics are not compatible with each other. Migration from narrow to wide metric is a 2-stage process using the **transition** keyword. The **metric-style transition** IS-IS router subcommand should first be configured on all routers that are using narrow metrics. Once the whole network support both old- and new-style TLVs, the wide-style metric can then be implemented using the **metric-style wide** IS-IS router subcommand on all routers.
- The metric defines the cost to a destination. ISO 10589 defined the following 4 types of metrics:

<b>Default</b>	Also referred to as cost. Every Integrated IS-IS router must support this metric. The default cost applied to the outgoing interface of a Cisco router interface is 10.
<b>Delay</b>	An optional metric that reflects the transit delay.
<b>Expense</b>	An optional metric that reflects the monetary expense of the network.
<b>Error</b>	An optional metric that is based on the reliability of the path.

**Note:** Optional metrics are chosen before the default metric; however, Cisco supports only the **default** metric.

- Another issue of the Cisco IS-IS implementation is that it does not scale the interface metric. All IS-IS interfaces have a default metric value of 10; however, this can be changed manually. If the default metric is not adjusted on each interface, the IS-IS metric becomes similar to the hop count metric of Routing Information Protocol (RIP).



**Figure 17-11: Default IS-IS Path Metric Calculation**

- New ideas cannot be easily expressed as with OSPF, as they require the creation of a new LSA. The OSPF description schema is difficult to extend due to compatibility issues and it was developed exclusively for IPv4. IS-IS was designed with simplicity in mind, well-structured data formats, and can be easily extended through the Type, Length, and Value (TLV) mechanism. TLV triplets encode all IS-IS updates. This protocol-independence feature makes IS-IS easily extensible. IS-IS can easily grow to cover IPv6 (Integrated IS-ISv6) or any other new protocols, as extending IS-IS is simply creating new TLV triplets.
- An organization might choose OSPF over IS-IS because OSPF is more optimized and was designed exclusively as an IP routing protocol. Besides that, it is relatively easy to find both networking equipments and personnel to implement and support an OSPF network infrastructure. Furthermore, OSPF documentation is much more readily available than IS-IS documentation.
- Below summarizes the differences between OSPF and Integrated IS-IS:

OSPF	Integrated IS-IS
Area border inside routers (ABRs)	Area border on links
Each link in only 1 area	Each router in only 1 area
Complex to extend the backbone	Simple to extend the backbone
Many small LSAs sent	Fewer LSPs sent
Runs on top of IP	Runs on top of data-link layer
Requires IP addresses	Requires IP and CLNS addresses
Default metric is scaled by interface bandwidth	Default metric is 10 for all interfaces
Not easy to extend	Highly extensible with new TLV triplets
Equipments, personnel, and documentation more readily available	Equipments, personnel, and documentation not as readily available

- IS-IS and OSPF have more similarities than differences. Both routing protocols have the following characteristics:
  - They are open standard link-state routing protocols that utilize the Dijkstra's Shortest Path First (SPF) algorithm for their operations.
  - They support VLSM and 2-level hierarchical routing.
  - They have similar flooding mechanisms using link-state advertisements (LSAs), link-state aging timers, and link-state database synchronization to maintain the LSDB.
  - They are successful in the largest and most-demanding deployments – ISP networks.
  - They converge quickly upon network changes.

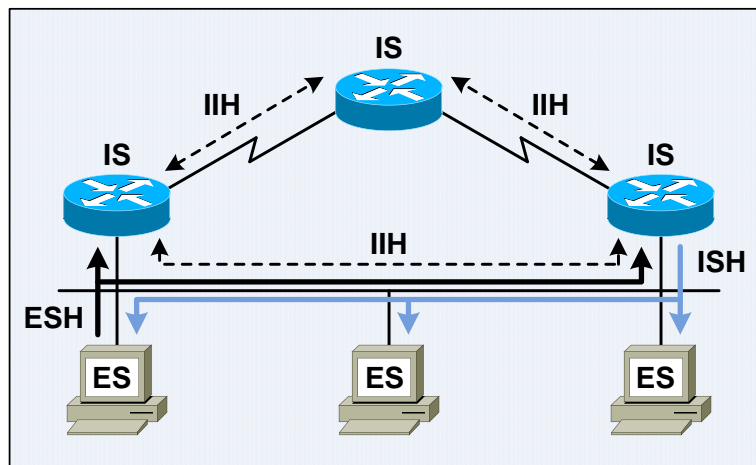
- The development of IS-IS began before OSPF development. Most of the development of the OSPF and IS-IS routing protocols was done concurrently. The cooperation and competition between the development groups produced 2 protocols that are very similar, yet one is better because of the other. The practical differences between the 2 protocols mainly deal with the issues of resources usage and customization.
- Most debates of the merits of IS-IS and OSPF are colored by their mutual history – different groups with different cultures developed them.
- IS-IS was originally developed by **Digital Equipment Corporation** (DEC) for DECnet Phase V. In 1987, the ANSI chose it to be the OSI IGP. At that time it could route only CLNP. IS-IS's evolution was ad-hoc; whereas OSPF was more formal. Below is a brief history of IS-IS:
  - 1985 – Originally called DECnet Phase V Routing.
  - 1988 – Adopted by ISO and renamed as IS-IS.
  - 1990 – Publication of RFC 1142 – OSI IS-IS Intra-domain Routing Protocol.
  - 1990 – Publication of RFC 1195 – Use of OSI IS-IS for Routing in TCP/IP and Dual Environments.
  - 1991 – Cisco IOS Software starts supporting IS-IS.
  - 1995 – Internet service providers (ISPs) start adopting IS-IS.
  - 2000 – Publication of IETF draft – IS-IS Extensions for Traffic Engineering.
  - 2001 – Publication of IETF draft – IS-IS Extensions in Support of Generalized MPLS.

**IETF – Internet Engineering Task Force**

- The ISO process is an international standards development process. ISO and many other groups did not approve TCP/IP due to its origin – the US Department of Defense (DoD) protocol. From the perspective of ISO, the development of IP was chaotic and imprecise, based on the famous maxim of “loose consensus and running code”. From the perspective of the early Internet engineers, the ISO process was slow, irritating, and disenfranchising.
- In 1988, the US National Science Foundation Network (NSFnet) was created. The IGP used was based on an early draft of IS-IS. The extensions to IS-IS for handling IP were developed in 1988. The development of OSPF just began during this time; OSPF was loosely based on IS-IS.
- OSPF Version 1 (OSPFv1) was published in 1989, conflicts ensued between the supporters of IS-IS and OSPF. Eventually the IETF supported both, but with the unofficial endorsement of the IETF and its continued favor of OSPF, OSPF which provide native IP support became more popular eventually.
- During the mid-1990s, large ISPs selected IS-IS as their IGPs for 2 reasons – IS-IS support IP and CLNS (solved 2 problems at once), and OSPF was still considered immature at the time.

## The End System to Intermediate System (ES-IS) Protocol

- Hosts are being referred to as **end systems** in the OSI specification and terminology. The **End System to Intermediate System (ES-IS)** protocol allows ESs (hosts) and ISs (routers) to discover each other, and allows ESs to learn their network layer addresses (similar to DHCP in TCP/IP). ES-IS handles topology information discovery and exchange between ESs and ISs. ES-IS is more like a discovery protocol than a routing protocol.
- ES-IS forms adjacencies between ESs and ISs. ES-IS performs the following tasks in a process known as **configuration**. Configuration must be completed prior to routing between ESs.
  - i) Identifies the area prefix to ESs (similar to DHCP in TCP/IP).
  - ii) Creates adjacencies between ESs and ISs.
  - iii) Creates data link to network address mappings (similar to ARP in TCP/IP).



**Figure 17-12:** The Operation of End System to Intermediate System (ES-IS)

- ESs (hosts) send **End System Hellos (ESHs)** to a well-known multicast addresses to announce their presence to ISs (routers). Routers listen to ESHs to discover the hosts on a segment. Routers include information on ESs in their LSPs to other routers (ISs). ESHs are generated by ESs and are sent to all ISs (L1, L2, and L1/L2) on the subnetwork. ISO end systems use ESHs to attach to intermediate systems. IP end systems do not send ESH, therefore Integrated IS-IS only attaches the directly connect subnets.
- ISs (routers) send **Intermediate System Hellos (ISHs)** to a well-known multicast addresses to announce their presence to ESs. ESs listen for ISHs and randomly select an IS on its directly attached network to forward their packets to other ESs. ISHs are generated by ISs and are sent to all ESs on the subnetwork.
- ISs use **IS-IS Hellos (IIHs)** to establish and maintain adjacencies (heartbeats) between them. IIHs are transmitted separately at Level 1 and Level 2.
- IP hosts do not use ES-IS. IP has its own processes and applications to handle the same functions as ES-IS, eg: Internet Control Message Protocol (ICMP), Address Resolution Protocol (ARP), and Dynamic Host Configuration Protocol (DHCP).
- Although Integrated IS-IS is able to support IP exclusively, yet it still uses CLNS to transmit reachability information and still forms adjacencies using ES-IS and IIHs.

## OSI Routing Levels

- There are 4 types of OSI routing operations. IS-IS is responsible for L1 and L2 OSI routing, while ES-IS is responsible for L0 OSI routing.

<b>Level 0 (L0) Routing</b>	OSI routing begins with ES-IS, with the ESs discover the nearest IS by listening to ISH packets. When an ES needs to send a packet to another ES (either on the same or another area), it sends the packet to an IS on an attached network. This process is known as L0 routing.
<b>Level 1 (L1) Routing (intra-area routing)</b>	Every ES and IS resides in a particular area. In order to forward traffic, the router looks up the destination address and forwards the packet via the best route. If the destination is on the same subnet, the IS knows the location of the ES by listening to the ESHs, therefore able to forwards the packet appropriately. The IS can also send a redirect message back to the source ES to tell it that a direct route is available. If the destination is on a different subnet but within the same area, the IS identifies the best path <b>using the System ID</b> , and forwards the traffic appropriately. This process is known as L1 routing.
<b>Level 2 (L2) Routing (inter-area routing)</b>	If a destination address is in another area, the L1 IS sends the packet to the nearest L1/L2 IS. Packet forwarding continues through L1/L2 and L2 ISs <b>using the area address</b> , until the packet reaches an L1/L2 IS in the destination area. This process is known as L2 routing. Within the destination area, L1 ISs forward the packet along the best path using the System ID, until the packet reaches the destination.
<b>Level 3 (L3) Routing (inter-domain routing)</b>	Routing between separate IS-IS domains is called L3 routing. L3 routing is similar to the <b>Border Gateway Protocol (BGP)</b> inter-domain routing in TCP/IP. L3 routing passes traffic between different autonomous systems which have different routing logic; therefore metrics cannot be compared directly. L3 OSI routing is not implemented in Cisco IOS but is specified and accomplished through the <b>Inter-Domain Routing Protocol (IDRP)</b> . <b>Note:</b> Cisco IOS does not support IDRP.

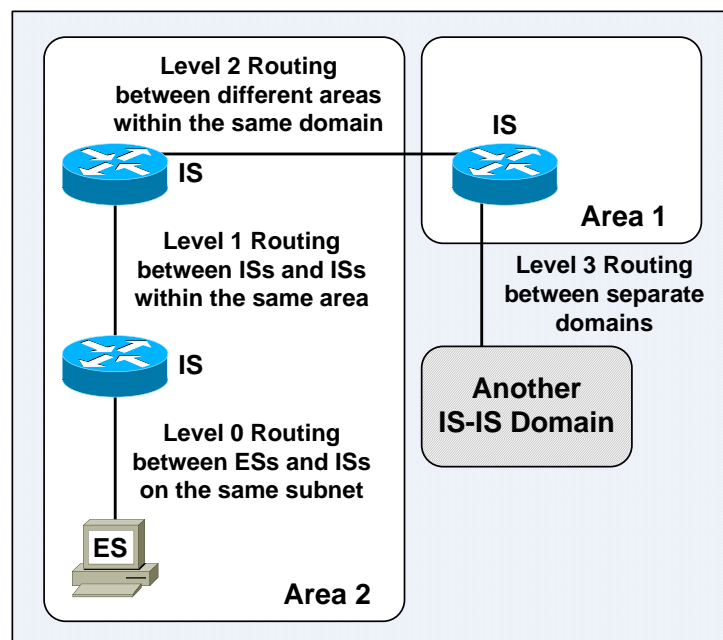


Figure 17-13: OSI Routing Levels

## IS-IS Routing Operation

- The network layer (OSI Layer 3) of the OSI protocol can operate as an ES (End System) or an IS (Intermediate System). The difference between is based on the way they handle NPDUs.
- L1 ISs maintain a copy of the L1 area LSDB; and L2 ISs maintain a copy of the L2 area LSDB. Every IS-IS router maintains a copy of the LSDBs for the levels it is responsible for.
- An L1/L2 router informs all L1 routers within its area that it is a potential exit point of the area. L1 routers use default route to forward traffic destined to other areas to the nearest L1/L2 router. L2 or L1/L2 routers in different areas exchange area address information and use Dijkstra's SPF algorithm to compute the best paths between areas, followed by forwarding traffic destined to other areas to the best L2 or L1/L2 router to reach the area.
- The System ID is used for routing within an area; the area address is not considered. The area address is used for routing between areas; the System ID is not considered.
- Since each IS-IS router makes its own best-path decisions upon every hop along the traffic path, there is a significant chance of **asymmetric routing**, in which traffic flow taking different paths in different directions. Therefore, it is important to know the traffic patterns within a network and fine-tune IS-IS for optimal path selection when necessary.

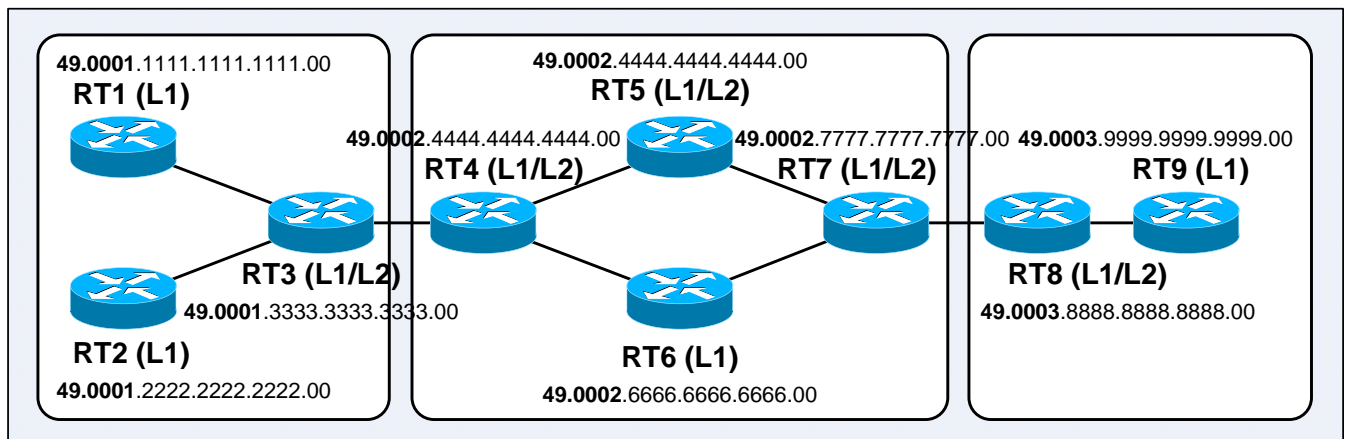


Figure 17-14: IS-IS Routing Example

- Below describes the traffic flow from RT1 to RT9 in the figure above:
  - RT1 notices the prefix of RT9 (49.0003) is not same as its prefix (49.0001). RT1 forwards the traffic to the nearest L1/L2 router – RT3. RT1 uses its L1 topology database to find the best path to RT3.
  - RT3 uses its L2 topology database to find the best next hop to reach the 49.0003 prefix – RT4. RT3 does not use the destination System ID for this decision.
  - RT4 uses its L2 topology database to find the best next hop to reach the 49.0003 prefix – RT7. RT4 does not use the destination System ID for this decision.
  - RT4 uses its L1 topology database to find the best path to reach RT7 – RT5, as they reside in the same area. RT4 uses the System ID of RT7 for this decision.
 

**Note:** RT5 must support L2 routing to ensure that the backbone is contiguous. When RT5 fails, RT6 cannot perform L2 routing even it provides a physical path across the area. RT6 should be configured as a L1/L2 router to provide redundancy.
  - RT7 uses its L2 topology database to find the best next hop to reach the 49.0003 prefix – RT8. RT7 does not use the destination System ID for this decision.
  - RT8 notices that the prefix of RT9 (49.0003) is same as its prefix (49.0003). RT8 forwards the traffic to RT9 using its L1 topology database to find the best path to RT9.

- The figure below shows a sample IS-IS network in which asymmetric routing occurs due to IS-IS L1 and L2 computations are separate – the L2 details are hidden from the L1 routers, which only recognize a default route to the nearest L1/L2 router.

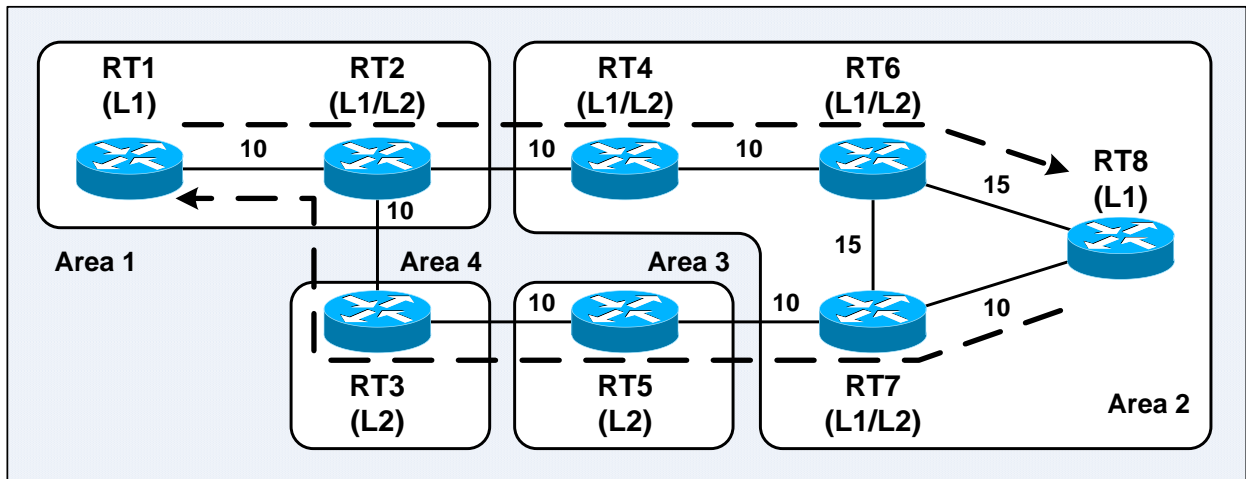


Figure 17-15: IS-IS Asymmetric Area Routing

- Below describes the traffic flow initiated from RT1 to RT8 and back to RT1:
  1. RT1 forwards the packets to its nearest L1/L2 router – RT2.
  2. RT2 forwards the packets along the shortest path to the destination area – area 2.
  3. The packets are then forwarded along the shortest intra-area path from RT4 – RT6 – RT8.
  4. RT8 forwards the return packets to RT1 via its nearest L1/L2 router – RT7.
  5. RT7 recognizes the best route to area 1 via area 3 based on the lowest-cost L2 path.

**Note:** Since L1 and L2 computations are separate, the path taken from RT8 back to RT1 is not necessary the lowest-cost path from RT8 to RT1.
- **Asymmetric routing** is not a serious networking problem but troubleshooting can be difficult and it may be a symptom of suboptimal routing. A good IS-IS network design is generally hierarchical and symmetric.
- **Route leaking** is a Cisco IOS feature that helps to avoid asymmetric routing and reduce suboptimal routing by leaking or redistributing L2 routes into L1 routers in a controlled manner. By having more details about inter-area routes, an L1 router is able to make better decisions upon forwarding traffic to the appropriate L1/L2 router.
- Route leaking is defined in RFC 2966 – Domain-wide Prefix Distribution with Two-Level IS-IS for use with the narrow metric TLV Type 128 and Type 130. The IETF also defined route leaking for use with the wide metric using TLV Type 135. An Up/Down bit in the TLV is used to indicate whether the route identified in the TLV has been leaked.
  - If the Up/Down bit is set to 0 – the route was originated within the L1 area.
  - If the Up/Down bit is set to 1 – the route has been redistributed into the area from L2.

The Up/Down bit is used to prevent routing loops – an L1/L2 router will not re-advertise any L1 routes that have the Up/Down bit set into L2.
- Route leaking should be planned and deployed carefully to avoid the situation where topology change in one area results in route recomputations in other areas.
- L1 internal routes have higher precedence and are chosen over L2 routes – forwarding packets outside an area instead of within an area is often a suboptimal route and can cause a routing loop.



## IS-IS Protocol Data Units (PDUs)

- The OSI stack defines a unit of data as a **PDU – Protocol Data Unit**. OSI addresses a frame as a data link PDU (DLPDU) and a packet or datagram as a network PDU (NPDU).

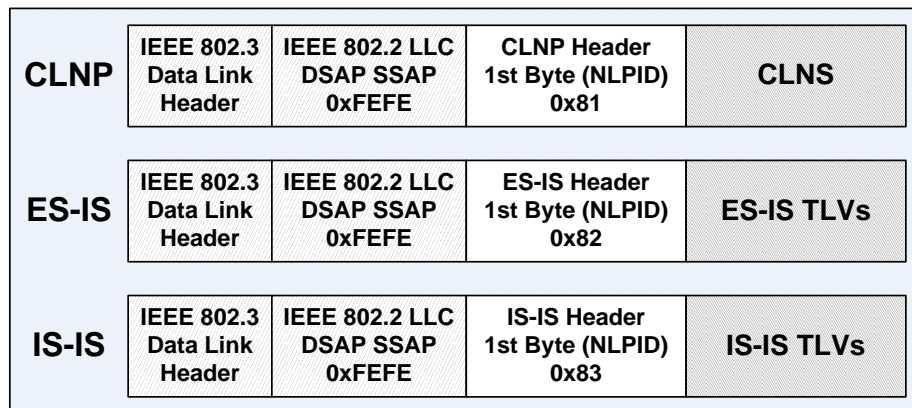


Figure 17-16: OSI CLNP, ES-IS, and IS-IS PDUs

- IS-IS and ES-IS PDUs are encapsulated directly in a data link PDU – frame, without CLNP header; CLNP PDUs contain a CLNP header between the data link headers and higher-layer CLNS info. The IS-IS and ES-IS PDUs contain variable-length fields depends upon the function of the PDU. Each field contains a TLV, which contains a type code, length, and appropriate value.
- IS-IS defines the following 4 types of PDUs:

<b>Hello PDU</b>	Establishes and maintains adjacencies. Includes ESH, ISH, and IIH.
<b>Link-State PDU (LSP)</b>	Distributes link-state information. The flooding or propagation of LSPs differs upon broadcast networks and point-to-point links.
<b>Complete Sequence Number PDU (CSNP)</b>	SNPs function similar to OSPF DBD, LSR, and LSack packets which are being used to synchronize LSDBs.
<b>Partial Sequence Number PDU (PSNP)</b>	CSNP briefly describes all the LSPs in the IS-IS LSDB of a router. PSNP may implicitly or explicitly acknowledges receives LSPs, and requests partial or missing pieces of link-state information;

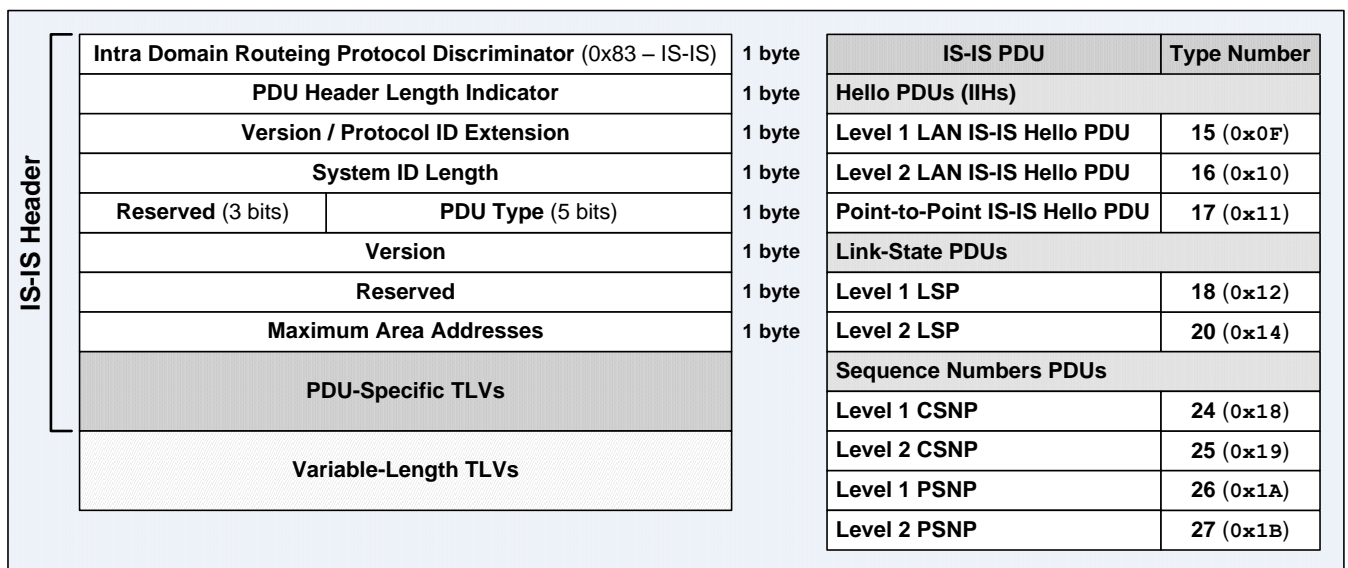
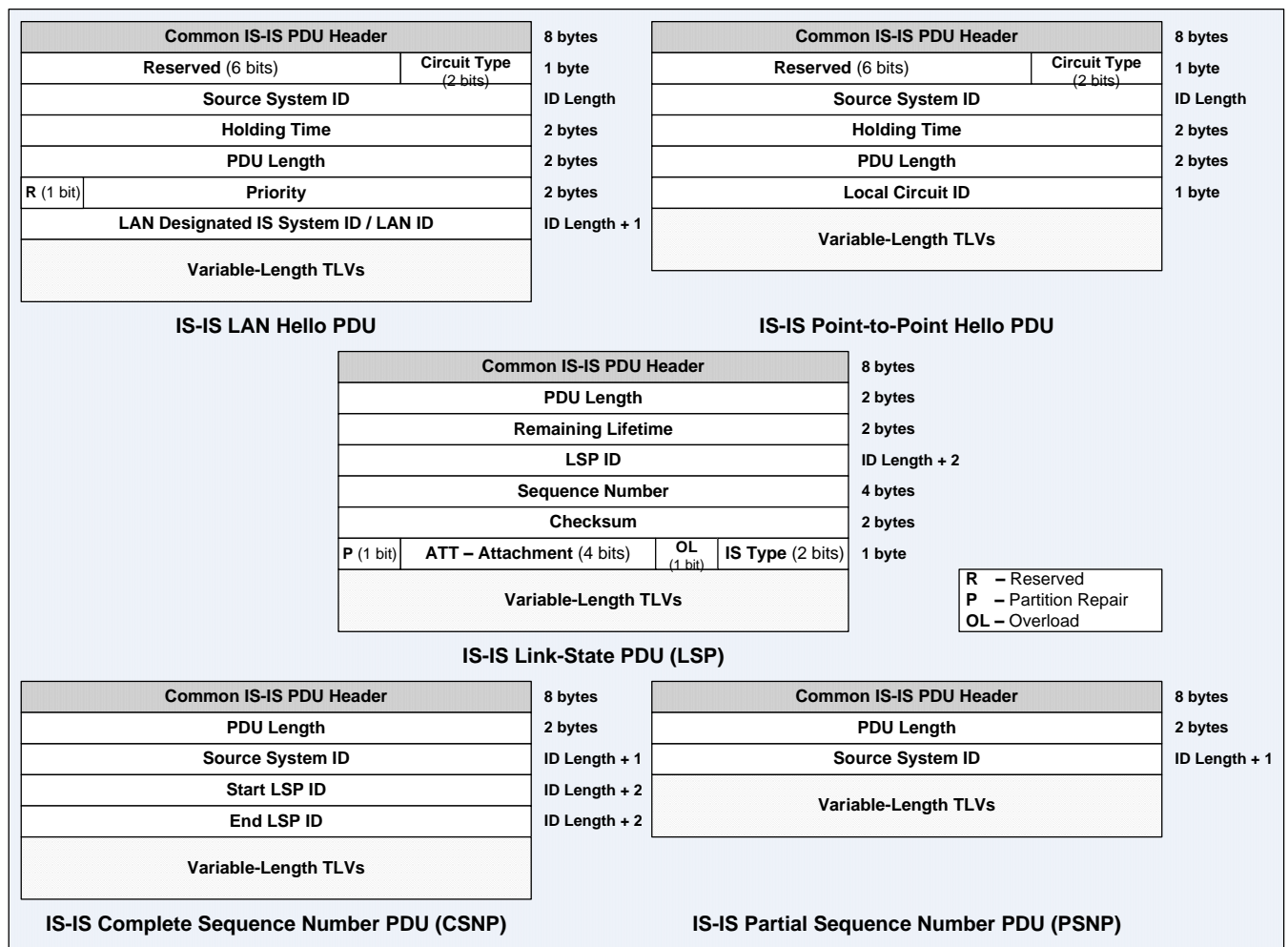


Figure 17-17: IS-IS PDU Format and IS-IS PDU Type Numbers



- The first 8 bytes of all IS-IS PDUs are header fields that are common to all types of IS-IS PDUs. **PDU Header Length Indicator** specifies the length of the fixed header in bytes. **Version / Protocol ID Extension** is always set to 0x01. **System ID Length** describes the length of the System ID field of NSAP addresses and NETs in a routing domain. This field is set to one of the following values:
  - An integer between 1 – 8 inclusive, indicating the length of the System ID field in bytes.
  - 0, indicating a System ID field of 6 bytes.
  - 255, indicating a null System ID field (0 bytes).
 Since the System ID of Cisco routers must be 6 bytes, therefore this field is always set to 0x00. **PDU Type** is a 5-bit field that contains one of the PDU type numbers. The preceding 3 bits are reserved and are always set to 0. **Version** is always set to 0x01, same as the Version / Protocol ID Extension in the 3rd octet. **Reserved** are always set to all zeros – 0x00. **Maximum Area Addresses** describes the number of area addresses permitted for this IS (router). This field is set to one of the following values:
  - An integer between 1 – 254 inclusive, indicating the number of areas allowed.
  - 0, indicating that the router supports a maximum of 3 area addresses.
 Cisco IOS supports maximum 3 area addresses by default; this field is always set to 0x00 unless the default has been changed using the **max-area-addresses** router subcommand.
- The PDU-specific fields following the common header fields are also part of the IS-IS header. They vary upon the different types of PDUs.



**Figure 17-18:** IS-IS PDU Format – LAN Hello PDU, Point-to-Point Hello PDU, LSP, CSNP, and PSNP

- Below describes the various IS-IS PDU-specific fields:

<b>Circuit Type</b>	A 2-bit field specifies whether the router is an L1 (01), L2 (10), or L1/L2 (11). If both bits are zero (00), the entire PDU is ignored. The preceding 6 bits are reserved and are always zero (000000).
<b>Source System ID</b>	The System ID of the originating router for the PDU.
<b>Holding Time</b>	The period a neighbor router should wait for the next IIIH before declaring the originating router is dead.
<b>PDU Length</b>	The length of the entire PDU in bytes or octets.
<b>Priority</b>	A 7-bit field used for DR election. It contains a value between 0 – 127; higher number has higher priority. L1 and L2 DRs are elected separately according to the priority values in L1 and L2 LAN IIIHs.
<b>LAN Designated IS System ID / LAN ID</b>	The System ID of the DIS + the Pseudonode ID (1 byte) to differentiate a LAN from other LAN connections that might have the same DIS.
<b>Local Circuit ID</b>	Assigned to a circuit by the router originating the P2P IIIH and is unique among the interfaces of the originating router. The Local Circuit ID in the IIIHs arrived at the other end of the P2P link might or might not contain the same value.
<b>Remaining Lifetime</b>	The LSP aging process ages out or removes outdated / expired, or invalid LSPs from the LSDB based on this value of the LSPs. The process uses a decreasing timer and is known as the count-to-zero operation. 1200 seconds (20 minutes) is the default start value.
<b>LSP ID</b>	The System ID + the Pseudonode ID + the Fragment ID of the LSP.
<b>Sequence Number</b>	Identifies duplicate LSPs and ensures that the latest LSP information is maintained in the link-state topology database for route computation. The sequence number of a router is set to 1 upon its reboot. The router then receives its previous LSPs back from its neighbors, which have the last sequence number before the router rebooted. The router then uses this number and reoriginates its LSPs with the next sequence number. This field contains a 32-bit (4-byte) unsigned integer.
<b>Checksum</b>	The checksum upon the contents of the LSP.
<b>Partition Repair (P)</b>	Although this bit exists in both L1 and L2 LSPs, it is relevant only in L2 LSPs. When this bit is set to 1, it indicates that the originating router supports the automatic repair of area partitions. Cisco IOS does not support this feature; it always originates LSPs with the P bit set to 0.
<b>Attachment (ATT)</b>	A 4-bit field indicating whether the originating router is attached to one or more areas. Although this bit exists in both L1 and L2 LSPs, it is relevant only in L1 LSPs originated by L1/L2 routers to indicate that it is also a L2 router, which is a potential exit to reach other areas. Reading from left to right (bits 7 – 4), the bits indicate the Error metric, the Expense metric, the Delay metric, and the Default metric. Cisco IOS supports only the default metric, so bits 5 – 7 are always 0.
<b>Overload (OL)</b>	The Link-State Database Overload bit. This bit is often set to 0. A router sets this bit on its LSPs when unable to store the entire LSDB. Routers receiving an LSP with the OL bit set will not use the originating router as a transit router as its routing table is incomplete, which may result in suboptimal routing and even routing loops; but they will still forward packets destined to the directly connected networks or interfaces of the originating router.

<b>IS Type</b>	A 2-bit field indicating whether the originating router is an L1 or L2 IS. 01 – L1; 11 – L2; 00 and 10 are unused values. An L1/L2 router sets the bits accordingly upon its L1 and L2 LSPs.
<b>Start LSP ID and End LSP ID</b>	A DIS periodically <sup>[1]</sup> multicasts a CSNP to describe all the LSPs in the link-state database of the pseudonode. Since there is an L1 database and an L2 database, therefore there are also L1 and L2 CSNPs. Some LSDBs can be really large that all the LSPs cannot be described in a single CSNP; the last 2 fields of the CSNP header, the Start LSP ID and the End LSP ID, describe the range of LSPs described in the CSNP for <b>fragmentation</b> purpose. The values of the Start LSP ID and End LSP ID are 0x0000.0000.0000.00-00 and 0xFFFF.FFFF.FFFF.FF-FF when all LSPs in the LSDB can be fit into a single CSNP. [1] – every 10 seconds

- Various router characteristics (eg: neighbor ISs, authentication, etc) are defined by an IS-IS LSP. An IS-IS LSP contains a common IS-IS PDU header and an IS-IS LSP header, followed by various TLV fields. IS-IS TLV triplets have similar functionalities as the TLV triplets in EIGRP. The TLV mechanism provides a flexible way of adding new data fields upon future extensions.

**Note:** Sometime TLV is also being referred to as **Code, Length, Value (CLV)**.

ISO uses the term Code, while IETF uses the term Type.

- The 1-byte Type (or Code) specifies the type of information or content of the Value field, the 1-byte Length specifies the length of the Value field, and the Value field contains the info itself. The Length field is important for error detection because the Value field is variable length. Due to the 1-byte size of the Length field, the maximum size of the Value field is 255 bytes.
- Below lists the basic and common IS-IS TLVs. The ISO-specified TLVs are designed for use with CLNP; however, most of them are also used with IP. The RFC-specified TLVs are designed only for IP. A router will ignore a TLV if it doesn't recognize and support the TLV Type / Code. This allows TLVs for CNLP, IP, or both to be carried using the same IS-IS LSP format.

Type	TLV Description	ISO 10589	RFC 1195	PDU Type								
				15	16	17	18	20	24	25	26	27
1 (0x01)	Area Address(es)	X		X	X	X	X	X				
2 (0x02)	IS Reachability	X					X	X				
3 (0x03)	ES Neighbors	X					X					
4 (0x04)	Partition Designed Level 2 IS	X						X				
5 (0x05)	Prefix Neighbors	X						X				
6 (0x06)	IS Neighbor(s)	X		X	X							
8 (0x08)	Padding	X		X	X	X						
9 (0x09)	LSP Entries	X							X	X	X	X
10 (0x0A)	Authentication	X		X	X	X	X	X	X	X	X	X
128 (0x80)	IP Internal Reachability		X				X	X				
129 (0x81)	Protocols Supported		X	X	X	X	X	X				
130 (0x82)	IP External Reachability		X				X	X				
131 (0x83)	Inter-Domain Routing Protocol		X					X				
132 (0x84)	IP Interface Address(es)		X	X	X	X	X	X				

**Figure 17-19:** Basic and Common IS-IS TLVs

- It is important to know the support of different TLVs on the network equipments because this determines the design and configuration of an Integrated IS-IS network.

## IS-IS Implementations on Broadcast Networks and Point-to-Point Links

- There are 2 general types of network topologies:

<b>Broadcast networks</b>	LAN links (eg: Ethernet, Token Ring, Fiber Distributed Data Interface – FDDI) and multipoint WAN links.
<b>Point-to-Point links</b>	Permanently established (eg: leased line, permanent virtual circuit – PVC) and dynamically established (eg: ISDN, switched virtual circuit – SVC) point-to-point links.

- IS-IS only supports the following 2 modes for its link-state information.

There are no commands to change the network type as with OSPF.

<b>Broadcast</b>	Default for LAN links and multipoint WAN links. Broadcast mode is recommended for use only on LAN interfaces.
<b>Point-to-Point</b>	Default for all other media types, including point-to-point subinterfaces and dialer interfaces.

**Note:** Avoid implementing IS-IS on dialup connections that incur usage-based costs, as IS-IS does not implement the Demand Circuit extension as in OSPF which may make such connections to stay up permanently and unwanted billing costs due to periodic IIs.

- IS-IS has no concept about NBMA. It is highly recommended to implement point-to-point links instead of multipoint links in NBMA environments (eg: X.25, ATM, or Frame Relay). IS-IS has no specific support for NBMA networks. When implementing a NBMA in broadcast mode using the **broadcast** keyword in the static DLCI mapping commands, Cisco IOS assumes that there is a NBMA network with full-mesh PVCs and able to advertise multicast updates. Static CLNS maps must also be created in addition to the static IP maps using the **frame-relay map clns {dlci-number} broadcast** interface subcommand.
- 2 IS-IS routers must support the same level of routing (L1 or L2) in order to form an adjacency. Separate IS-IS adjacencies are established for each level of routing. 2 neighboring routers on the same area perform both L1 and L2 routing would establish both L1 and L2 adjacencies. An IS-IS router maintains the L1 and L2 adjacencies in separate L1 and L2 adjacency tables. IS-IS routers on a LAN establish L1 and L2 adjacencies with all other routers on the LAN using specific L1 and L2 IIs; while OSPF routers on a LAN establish FULL adjacencies only with the Designated Router (DR) and Backup Designated Router (BDR).
- IIs announce the area address, and the L1 and L2 neighbors of the originating routers. An adjacency is formed when the area address and the IS type as communicated via the IIs are matched. L1 routers accept L1 IIH PDUs from their own area and establish adjacencies with other routers in their own area. L2 routers (and also the L2 process within an L1/L2 router) accept only L2 IIH PDUs and establish only L2 adjacencies. Unlike OSPF, the Hello intervals and holding time between 2 IS-IS neighboring routers do not need to be matched.
- IIs are padded to the full MTU size, which allows early error detection due to transmission problems with large frames or errors caused by mismatched MTUs on adjacent interfaces. Hello padding can be disabled in order to conserve network bandwidth in case the MTU of both interfaces is the same or translational bridging. When hello padding is disabled, Cisco routers still send the first 5 IIs padded to the full MTU size in order to discover MTU mismatches. The **no hello padding** IS-IS router subcommand and the **no isis hello padding** interface subcommand disable hello padding for all interfaces and a particular interface respectively.

- In IS-IS, a broadcast LAN is modeled as a directed graph or digraph with all the attached routers connected to a virtual router or pseudonode in a star topology manner. The pseudonode is actually the designated router or DIS (**Designated Intermediate System**) of the LAN. The virtual router or pseudonode makes the broadcast medium appeared as a virtual router and the routers appeared as its connected interfaces. It is responsible for generating LSPs on behalf of the LAN upon changes of its connections, eg: when a new neighbor comes online or offline. All routers maintain adjacencies to only the pseudonode instead of all routers on the LAN; thus reducing memory, CPU, and bandwidth resources. The adjacencies are managed by the DIS.
- An IS-IS router on a LAN establish adjacencies with all other routers (including the DIS) through the pseudonode. Each router (including the DIS) establishes a single adjacency to the pseudonode rather than having each router establishes an adjacency with every router on the LAN. Otherwise, there are  $n \times 2$  adjacencies established on broadcast network with  $n$  connected routers, and each router would required to establish  $n$  adjacencies to every router; nevertheless, generating LSPs for every adjacency during LSDB synchronization creates considerable overhead!
- A pseudonode is simply a virtual router; a real router must perform the tasks of the pseudonode. The DIS of the LAN takes on the responsibilities of the pseudonode which includes creating and maintaining adjacencies with all routers on the LAN, creating and updating the pseudonode LSP, and flooding LSPs over the LAN. The DIS sends out separate L1 and L2 LSPs for the pseudonode.
- The criteria for selecting the DIS are the router highest priority followed by the highest SNPA (the SNPA on LANs is the MAC address). Cisco router interfaces have a default L1 and L2 priority of 64. The priority value from 0 to 127 can be configured for L1 and L2 independently using the **isis priority** {priority-value} [**level-1** | **level-2**] interface subcommand. The L1 DIS and L2 DIS on a LAN may or may not be the same router, as an interface can have different L1 and L2 priorities. Setting the priority to 0 only lowers the chance of a router to become the DIS, but does not prevent it. When a router with a higher priority is being introduced to the LAN, it will preempt and take over the DIS role from the current DIS (different than OSPF). Since the IS-IS LSDB is synchronized frequently on a LAN (every 10 seconds), handing over the DIS role to another router is not a significant issue.
- When the current DIS fails, another router would take over and become the new DIS instantly with little or no impact upon the network. There is no backup designated router or DIS in IS-IS. Contrast this behavior with OSPF, where the DR and BDR are selected and the other routers on the LAN establish FULL adjacencies only with the DR and BDR. In case of DR failure, the BDR is promoted to become DR, and a new BDR is elected.

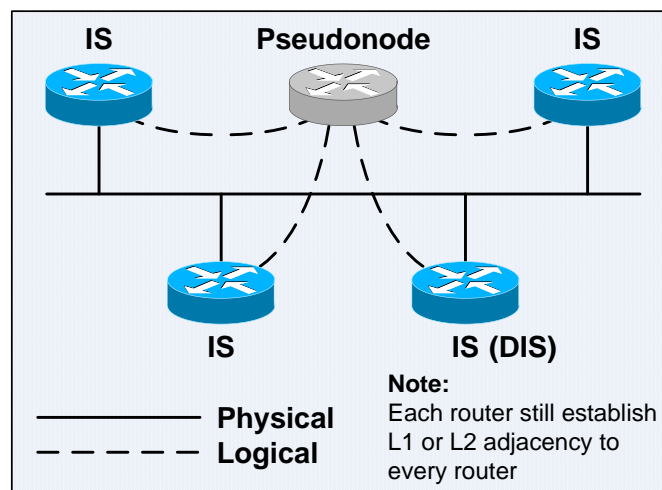
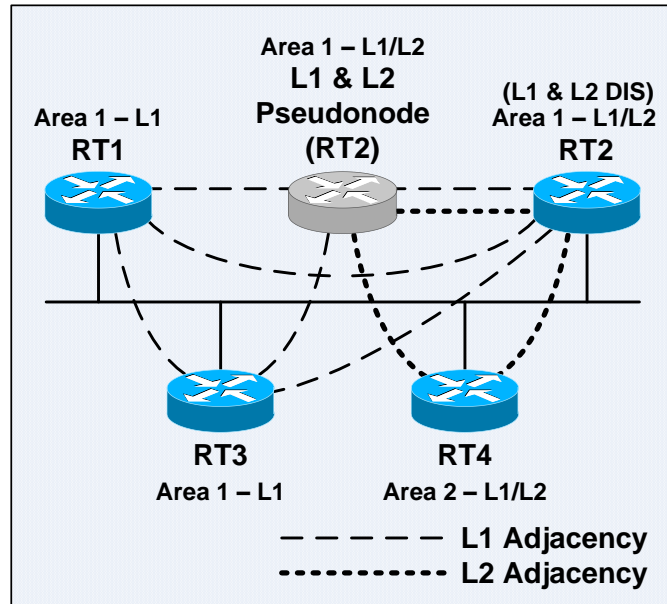


Figure 17-20: IS-IS Adjacencies over a Broadcast Network

- Figure above shows the IS-IS adjacencies over a broadcast network and how the DIS generates the Pseudonode LSPs. A pseudonode LSP details only the adjacent router connected to the LAN. The Pseudonode is logically connected to all routers; all routers still establish adjacencies among themselves. The Pseudonode LSP is used to build the network map and eventually the SPT. The Pseudonode LSP is the equivalent of a Type-2 Network-LSA in OSPF.  
**Note:** The DIS doesn't actually establish adjacencies and synchronize LSDB with all routers; it is the Pseudonode, a virtual router that created by the DIS.
- IS-IS uses a 2-level area hierarchical. The link-state information for the 2 levels is distributed separately using L1 and L2 LSPs. Each IS originates its own LSPs (one for L1 and one for L2). L1 and L2 IS-IS LAN PDUs are sent periodically as multicasts using multicast MAC addresses. L1 and L2 IIHs, LSPs, and SNPs on a LAN are sent to the AllL1IS multicast MAC address 0180.C200.0014 and the AllL2IS multicast MAC address 0180.C200.0015 respectively. IS-IS PDUs are sent out as multicasts on broadcast networks; and as unicasts on point-to-point links.
- IIHs are used to establish and maintain adjacencies between routers. If a router does not receive an IIH from a neighboring router within the **holding time**, the neighboring router is declared dead and all routing entries associated with the router are removed from the routing table. Note that the database entries associated with the router still remain in the link-state database. The holding time is calculated as the product of the Hello multiplier and Hello interval. The default Hello interval is 10 seconds and the default Hello multiplier is 3; therefore the default holding time is 30 seconds. The Hello interval can be adjusted using the **isis hello-interval** {sec} [level-1 | level2] interface subcommand. Unlike OSPF, the Hello intervals and holding time between 2 IS-IS neighboring routers do not need to be matched.
- The IS-IS adjacencies on a LAN is maintained by the DIS. The DIS sends out Hellos 3 times faster than the Hello interval of other routers – 3.3 seconds, in order to detect DIS failure quickly.
- When a network consists of only 2 IS-IS routers over a broadcast network, the connection can be treated as a point-to-point link instead if a broadcast network.
- Unlike LAN interfaces which generate separate L1 and L2 IIHs, point-to-point links have a common point-to-point IIH format that specifies whether the Hello is for L1 or L2 or both. Point-to-point IIHs are sent to the unicast address of the neighboring router at the other end.
- Below summarizes the differences between IS-IS broadcast and point-to-point modes:

	<b>Broadcast Mode</b>	<b>Point-to-Point Mode</b>
<b>Usage</b>	LANs and full-mesh WANs.	PPP, HDLC, and partial-mesh WANs.
<b>Hello interval</b>	3.3 seconds for DIS; 10 seconds for others.	10 seconds
<b>Adjacencies</b>	$n \times 2$	$n - 1$
<b>Uses DIS?</b>	Yes	No
<b>IIH Type</b>	L1 IIH and L2 IIH	Point-to-Point IIH

## IS-IS LAN and WAN Adjacencies



**Figure 17-21:** IS-IS L1 and L2 Adjacencies over a Broadcast Network

- Figure above shows a LAN attached with routers from 2 areas.
  - L1 routers from one area accept L1 IIHs only from their own area and therefore establish adjacencies only with other L1 routers reside in their own area (RT1, RT2, RT3).
  - L1 routers reside in another area establish adjacencies using L1 IIHs of their own area.
  - L2 routers (or the L2 process within an L1/L2 router) reside in the same or different areas establish only L2 adjacencies using L2 IIHs.

Fewer adjacencies are formed in OSPF on a LAN – each router form adjacencies only with the DR and BDR. In IS-IS, each router forms an adjacency with every router on a LAN.

- L1 and L2 IIHs share a common format on point-to-point WAN links. The area address and router type (L1 or L2) are announced in the IIHs.
  - 2 L1 routers in the same area (which includes the links between L1 and L1/L2 routers) exchange L1 IIHs and establish an L1 adjacency.
  - 2 L1 routers from different areas do not establish an adjacency.
  - 2 L2 routers (in the same area or between areas, including the links between L2 and L1/L2 routers) exchange L2 IIHs and establish an L2 adjacency.
  - 2 L1/L2 routers in the same area establish both L1 and L2 adjacencies using L1 and L2 IIHs.
  - 2 L1/L2 routers from different areas establish only an L2 adjacency using L2 IIHs.

	Area 1 L1	Area 1 L2	Area 1 L1/L2	Area 2 L1	Area 2 L2	Area 2 L1/L2
Area 1 – L1	L1	None	L1	None	None	None
Area 1 – L2	None	L2	L2	None	L2	L2
Area 1 – L1/L2	L1	L2	L1/L2	None	L2	L2
Area 2 – L1	None	None	None	L1	None	L1
Area 2 – L2	None	L2	L2	None	L2	L2
Area 2 – L1/L2	None	L2	L2	L1	L2	L1/L2

**Figure 17-22:** IS-IS Adjacency Matrix over Point-to-Point WAN Links

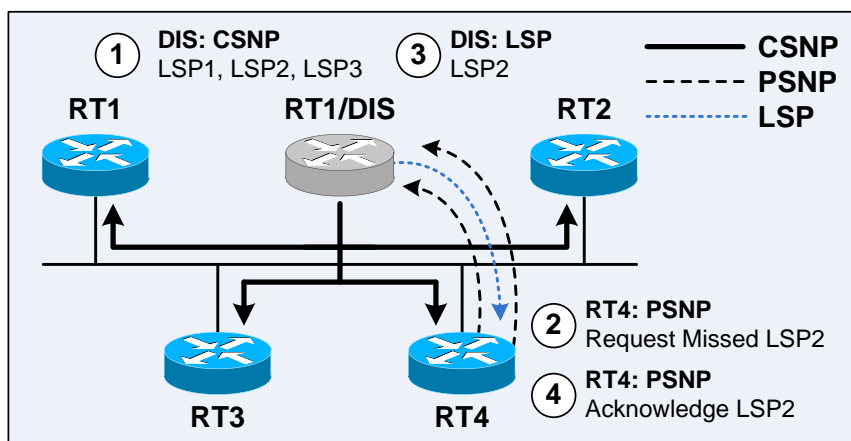
## IS-IS Link-State Database Synchronization

- An IS-IS update process is responsible for flooding the LSPs throughout an IS-IS domain. An LSP is flooded to all adjacent neighbors except the neighbor from which it was received. An L1 LSP is flooded to all routers within an area; it lists the adjacencies to other L1 routers. An L2 LSP is flooded throughout the backbone to all L2 routers in the routing domain; it lists the adjacencies to other L2 routers and the areas that attached to the originating router. An IS-IS router maintains the L1 and L2 LSPs in separate L1 and L2 LSDBs.
- IS-IS ignores and flooded unrecognized LSPs; OSPF ignores and discards unrecognized LSAs. An LSP is flooded to neighbors upon an adjacency up / down event, an interface changes state or is assigned with a new metric, or a change upon the routing table due to route redistribution.
- Each IS originates its own L1 and/or L2 LSPs. These LSPs are identified by the System ID of the originator and an LSP Number (or Fragment ID) starting at 0. When an LSP exceeds the MTU, it is fragmented into several LSPs with Fragment IDs numbered with 1, 2, 3, and so on. L1 and L2 LSPs can share the same format, as they express routing info using different TLVs.  
**Note:** IS-IS which runs over the data link layer unable to utilize the fragmentation service provided by the network layer; fragmentation is performed by itself to make sure that the size of any LSP does not exceed the MTU of any segment.
- When an IS receive an invalid LSP which failed the checksum, the IS would discard it and flood it as an expired LSP with a lifetime of 0. If a valid LSP is newer than the entry in the LSDB, it is retained, acknowledged, and refreshed with a lifetime of 1200 seconds. The LSP lifetime is decremented every second until it reaches 0, the point that it is considered expired. As soon as an LSP expires, it is kept for an additional 60 seconds before it is being flooded as an expired LSP. The IS-IS refresh interval is 15 minutes (900 seconds) minus a random jitter of up to 25%.  
The **isp-refresh-interval** {sec} IS-IS router subcommand sets the LSP refresh interval.
- **Sequence Number PDUs** (SNPs) are used to acknowledge the receipt of LSPs and maintain LSDB synchronization. There are 2 types of SNPs – **Complete Sequence Number PDU** (CSNP) and **Partial Sequence Number PDU** (PSNP). The usage of SNPs differs between broadcast networks and point-to-point links.
- CSNPs and PSNPs share the same format and carry summarized LSP information. The main difference between them is that CSNPs contain summaries of all LSPs in the LSDB; whereas PSNPs contain only a subset of LSP entries. Specific L1 and L2 CSNPs and PSNPs are being used for L1 and L2 routing. SNPs are never flooded but only sent between neighbors.
- Adjacent IS-IS routers exchange CSNPs to compare their LSDB. On broadcast networks, only the DIS transmits CSNPs. All adjacent neighbors compare the summary of LSPs received in the CSNP with the contents of their local LSDBs to determine whether their LSDBs are synchronized and have all the same LSPs as other routers for the appropriate levels and areas. Multicast CSNPs are sent periodically every 10 seconds by the DIS to ensure LSDB accuracy. (\*TBC\*) Adjacent IS-IS routers send PSNPs ~~to acknowledge the receipt of LSPs and~~ to request transmission of missing or newer LSPs. DIS and all ISs don't send explicit ACKs for each LSP.
- If there are too many LSPs to include in a single CSNP, they are sent in ranges. The CSNP header indicates the starting and ending LSP ID in the range. If all LSPs in the LSDB can be fit into a single CSNP, the range is set to default values – Start LSP ID (0x0000.0000.0000.00-00) and End LSP ID (0xFFFF.FFFF.FFFF.FF-FF).



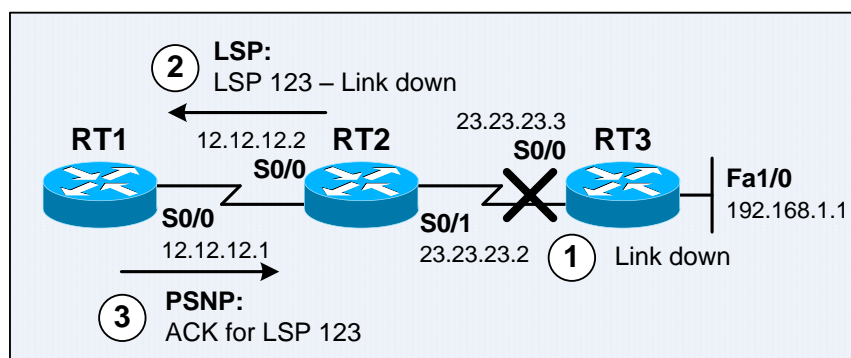
- On a LAN, the DIS periodically (every 10 seconds) multicasts CSNPs that list the LSPs in its LSDB to all L1 or L2 IS-IS routers on the LAN – the DIS is continually performing full LSDB synchronization with all IS-IS routers on the LAN. The DIS is responsible for flooding LSPs to all IS-IS routers on the LAN; having only the DIS to send LSPs minimizes the amount of traffic upon synchronizing the LSDBs.

**Note:** Unlike OSPF, IS-IS does not explicitly acknowledge LSPs flooded over a LAN.



**Figure 17-23:** IS-IS LSDB Synchronization on a Broadcast Network

- Figure above shows IS-IS LSDB synchronization over a LAN. RT4 compares the LSPs in the CSNP sent by the DIS with its LSDB. If its LSDB has a newer version of the LSP in the CSNP, or the CSNP does not include an LSP in its LSDB, it would multicast the LSP onto the LAN. In this case, RT4 is missing an LSP – LSP2. It sends a PSNP to the DIS (RT1) to request the missing LSP. Although the PSNP is a multicast packet, only the DIS will take action and reissue the missing LSP – LSP2 to RT4. Finally RT4 acknowledges it with a PSNP.
- Caution:** This scenario might be inaccurate, because no PSNPs were seen on LAN environments yet.
- CSNPs are not being periodically sent out on point-to-point links as on broadcast networks. A CSNP is sent only once upon a point-to-point link first comes up to synchronize the LSDB. After that, an LSP is sent upon topology change or IS-IS refresh, and it is being acknowledged using a PSNP.
- When an adjacency is established over a point-to-point link, each end router sends a CSNP that summarizes the LSP entries (LSP-ID, Sequence Number, Remaining Lifetime, and Checksum) in its LSDB to another router. When a router has an LSP in its LSDB that is not listed in the CSNP received from the other end, it would send the missing LSP to the other router; when a router realizes that its LSDB is missing any LSP as listed in the received CSNP, it would send a PSNP to request the full LSP. The LSP is then acknowledged via a PSNP. The minimumLSPTransmissionInterval timer (default 5 seconds) is set upon sending an LSP. The LSP is resent if the explicit PSNP acknowledge is not received before the timer expires.



**Figure 17-24:** IS-IS LSDB Synchronization over a Point-to-Point Link upon a Link Failure

- Figure above shows IS-IS LSDB synchronization over a point-to-point link upon a link failure.
  - 1) The link between RT2 and RT3 fails.
  - 2) RT2 notices the link failure and issues a new LSP describing the topology change.
  - 3) RT1 receives the LSP, stores it in its LSDB, and sends a PSNP back to RT2 to acknowledge the receipt of the LSP.
  
- IS-IS routers use the following processes to build the OSI forwarding database (or the CLNS routing table) to select the best path to a destination.
  - Performs SPF calculation twice upon the information in both L1 and L2 LSDBs to build the SPTs to OSI L1 and L2 devices (NETs).
  - Calculates ES reachability using **Partial Route Computation (PRC)** based on the L1 and L2 SPTs. There are no ESs in a pure IP Integrated IS-IS environment.
  - Inserts the best paths in the OSI forwarding database (or the CLNS routing table).
  
- Integrated IS-IS includes IP prefix reachability information in the LSPs, treating it as if it were ES information. IP subnets are treated the leaf objects of the IS-IS SPT. Therefore, processing IP reachability info requires only a PRC, similar to ES reachability.
  
- The PRC generates best-path for IP subnets and offers the routes to the IP routing table, where they are accepted based on normal *administrative distance* rule of IP routing table. IS-IS IP routes are shown as i L1 or i L2 routes accordingly in the IP routing table.
  
- The separation of IP reachability from the core IS-IS network architecture makes Integrated IS-IS much scalable than OSPF.
  - OSPF sends LSAs for individual IP subnets. An LSA will be flooded throughout the network upon an IP subnet failure, which all routers must run a full CPU-intensive SPF calculation upon the convergence process.
  - Integrated IS-IS builds the SPT using CLNS information. Since IP subnets are treated as leaf objects of the IS-IS SPT, the loss of an IP subnet does not affect the underlying CLNS architecture – the SPT is unaffected, and hence only required to perform a PRC. Any time an internal link between routers or a router fails, a full SPF calculation must be performed for that area.

- The routing process for IS-IS is divided into 4 stages:
  - **Update.** Routers forms neighbor relationship and exchange routing information between them using IIHs, LSPs, and SNPs prior to forwarding packets.
  - **Decision.** After the LSDBs have been synchronized, each router builds SPTs by placing itself at the root of the trees, and uses its LSDB to calculate the shortest paths to all devices within the same L1 area as well as to other L1 areas through the L2 backbone.
  - **Forwarding.** The forwarding database can then be built after the SPT has been built during the decision process. The forwarding table is essentially a lookup table for the longest match to forward and load balance packets multiple equal-cost paths. The forwarding table for Integrated IS-IS is more relevant to CLNS than to IP because the IP routing information is entered directly into the IP routing table, where IP routes are leaves on the IS-IS SPT.
  - **Receive.** If the frame is valid, the receive process passes user data and error reports to the forwarding process, whereas routing information Hellos, LSPs, and SNPs are sent to the update process. The receive process is not described in depth here because it is mainly applicable for CLNS routing. A detailed description is given in the ISO 10589 standards document.

- IS-IS LSPs contain the following 3 fields that help to determine whether a received LSP is more recent than the entry held in the LSDB, and even if it is corrupted.

- **Remaining Lifetime.** Used to age out expired LSPs. The IS-IS refresh interval is 15 minutes. If an LSP has been held in the LSDB for 20 minutes, it is assumed that the originating router is dead. When the lifetime expires, the LSP has the content removed, and leaving only the header. The lifetime is set to 0 and flooded through the network (to show that it is a new LSP). All receiving routers accept the incomplete LSP, recognize that the route is no longer valid, and purge the existing LSP from their LSDBs. IS-IS protects against flooding loops by decrementing the lifetime of an LSP by at least 1 at each flooding hop.

The **max-lsp-lifetime** {sec} IS-IS router subcommand sets the maximum time that LSPs can remain in the LSDB without being refreshed. The value set for the **lsp-refresh-interval** command must be less than the value set for this command; otherwise, LSPs will time out before they are refreshed.

If the LSP lifetime is misconfigured to be too low compared to the LSP refresh interval, Cisco IOS will reduce the LSP refresh interval to prevent the LSPs from timing out.

```
Router(config-router)#max-lsp-lifetime 800
% ISIS: max-lsp-lifetime should be greater than lsp-refresh-interval(900)
% ISIS: Setting lsp-refresh-interval to 770
Router(config-router)#do sh run | in lsp
max-lsp-lifetime 800
lsp-refresh-interval 770
Router(config-router)#
```

- **Sequence Number.** An unsigned 32-bit linear number. The 1st LSP is allocated the sequence number of 1, and subsequent LSPs are incremented by 1. Receiving a valid LSP that has the same sequence number as the one in the LSDB is simply ignored.
  - **Checksum.** Upon receiving an LSP that has an invalid checksum, the router would discard it and flood it as an expired LSP with a lifetime of 0. All routers purge the LSP, and the originating router resends a new LSP.
- IS-IS ignores incomplete fragment PDUs, which often caused by packet loss or corruption. Any fragmented LSP received is ignored if the starting fragment is not being received. The sequence of fragmented PDUs are indicated using the LSP Number (or Fragment ID) in the LSP ID field of the IS-IS PDU header.

- Below summarizes the technical differences between Integrated IS-IS and OSPF:

<b>Technology</b>	<b>Integrated IS-IS</b>	<b>OSPF</b>
Areas	Boundaries are defined on links. A router can only be in one area. <b>Note:</b> the multiarea IS-IS is mainly used during area migrations and transitions.	Boundaries are defined on routers. Interfaces can be in different areas. A router can be in many areas. An Integrated IS-IS L1 area is similar to an OSPF stub area.
Designated Router (DR)	A router with higher priority (or higher MAC address if the priority is same) will preempt and become the new DIS. Adjacencies are created between the DIS with all routers on a LAN. Each IS sends a multicast LSP to all ISs on the LAN. The LSP is unacknowledged.	A router with higher priority does not preempt the existing DR. Adjacencies are formed with the DR and BDR only on a LAN. All LSAs are acknowledged.
Encapsulation	Runs on top of the data link layer. A network layer protocol that defines its own Layer 3 PDUs. Fragmentation is the responsibility of Integrated IS-IS.	OSPF is an IP application. Has an OSPF header and encapsulated inside an IP packet. Fragmentation is the responsibility of IP.
LAN Flooding	All ISs maintain adjacencies with all other ISs on a LAN. DIS sends CSNP to all ISs. Periodic CSNPs (every 10 seconds) ensure the databases are synchronized.	Multicast updates and Hellos sent to the AllDRouters – DR and BDR. Unicast acknowledgments sent from all routers to the AllDRouters – DR and BDR.
LSAs	2 types of LSP – L1 LSP and L2 LSP. LSPs are TLV-encoded. Ignores and floods unrecognized LSPs. LSPs are always flooded across all media by the originating router.	7 types of LSA. Discards and does not flood unrecognized LSAs. Many small and separated LSAs for summary and external route updates. Every router generates LSUs.

## Integrated IS-IS Network Design Considerations

- Optimizing networks depends on careful planning and design. Although each network is constrained by physical, technical, and financial limitations, network architects must strive hard to design a network to fulfill the needs of its users and the demands of various applications. Network designs are often required to compromise between the trade-off of reliability and speed.
- Designing Integrated IS-IS networks with the hierarchical design requires the consideration of the data flow in addition to the bandwidth and CPU resources required by the routing protocol. Tuning the routing process to be more efficient allows the databases to converge more quickly, but often results in the compromise of resources and reliability.
- Additional resources, eg: CPU, memory, and network bandwidth for propagating IS-IS PDUs, are required for L1/L2 routers that handle the processing of L1 and L2.
- Below are some typical IS-IS network design considerations:
  - **A totally flat network that uses only L1 routing.** This design is not scalable because any change upon the network requires flooding of LSPs to all routers and trigger frequent router computation. However, this simplified design has some advantages, in which there will be only 1 link-state database and no problem of suboptimal routing between areas.
  - **A totally flat network using only L2 routing.** L1 areas can be easily added as the network expands. The L2 backbone has complete knowledge with the advantage of running a single SPF instance. L1 areas can be viewed as organizations connect to an ISP, in which new L1 areas would be added as new customers come online.
  - **A totally flat network using the Cisco default of every router running L1/L2 routing.** This allows for easy migration to a hierarchical design with multiple areas and avoids suboptimal routing when an L1 router always defaults to its nearest L1/L2 router. However, this design requires resources to maintain 2 separate L1 and L2 LSDBs.
  - **A hierarchical network where the core is running L2 routing, with L1 areas connecting into the core.** L1/L2 routers are used to connect the L1 and L2 areas. Although this is an excellent design, but there are still concerns that should be considered. This design results additional design and implementation efforts and the possibility of causing suboptimal routing. It requires an extensive knowledge of the network topology to ensure that a routing problem will not be compounded. The route leaking feature which introduced by recent IS-IS developments allowed allows L2 routers to pass specific routing information into L1 areas for facilitating the optimal routing decision.
- Scalability is achieved by minimizing the size of the LSDBs and routing tables, the number of network updates, and the amount of processing. Implement route summarization whenever possible. Route summarization can be accomplished only where the careful address planning permits grouping addresses by a common prefix. This condition is true for OSI and IP. Therefore, it is very important to carefully plan the IS-IS areas, NET addresses, and IP addresses.

## Integrated IS-IS Lab

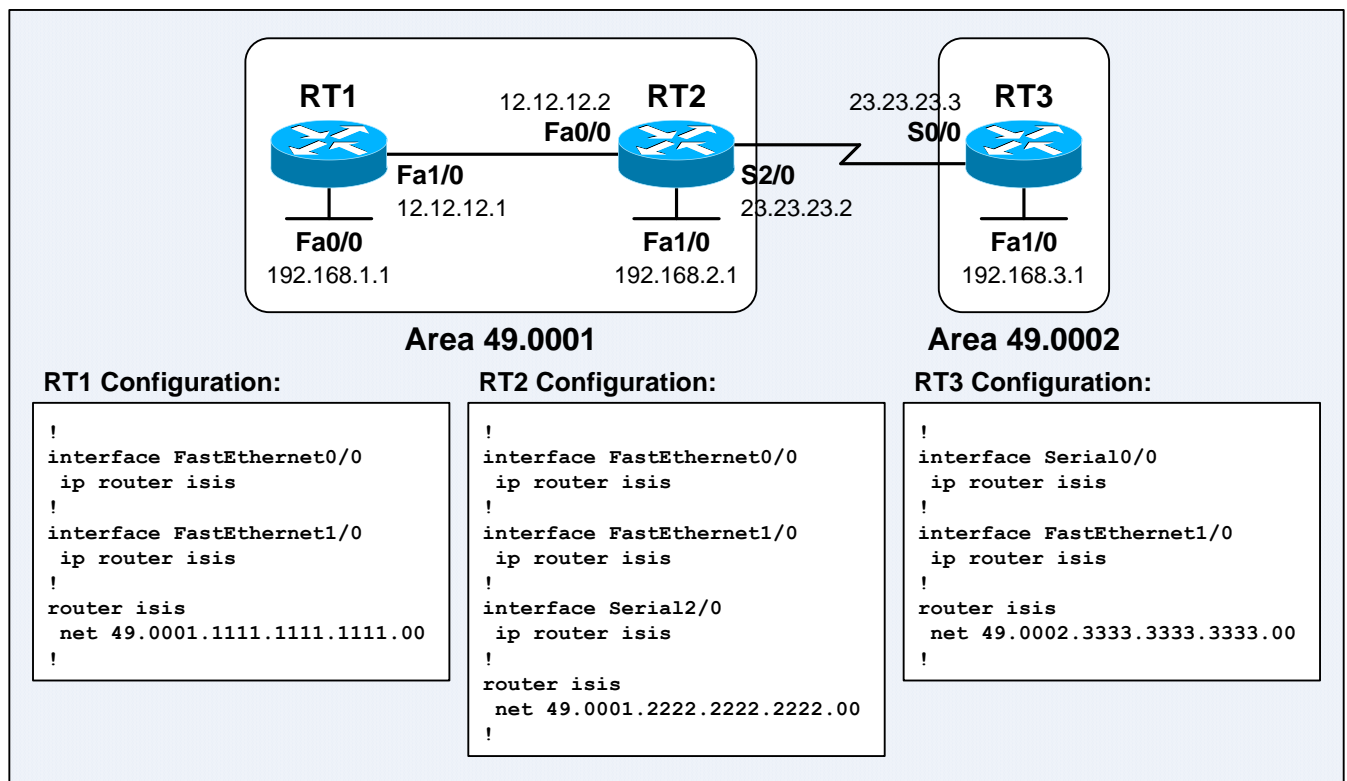


Figure 18-1: Sample IS-IS Multi-Area Network

- The **router isis** [area-tag] global configuration command specifies an IS-IS process, enables the Integrated IS-IS IP routing protocol, and assigns an optional tag to the process. The **ip router isis** [area-tag] interface subcommand enables an interface to participate in an IS-IS routing process. This is different from other IP routing protocols which define interfaces using the **network** router subcommands; there is no **network** command in Integrated IS-IS.
- The *area-tag* is the name for an IS-IS routing process. If it not specified, a null tag is assumed and the routing process is referenced with a null tag. The name must be unique among all IP and CLNS routing processes on a router. It is required for multiarea (multi-process to be precise) IS-IS configuration; and optional for conventional IS-IS configuration.
- Cisco routers are L1/L2 routers by default. This configuration is convenient because a router would inform other L1 routers that it is a L2 router which can forward traffic to other areas; and inform other L2 routers of the areas to which it is connected. However, it consumes more CPU, memory, and bandwidth resources for maintaining the L1 and L2 LSDBs at the same time.
- Be careful when configuring IP addressing on Integrated IS-IS routers, as it is difficult to troubleshoot IP address misconfigurations with IS-IS. The IS-IS neighborhood is established over OSI CLNS; not over IP. Both end routers of an IS-IS adjacency can have IP addresses on different subnets with no impact upon the operation of IS-IS.
- IP routing is enabled by default; CLNS routing is disabled by default. The **clns routing** global configuration command enables routing of CLNP packets. The **clns router isis** [area-tag] interface subcommand enable an interface for CLNS routing. **Note:** The **clns routing** command is not required for Integrated IS-IS to perform only IP routing. **Note:** The **clns routing** command should have been called **clnp routing**; because CLNS is actually the service for upper transport layers, while CLNP is the actual L3 routed protocol.

- Below shows the routing tables on RT1, RT2, and RT3:

```

RT1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

      23.0.0.0/24 is subnetted, 1 subnets
i L1  23.23.23.0 [115/20] via 12.12.12.2, FastEthernet1/0
      12.0.0.0/24 is subnetted, 1 subnets
C      12.12.12.0 is directly connected, FastEthernet1/0
C     192.168.1.0/24 is directly connected, FastEthernet0/0
i L1  192.168.2.0/24 [115/20] via 12.12.12.2, FastEthernet1/0
i L2  192.168.3.0/24 [115/30] via 12.12.12.2, FastEthernet1/0
RT1#
=====

RT2#sh ip route

Gateway of last resort is not set

      23.0.0.0/24 is subnetted, 1 subnets
C      23.23.23.0 is directly connected, Serial2/0
      12.0.0.0/24 is subnetted, 1 subnets
C      12.12.12.0 is directly connected, FastEthernet0/0
i L1  192.168.1.0/24 [115/20] via 12.12.12.1, FastEthernet0/0
C     192.168.2.0/24 is directly connected, FastEthernet1/0
i L2  192.168.3.0/24 [115/20] via 23.23.23.3, Serial2/0
RT2#
=====

RT3#sh ip route

Gateway of last resort is not set

      23.0.0.0/24 is subnetted, 1 subnets
C      23.23.23.0 is directly connected, Serial0/0
      12.0.0.0/24 is subnetted, 1 subnets
i L2  12.12.12.0 [115/20] via 23.23.23.2, Serial0/0
i L2  192.168.1.0/24 [115/30] via 23.23.23.2, Serial0/0
i L2  192.168.2.0/24 [115/20] via 23.23.23.2, Serial0/0
C     192.168.3.0/24 is directly connected, FastEthernet1/0
RT3#

```

- The **show clns EXEC** command displays general CLNS information on a router.

```

RT1#sh clns
Global CLNS Information:
  2 Interfaces Enabled for CLNS
  NET: 49.0001.1111.1111.1111.00
  Configuration Timer: 60, Default Holding Timer: 300, Packet Lifetime 64
  ERPDU's requested on locally generated packets
  Running IS-IS in IP-only mode (CLNS forwarding not allowed)
RT1#

```

- The **show clns protocol** EXEC command displays information for the IS-IS processes on a router.

```
RT1#sh clns protocol
IS-IS Router: <Null Tag>
  System Id: 1111.1111.1111.00  IS-Type: level-1-2
  Manual area address(es):
    49.0001
  Routing for area address(es):
    49.0001
  Interfaces supported by IS-IS:
    FastEthernet1/0 - IP
    FastEthernet0/0 - IP
  Redistribute:
    static (on by default)
  Distance for L2 CLNS routes: 110
  RRR level: none
  Generate narrow metrics: level-1-2
  Accept narrow metrics:   level-1-2
  Generate wide metrics:   none
  Accept wide metrics:     none
RT1#
```

- The **show clns interface {type num}** EXEC command displays information about the interfaces that are currently running IS-IS.

```
RT1#sh clns interface
FastEthernet0/0 is up, line protocol is up
  Checksums enabled, MTU 1497, Encapsulation SAP
  ERPDUs enabled, min. interval 10 msec.
  CLNS fast switching enabled
  CLNS SSE switching disabled
  DEC compatibility mode OFF for this interface
  Next ESH/ISH in 20 seconds
  Routing Protocol: IS-IS
    Circuit Type: level-1-2
    Interface number 0x0, local circuit ID 0x1
    Level-1 Metric: 10, Priority: 64, Circuit ID: RT1.01
    Level-1 IPv6 Metric: 10
    Number of active level-1 adjacencies: 0
    Level-2 Metric: 10, Priority: 64, Circuit ID: RT1.01
    Level-2 IPv6 Metric: 10
    Number of active level-2 adjacencies: 0
    Next IS-IS LAN Level-1 Hello in 1 seconds
    Next IS-IS LAN Level-2 Hello in 856 milliseconds
  --- output omitted ---
RT1#
```



- The **show clns neighbors** {type num} EXEC command displays the ES and IS neighbors. The optional **detail** keyword displays comprehensive information about the neighbors. Specify the optional interface type and number to list of neighbors across a particular interface.

```

RT1#sh clns neighbors

System Id      Interface  SNPA          State  Holdtime  Type Protocol
RT2            Fa1/0     cc01.0d0c.0000 Up      9         L1L2 IS-IS
RT1#
RT1#sh clns neighbors detail

System Id      Interface  SNPA          State  Holdtime  Type Protocol
RT2            Fa1/0     cc01.0d0c.0000 Up      9         L1L2 IS-IS
  Area Address(es): 49.0001
  IP Address(es):  12.12.12.2*
  Uptime: 00:01:17
  NSF capable
RT1#

```

- The **show isis route** EXEC command displays the IS-IS L1 routing table, which includes all other System IDs within the same area. This command is available only if CLNS routing is enabled both globally (with the **clns routing** global configuration) and at the interface level (with the **clns router isis** interface subcommand). CLNS routing is not being implemented and enabled in this scenario.

```

RT1#sh isis route

IS-IS not running in OSI mode (*) (only calculating IP routes)

(*) Use "show isis topology" command to display paths to all routers
RT1#

```

- The **show clns route** EXEC command displays the IS-IS L2 routing table. CLNS routing is not being implemented in this scenario therefore it only shows the directly connected subnets.

```

RT1#sh clns route
Codes: C - connected, S - static, d - DecnetIV
       I - ISO-IGRP, i - IS-IS, e - ES-IS
       B - BGP,      b - eBGP-neighbor

C 49.0001.1111.1111.1111.00 [1/0], Local IS-IS NET
C 49.0001 [2/0], Local IS-IS Area

RT1#

```

- The **show isis database** EXEC command displays contents of the IS-IS LSDB. Issue the **clear isis \*** privileged command to force IS-IS to refresh its LSDB and recalculate all routes.

```
RT1#sh isis database

IS-IS Level-1 Link State Database:
LSPID          LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
RT1.00-00      * 0x00000004  0x42EE        1148           1/0/0
RT2.00-00      0x00000004  0x9039        1146           1/0/0
RT2.01-00      0x00000001  0x2630        1121           0/0/0
IS-IS Level-2 Link State Database:
LSPID          LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
RT1.00-00      * 0x00000004  0x2D18        1128           0/0/0
RT2.00-00      0x00000005  0xF505        1141           0/0/0
RT2.01-00      0x00000001  0xB529        1121           0/0/0
RT3.00-00      0x00000002  0xAC54        1139           0/0/0
RT1#
```

- The **show isis topology** EXEC command displays the IS-IS L1 and L2 topology tables, which show the least-cost IS-IS paths to the ISs.

```
RT1#sh isis topology

IS-IS paths to level-1 routers
System Id      Metric  Next-Hop      Interface      SNPA
RT1            --
RT2            10      RT2            Fa1/0          cc01.0d0c.0000

IS-IS paths to level-2 routers
System Id      Metric  Next-Hop      Interface      SNPA
RT1            --
RT2            10      RT2            Fa1/0          cc01.0d0c.0000
RT3            20      RT2            Fa1/0          cc01.0d0c.0000
RT1#
```

- The **show isis neighbors** EXEC command displays brief information about directly connected IS-IS neighbors.

```
RT1#sh isis neighbors

System Id      Type Interface IP Address      State Holdtime Circuit Id
RT2            L1  Fa1/0  12.12.12.2     UP    8          RT2.01
RT2            L2  Fa1/0  12.12.12.2     UP    7          RT2.01
RT1#
RT1#sh isis neighbors detail

System Id      Type Interface IP Address      State Holdtime Circuit Id
RT2            L1  Fa1/0  12.12.12.2     UP    8          RT2.01
  Area Address(es): 49.0001
  SNPA: cc01.0d0c.0000
  State Changed: 00:01:17
  LAN Priority: 64
  Format: Phase V
RT2            L2  Fa1/0  12.12.12.2     UP    7          RT2.01
  Area Address(es): 49.0001
  SNPA: cc01.0d0c.0000
  State Changed: 00:01:17
  LAN Priority: 64
  Format: Phase V
RT1#
```

- The **show isis spf-log** EXEC command displays the last 20 occurrences about when and why an IS-IS router has performed a full SPF calculation.

```

RT1#sh isis spf-log

  level 1 SPF log
  When   Duration  Nodes  Count  First trigger LSP   Triggers
00:01:21      0      1      2      RT1.00-00 PERIODIC NEWLSP
00:01:11      4      3      4      RT1.00-00 NEWADJ NEWLSP TLVCONTENT
00:00:45      4      3      3      RT2.00-00 ATTACHFLAG LSPHEADER

  level 2 SPF log
  When   Duration  Nodes  Count  First trigger LSP   Triggers
00:01:21      0      1      2      RT1.00-00 PERIODIC NEWLSP
00:01:11      4      3      4      RT1.00-00 NEWADJ NEWLSP TLVCONTENT
00:00:51      4      4      2      RT2.00-00 NEWLSP TLVCONTENT

RT1#

```

- Below lists the possible triggers of a full SPF calculation:

Trigger	Description
ATTACHFLAG	The router is now attached to or has just lost contact to the L2 backbone.
ADMINDIST	Another administrative distance was configured for the IS-IS process on the router.
AREASET	Set of learned area addresses in this area changed.
BACKUPOVFL	An IP prefix disappeared. The router knows there is another way to reach that prefix but not stored that backup route. The only way to find the alternative route is through a full SPF run.
DBCHANGED	The <b>clear isis *</b> privileged command was issued on the router.
IPBACKUP	An IP route disappeared, which was not learned via IS-IS, but via another routing protocol with a better administrative distance. IS-IS will run a full SPF to install an IS-IS route for the disappeared IP prefix.
IPQUERY	The <b>clear ip route</b> privileged command was issued on the router.
LSPEXPIRED	An LSP in the LSDB has expired.
LSPHEADER	The ATT/P/OL bits or the is-type in an LSP header has changed.
NEWADJ	The router has established a new adjacency to another router.
NEWAREA	A new area (via the Network Entity Title – NET) was configured on the router.
NEWLEVEL	A new level (via the is-type) was configured on the router.
NEWLSP	A new router or pseudonode appeared in the IS-IS topology.
NEWMETRIC	A new metric was configured on an interface of the router.
NEWSYSID	A new System ID (via the NET) was configured on the router.
PERIODIC	An IS-IS router runs a periodic full SPF calculation every 15 minutes.
RTCLEARED	The <b>clear clns route</b> privileged command was issued on the router.
TLVCODE	TLV code mismatch, indicating that different TLVs are included in the newest version of an LSP.
TLVCONTENT	TLV contents changed. This normally indicates that an adjacency somewhere in the area has come up or gone down. The Last trigger LSP column indicates where the instability may have occurred.

## Tuning and Optimizing IS-IS

- The Cisco IOS IS-IS configuration that configures an L1/L2 IS-IS router by default can result in the inefficient use of router and network resources and results in suboptimal routing. Although this configuration has the advantage of allowing all routers to communicate and converge without much administrative effort, it is not the most efficient way to build an IS-IS network, because routers with the default configurations send out both L1 and L2 Hellos and maintain both L1 and L2 LSDBs. Therefore, a network administrator must know how to tune IS-IS to conserve memory and bandwidth resources (only need to maintain the LSDB and send Hellos, LSPs, and SNPs for the necessary level) to achieve efficient and optimum performance.

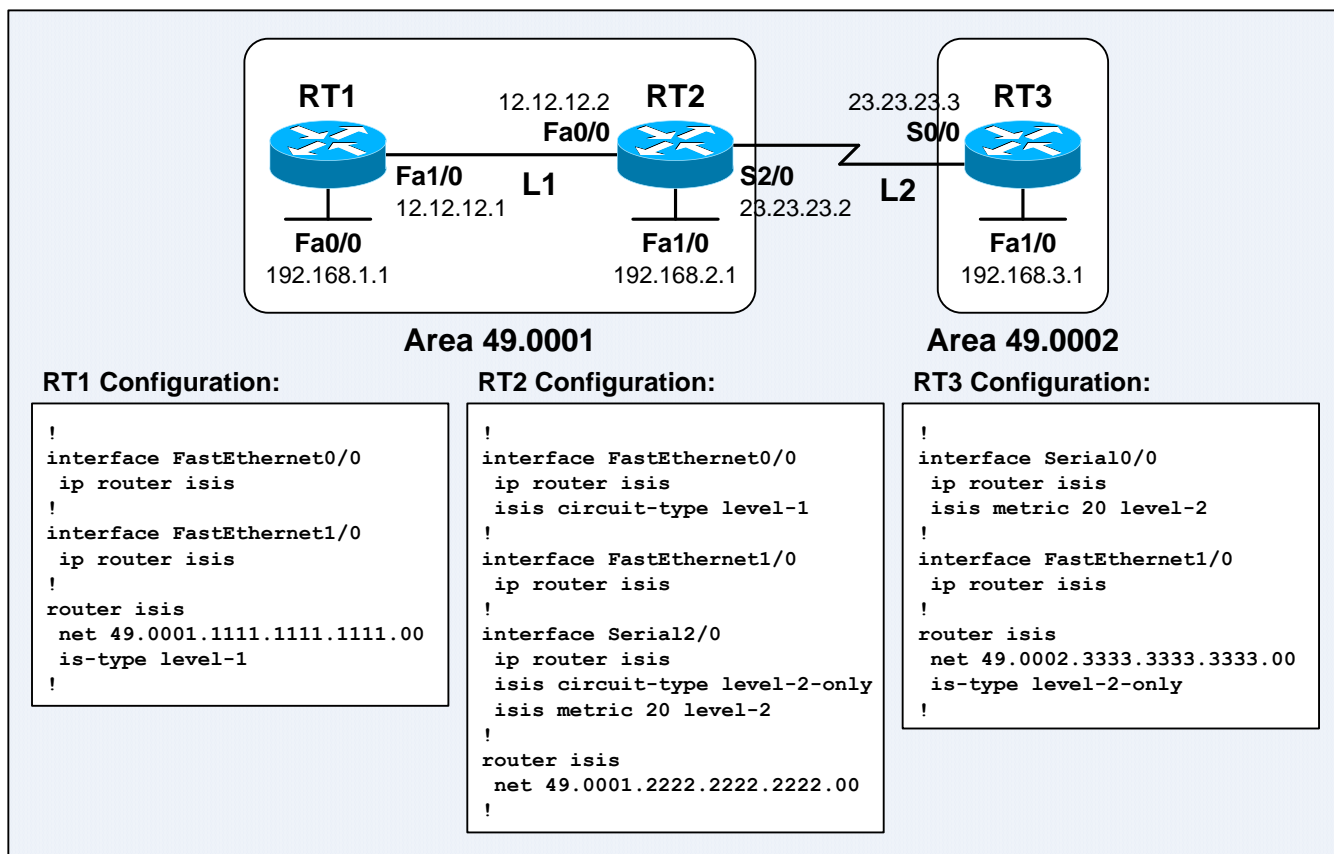


Figure 18-2: Tuning IS-IS

- The **is-type level-1** and **is-type level-2-only** IS-IS router subcommands specify a router to act only as an internal area (L1) router or as a backbone (L2) router respectively. The default configuration is **is-type level-1-2**; this is not shown in the router configuration.
- Although a router can be an L1/L2 router, it might not be required to establish both L1 and L2 adjacencies over all interfaces. If the router only needs to establish adjacency with an L1 router through a particular interface, it doesn't need to send L2 Hellos out that interface, and vice versa. It wastes router and bandwidth resources to try to establish adjacencies that do not exist. Configure an interface to send only L1 or L2 Hellos using the **isis circuit-type {level-1 | level-2-only}** interface subcommand to tune IS-IS in such situations. Cisco IOS attempts to establish both L1 and L2 adjacencies over an interface (**level-1-2**) by default.  
**Warning:** Changing the **isis circuit-type** will tear down existing adjacencies over the interface.

- Unlike most other IP routing protocols, Integrated IS-IS does not take account of the line speed or bandwidth when setting the link metrics. All interfaces are assigned a metric value of 10, which results in suboptimal routing for networks with links of varying types and speeds. The **isis metric** {metric} {level-1 | level2} interface subcommand changes the metric value (from 1 to 63). An interface can have different L1 and L2 metric values.
- Below verifies the IS-IS configuration on RT1.  
It has a default route `i*L1 0.0.0.0/0` to the nearest L1/L2 router – RT2.

```

RT1#sh ip route

Gateway of last resort is 12.12.12.2 to network 0.0.0.0

    12.0.0.0/24 is subnetted, 1 subnets
C       12.12.12.0 is directly connected, FastEthernet1/0
C       192.168.1.0/24 is directly connected, FastEthernet0/0
i L1 192.168.2.0/24 [115/20] via 12.12.12.2, FastEthernet1/0
i*L1 0.0.0.0/0 [115/10] via 12.12.12.2, FastEthernet1/0
RT1#
RT1#sh clns protocol

IS-IS Router: <Null Tag>
  System Id: 1111.1111.1111.00  IS-Type: level-1
  Manual area address(es):
    49.0001
  Routing for area address(es):
    49.0001
  Interfaces supported by IS-IS:
    FastEthernet1/0 - IP
    FastEthernet0/0 - IP
  Redistribute:
    static (on by default)
  Distance for L2 CLNS routes: 110
  RRR level: none
  Generate narrow metrics: level-1-2
  Accept narrow metrics:   level-1-2
  Generate wide metrics:   none
  Accept wide metrics:     none
RT1#
RT1#sh isis database

IS-IS Level-1 Link State Database:
LSPID                LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
RT1.00-00             * 0x00000003  0x3A02        1123          0/0/0
RT2.00-00             0x00000004  0x1F62        1148          1/0/0
RT2.01-00             0x00000001  0x2630        1122          0/0/0
RT1#
RT1#sh isis neighbors

System Id           Type  Interface  IP Address      State Holdtime  Circuit Id
RT2                L1  Fa1/0      12.12.12.2     UP      9          RT2.01
RT1#

```

## Route Summarization

- Route summarization provides many advantages, which includes reducing the router resources (CPU and memory) required due to reduced routing table size, and scoping or isolating route flapping issues within an area. When a router is unaware of a change or problem in the network, its LSDB is not being updated, and SPF recalculation is not being performed. Summarization allows routers within areas to only maintain and manage internal knowledge of the areas they reside and summarize the knowledge across area boundaries.
  - IS-IS allows manual summarization with some limitations. IS-IS internal route summarization within an area is not viable; therefore it cannot be implemented and configured on an L1 router. Multiple L1 internal routes can be summarized between areas as a single L2 route. An L1/L2 router (area border router) is where route summarization is being implemented.
  - IS-IS route summarization requires extra attention when there are multiple L1/L2 routers between 2 areas. If one of them is summarizing routes, others must perform summarization too. As if one of them is advertising a summary route while others advertise the more specific routes, packets will be routed through other routers due to the most-specific prefix matching rule.
  - The rules for route summarization on an OSPF ABR are applicable to Integrated IS-IS. Below lists the rules for summarizing IP routes in Integrated IS-IS:
    - L1 routes cannot be summarized within an area. <sup>[1]</sup>
    - L1/L2 routers can summarize the internal routes within their areas. The summarized routes are propagated into the L2 backbone; similar to summarization on an OSPF ABR.
    - If route summarization is configured on an L1/L2 router, the same configuration must also be implemented on all L1/L2 routers reside in the area to prevent suboptimal routing.
- [1] Cisco IOS allows the configuration of the **summary-address** IS-IS router subcommand on an L1 router without any warning message; though it has no effect on an L1 router.

- The **summary-address** *{prefix} {mask} [level-1 | level-1-2 | level-2] [metric metric]* IS-IS router subcommand creates an L1 or L2 aggregate or summarized address. External routes learned from other routing protocols can also be summarized.

<b>level-1</b>	Only routes redistributed into L1 are summarized with the configured <i>prefix mask</i> .
<b>level-1-2</b>	Summary routes are applied when redistributing routes into L1 and L2 IS-IS, and when L2 IS-IS advertises L1 routes as reachable in its area.
<b>level-2</b>	Routes learned by L1 routing are summarized into the L2 backbone with the configured <i>prefix mask</i> . Routes redistributed into L2 IS-IS will be summarized too.

- The following line will be seen on an L1/L2 summarizing router.  

```
i su prefix/mask [115/10] via 0.0.0.0, Null0
```
- The `i su` code indicates an IS-IS summary route; while the Null route is inserted into the routing table automatically upon configuring an IS-IS summary route to prevent routing loops from occurring in case receiving packets that match the summary route but the summarizing router does not have a more-specific route.
- The drawback of route summarization is that routers might have lesser information to calculate the most optimal paths for all destinations.

## IS-IS Authentication

- IS-IS supports authentication for a link, an L1 area, or a L2 domain. Routers must exchange IIHs and LSPs for the corresponding L1 and L2 routing levels using the mutually agreed passwords. Authentication is not enabled by default.
- Cisco IOS supports the following 3 types of IS-IS authentication:
  - **IS-IS authentication** – Clear-text IS-IS authentication.
  - **IS-IS HMAC-MD5 authentication** – Inserts an HMAC-MD5 digest TLV in IS-IS PDUs.
  - **Enhanced clear-text authentication** – Clear-text IS-IS authentication using a series of authentication key chain and authentication mode commands that provide easier password management and modification.

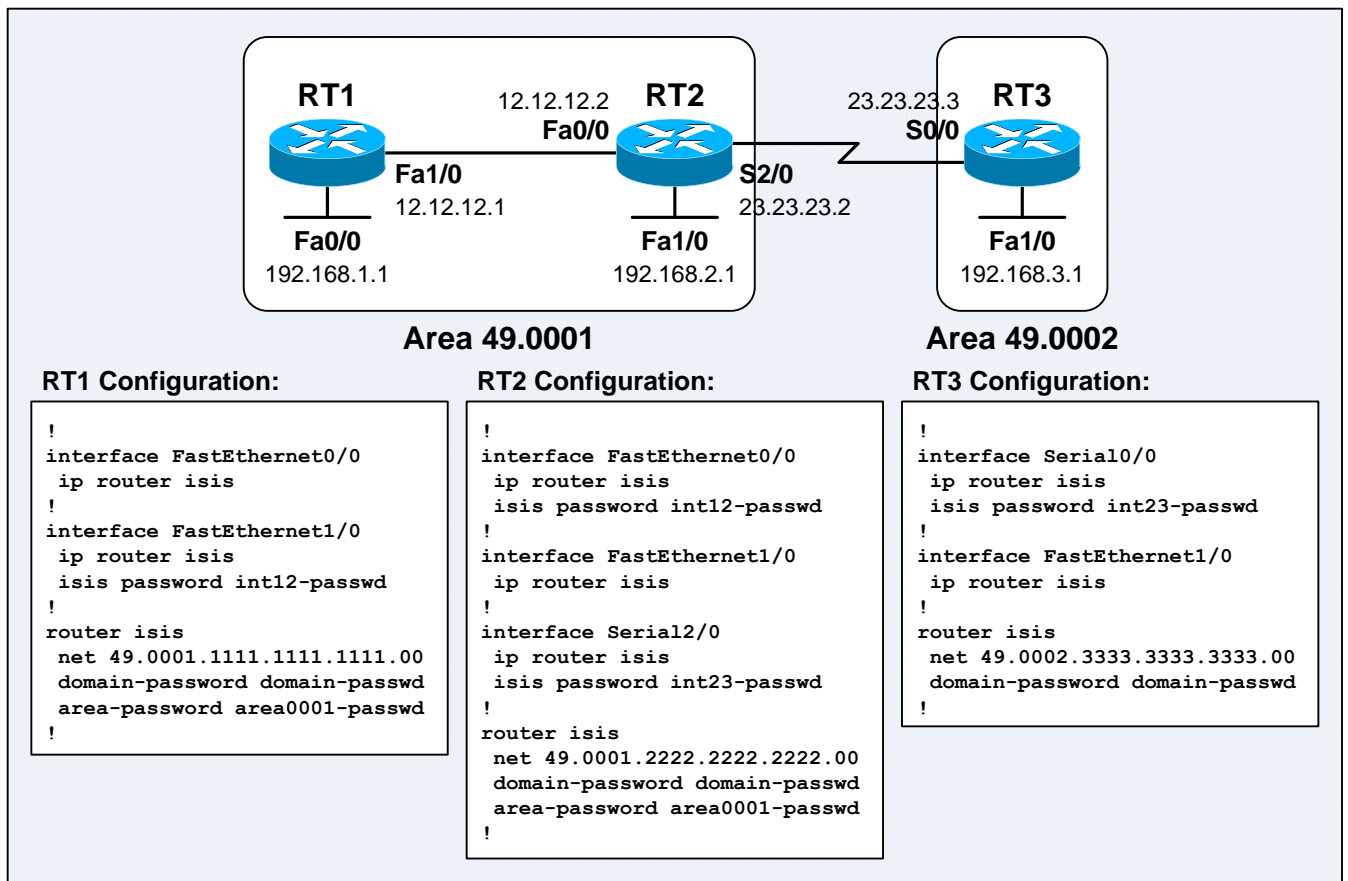
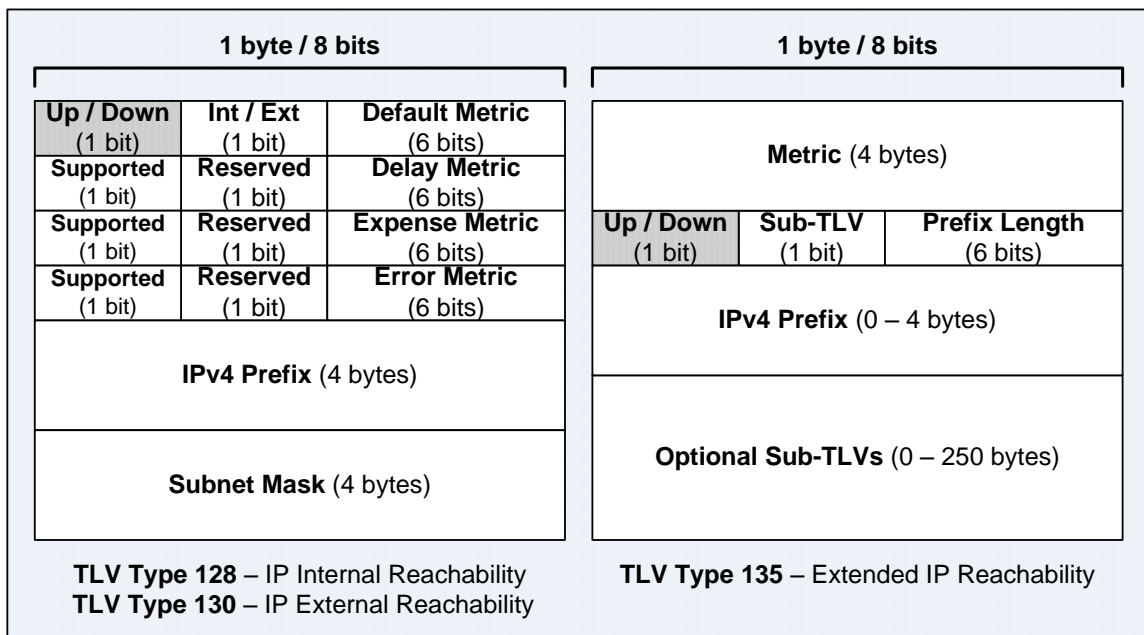


Figure 18-3: IS-IS Interface, Area, and Domain Authentications

- The **isis password {passwd} [level-1 | level-2]** interface subcommand configures IS-IS authentication for L1 or L2 routing on an interface in order to prevent forming adjacencies with unauthorized routers. If the routing level is not specified, the router will enable both levels and send out L1 and L2 IIHs that contain the Authentication TLV.
- The **area-password {passwd} [authenticate snp {send-only | validate}]** IS-IS router subcommand configures an IS-IS area authentication password in order to prevent receiving false routing information from unauthorized routers. The router inserts the Authentication TLV into L1 LSPs, as well as L1 CSNPs and L1 PSNPs with the optional **authenticate snp** keyword.
- The **domain-password {passwd} [authenticate snp {send-only | validate}]** IS-IS router subcommand configures the IS-IS routing domain authentication password. The router inserts the Authentication TLV into L2 LSPs, as well as L2 CSNPs and L2 PSNPs with the optional **authenticate snp** keyword.

## IS-IS Route Leaking

- Packets destined to other L1 areas are routed to the nearest L1/L2 router and to be forwarded to the destination area. Routing to the nearest L1/L2 router can lead to suboptimal routing when the shortest path to the destination area is via a different L1/L2 router. Route leaking is a mechanism for leaking or redistributing L2 information into L1 areas in order to reduce suboptimal routing. By having more details about the inter-area routes, an L1 router is able to make a better choice upon which L1/L2 router to forward the packets to a particular destination area.
- **RFC 2966 – Domain-wide Prefix Distribution with Two-Level IS-IS** defines route leaking for use with the 6-bit narrow metric TLV types 128 and 130; while **RFC 5305 – IS-IS Extensions for Traffic Engineering** defines route leaking for use with the 32-bit wide metric TLV type 135. Both RFCs define an Up/Down bit to indicate whether or not the route advertised in the LSP has been leaked. An Up/Down bit of 0 indicates that a route was originated within an L1 area; while L1/L2 routers set the Up / Down bit to 1 for prefixes or routes that were derived from L2 routing and advertised into L1 LSPs that get propagated into the L1 area. The Up/Down bit is used to prevent routing loops. An L1/L2 router never readvertises any L1 route with the Up/Down bit set back into the L2 backbone.



**Figure 18-4:** The Up / Down bit for IS-IS Route Leaking

- The **redistribute isis ip level-2 into level-1 distribute-list {100 – 199}** IS-IS router subcommand implement IS-IS route leaking. An IP extended access list must be defined to match the routes that are to be leaked from L2 into L1. The **metric-style wide** IS-IS router subcommand is optional but recommended.  
**Note:** If wide-style metric is not enabled, the metric in the TLV will be interpreted wrongly – more than 63! Because the Up/Down bit is also being interpreted along with the 6-bit metric. 😊
- Cisco documentations mentioned that the command syntax differs for 12.0S and 12.2S Cisco IOS releases, which uses the **advertise ip l2-into-l1 {100 – 199}** IS-IS router subcommand. Those releases only support route leaking using TLV type 135; therefore the **metric-style wide** IS-IS router subcommand must be configured when implementing route leaking.  
**Note:** Recent 12.0S and 12.2S Cisco IOS releases use the **redistribute** command to implement route leaking.



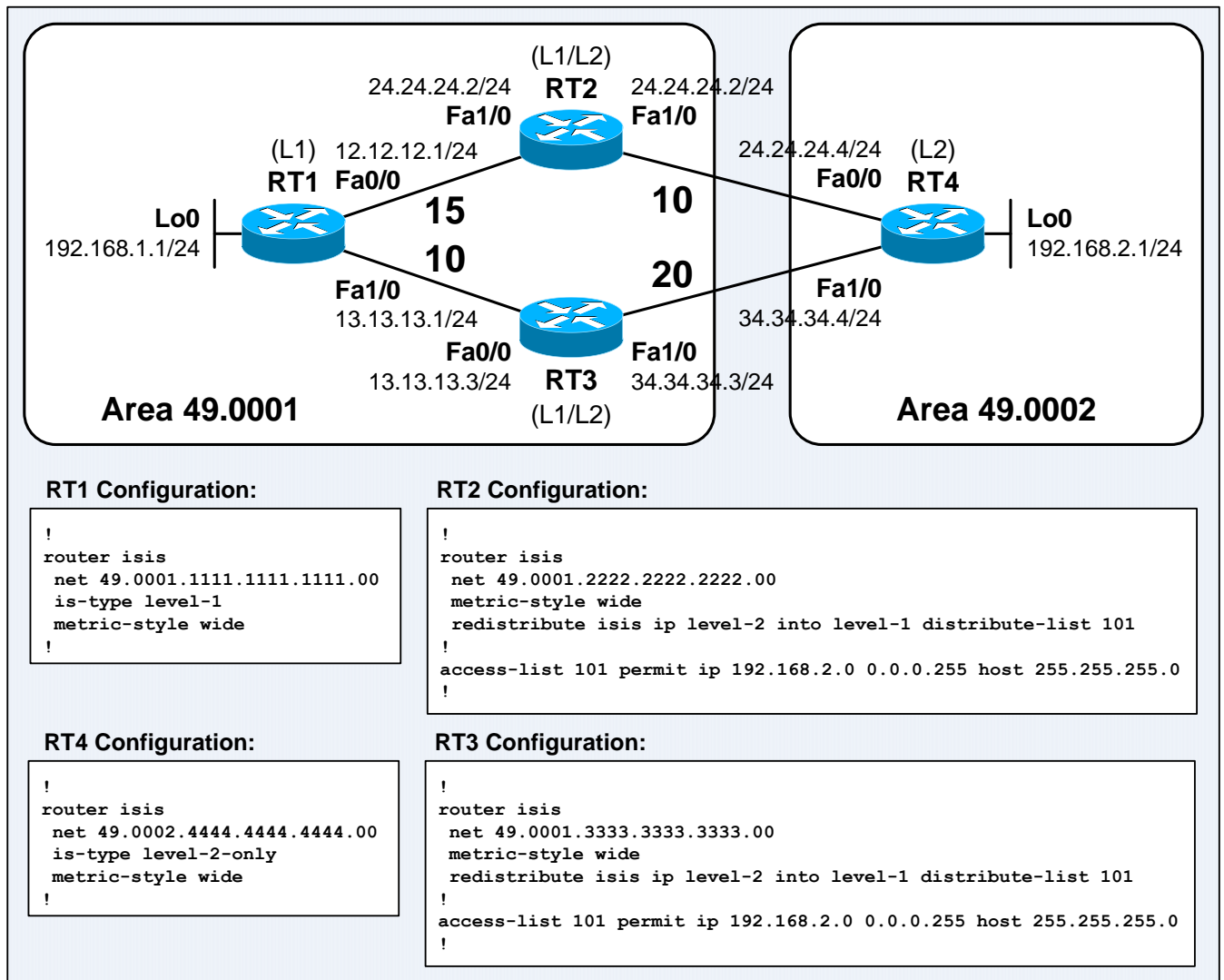


Figure 18-5: IS-IS Route Leaking

- In the sample scenario above, RT1 routes packets destined to 192.168.2.0/24 to the nearest L1/L2 router – RT3 by default. However, that route is not the most optimal path.
- Below shows the IP routing table on RT1 after implemented route leaking on RT2 and RT3. RT1 will now use the path via RT2 instead of RT3 to reach 192.168.2.0/24. Route leaking allows RT1 to determine the true cost to reach 192.168.2.0/24 and forward packets accordingly.

```

RT1#sh ip route

Gateway of last resort is 13.13.13.3 to network 0.0.0.0

    12.0.0.0/24 is subnetted, 1 subnets
C       12.12.12.0 is directly connected, FastEthernet0/0
C       192.168.1.0/24 is directly connected, Loopback0
    13.0.0.0/24 is subnetted, 1 subnets
C       13.13.13.0 is directly connected, FastEthernet1/0
i ia 192.168.2.0/24 [115/35] via 12.12.12.2, FastEthernet0/0
i*L1 0.0.0.0/0 [115/10] via 13.13.13.3, FastEthernet1/0
RT1#

```

- Below shows the L1 LSP generated by RT2 for the leaked route – 192.168.2.0/24. Note that wide-style metric has been enabled on all routers in this sample scenario.

```

RT1#sh isis database RT2.00-00 detail

IS-IS Level-1 LSP RT2.00-00
LSPID                LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
RT2.00-00            0x00000005   0x6884        1194          1/0/0
  Area Address: 49.0001
  NLPID:         0xCC
  Hostname: RT2
  IP Address:    12.12.12.2
  Metric: 15    IP 12.12.12.0/24
  Metric: 15    IS-Extended RT2.01
  Metric: 20    IP-Interarea 192.168.2.0/24
RT1#

```

- Leaked routes are called **inter-area routes**, which shown as IP-Interarea in the IS-IS LSDB and marked with an `ia` designation in the IP routing table.
- 2 common BGP practices benefit greatly from the route leaking ability:
  - One criterion used in the BGP path-selection process is the IGP cost to the BGP next-hop address. Many ISPs rely on the IGP metric to choose the best path through their ASs. This practice is known as *shortest exit routing*.
  - Another common practice is to use the IGP metric for the value of the MED when advertising routes to other ASs. This provides the ability to request other ASs to use the shortest path through the AS when making routing decisions.

Before route leaking, if multiple areas were used within the AS, the IS-IS metric did not represent the true internal cost and did not work well with either of these practices. Leaking routes for all the BGP next-hop addresses implements a multiarea hierarchy while maintaining accurate end-to-end IGP metrics.
- In MPLS VPN environments, reachability information for the loopback addresses of every **Provider Edge (PE)** router is needed. Leaking routes for the PE loopback addresses allows a multiarea hierarchy to be used in MPLS VPN implementations.
- Route leaking can also be used to implement a crude form of traffic engineering. Leaking routes for individual addresses or services from specific L1/L2 routers control the exit point from an L1 area used to reach those addresses and services.
- By default, IS-IS L1/L2 routers set the ATTached bit in their LSPs generated for their L1 areas. This original concept soon became obsolete after the route leaking capability was introduced. The **ignore-attached-bit** hidden IS-IS router subcommand can be configured on an L1/L2 router to ignore the ATTached bit in the LSPs from another L1/L2 router, and therefore does not install the IS-IS-learned default route into the IP routing table.

*This page is intentionally left blank*

## IP Multicast Routing

- Many types of data can be transferred between devices over an IP network, eg: document files, image files, voice, video, etc. However, IP network is not efficient to deliver the same data to many destinations – the data is sent in unicast packets and being replicated for each destination, which would unnecessarily consume many network bandwidth resources.
- IP multicast technology delivers data over networks to a group of destinations in the efficient way. The data is sent from the source as a single data stream that travels as far as it can in the network. Network devices only replicate the data when they need to send it out on multiple interfaces or segments to reach all members of the multicast group. Multicast traffic is generally unidirectional.
- IP multicast groups are identified by the Class D IP addresses, which are in the range from 224.0.0.0 to 239.255.255.255. IP multicast introduced some new network protocols, including 2 for informing the network devices for which end systems require which multicast data stream – **Internet Group Management Protocol (IGMP)** and **Cisco Group Management Protocol (CGMP)**; and a few for determining the best way to route multicast traffic – **Distance Vector Multicast Routing Protocol (DVMRP)**, **Multicast OSPF (MOSPF)**, **Core-Based Trees (CBT)**, and **Protocol Independent Multicast (PIM)**.
- When IP multicast is used to send packets to multiple receivers, the packets are not duplicated for every receiver, but instead are sent in a single stream using a single copy of each packet. Downstream routers and switches replicate the packets only on links where receivers exist. The sender or source of multicast does not have to know the unicast addresses of the receivers. Unicast transmission sends multiple copies of data packets, one copy for each receiver.

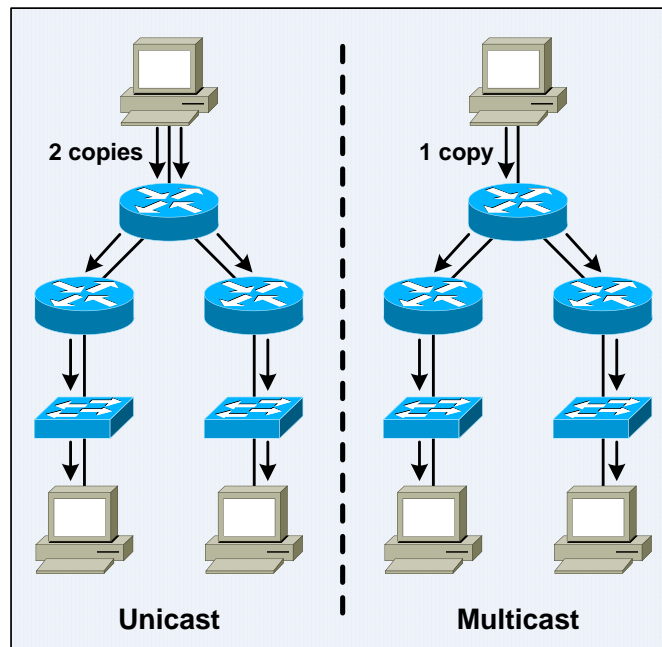


Figure 19-1: Unicast vs Multicast

- The most common types of multicast applications are **one-to-many** and **many-to-many**. In one-to-many applications, one sender sends data to many receivers, eg: applications used for audio or video distribution, push-media, announcements, monitoring, etc. If the sender of a one-to-many application needs feedback from the receivers, it becomes a many-to-many application. In many-to-many applications, multiple hosts can send data to the same multicast group, many receivers also act as senders, and a host can act as a sender and receiver at the same time. Receiving data from multiple sources increases the complexity of applications and introduces various management challenges. Using the many-to-many multicast concept, new applications, eg: collaboration, concurrent processing, distributed interactive simulations, can then be built.
- Another model of multicast applications is **many-to-one**, mainly used in financial applications, in which many receivers are sending data back to one sender (via unicast or multicast) that may be used for resource discovery, data collection, auctions, polling, etc.
- Many new multicast applications are emerging as demand. Real-time applications include live TV, radio, corporate broadcasts, financial data delivery, whiteboard collaboration, e-learning or distance learning, and videoconferencing. Non-real-time applications include file transfer, data and file replication, and video on demand (VoD). Ghosting multiple PC images simultaneously is a file transfer application. Some types of e-learning are also non-real-time.  
**Note:** Some web technologies, eg: webcasting – transmission of audio or video over the Internet, actually use unicast to deliver data to each individual media player through several data streams.
- Below lists some advantages of multicast transmission over unicast transmission:
  - **Enhanced efficiency.** Available network bandwidth and end system resources are utilized more efficiently as multiple data streams are replaced with a single data stream.
  - **Optimized performance.** The sender requires much less processing resources and network bandwidth as fewer copies of data require forwarding and processing.
  - **Support for distributed applications.** Distributed multipoint applications using unicast transmission do not scale well as the demand and usage grows, as the traffic load and the number of clients increase proportionally at a 1:1 rate with unicast transmission. Multicast enables many new applications that were not possible with unicast.
- Most multicast applications that are implemented based upon the User Datagram Protocol (UDP) have some disadvantages when compared to unicast applications implemented upon the Transmission Control Protocol (TCP). Below lists some disadvantages of multicast applications:
  - UDP does not have any reliability mechanisms. Reliability issues must be addressed by the multicast applications themselves if reliable data transmission is necessary.
  - UDP best-effort delivery without the acknowledgement process results in occasional packet drops. Multicast applications must not expect reliable data delivery and should be designed accordingly to handle reliable data transmission at the application layer.
  - UDP without any congestion control mechanism, eg: windowing and slow-start as in TCP, may result in network congestion and service degradation as the popularity and wide deployment of UDP-based multicast applications. Multicast applications should be designed to attempt to detect and avoid congestion conditions if possible.
  - Duplicate packets may be generated upon the changes of multicast network topologies. Multicast applications should be designed to expect occasional duplicate packets and handle them accordingly.
  - Out-of-order packet delivery may occur upon the convergence of the underlying network. Multicast applications should be designed to handle them accordingly.
  - Some security issues with multicast, eg: restricting multicast traffic to be delivered to a selected group of receivers to avoid eavesdropping, are not successfully addressed yet. Some commercial applications will become possible only when these issues are resolved.

## Multicast IP and MAC Addresses

- The **Internet Assigned Numbers Authority** (IANA) assigns the ranges and multicast addresses. The latest list can be obtained from <http://www.iana.org/assignments/multicast-addresses/>. Multicast IP addresses use the Class D address space, which is indicated by the high-order 4 bits set to binary 1110. Therefore, the multicast address range is 224.0.0.0 to 239.255.255.255.
- **Local Scope multicast addresses** are in the range of 224.0.0.0 to 224.0.0.255 and are reserved for network protocol use. This address range is also known as the **local network control block**. Packets destined for multicast addresses in this range are never forwarded out from the local network broadcast domain regardless of the Time to Live (TTL) field in the IP packet header. The TTL is usually set to 1. Below lists some examples of local scope multicast IP addresses:

<b>224.0.0.1</b>	All multicast systems on the subnet
<b>224.0.0.2</b>	All multicast routers on the subnet
<b>224.0.0.4</b>	All Distance Vector Multicast Routing Protocol (DVMRP) routers on the subnet
<b>224.0.0.5</b>	All Open Shortest Path First (OSPF) routers on the subnet
<b>224.0.0.6</b>	All Open Shortest Path First (OSPF) designated routers (DRs) on the subnet
<b>224.0.0.9</b>	All Routing Information Protocol Version 2 (RIPv2) routers on the subnet
<b>224.0.0.10</b>	All Enhanced Interior Gateway Routing Protocol (EIGRP) routers on the subnet

- Transient multicast IP addresses are dynamically assigned for multicast applications and then returned for others to use when no longer needed. Transient multicast IP addresses are assigned from the remainder of the IP multicast address space and are divided into the following 2 types:
  - **Globally-scoped multicast addresses**, in the range of 224.0.1.0 to 238.255.255.255, are to be allocated dynamically throughout the Internet. Ex: The 224.2.0.0/16 range may be used for Multicast Backbone (Mbone) applications. Mbone consists of Internet routers that support IP multicasting and form a virtual network (multicast channel) for delivering various public and private audio and video programs. Mbone was originally created by the IETF to multicast audio and video meetings.
  - **Limited-scoped or Administratively-scoped multicast addresses**, in the range of 239.0.0.0 to 239.255.255.255. RFC 2365 – Administratively Scoped IP Multicast defines that these multicast addresses are reserved for use inside the private domains. The Administratively-scoped multicast address space includes the following scopes:
    - **Site-Local Scope** – 239.252.0.0/16, 239.253.0.0/16, 239.254.0.0/16, 239.255.0.0/16.
    - **Organization-Local Scope** – 239.192.0.0 to 239.251.255.255
- The IEEE 802.3 LAN specification defines that the bits 0 and 1 (most significant bits) of the 1st octet indicate a broadcast or multicast frame and a locally administrated frame respectively. A universally administered address is uniquely assigned to a device by its manufacturer – burned-in address (BIA). A locally administered address is assigned to a device by an admin.

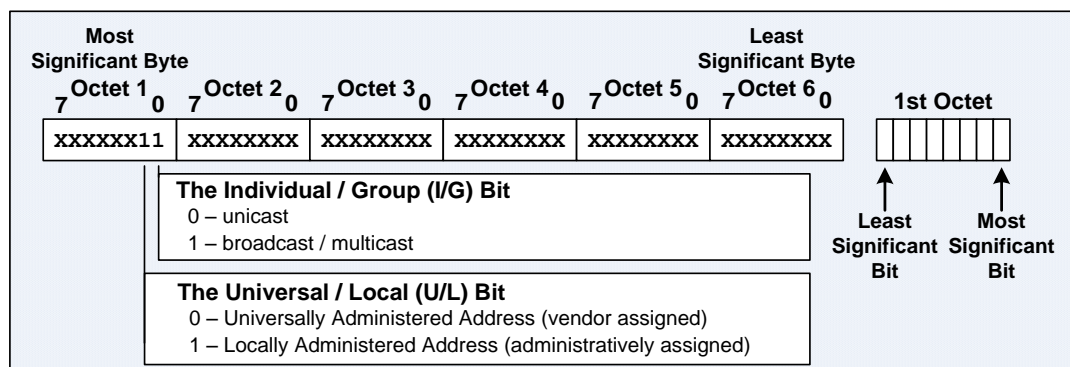
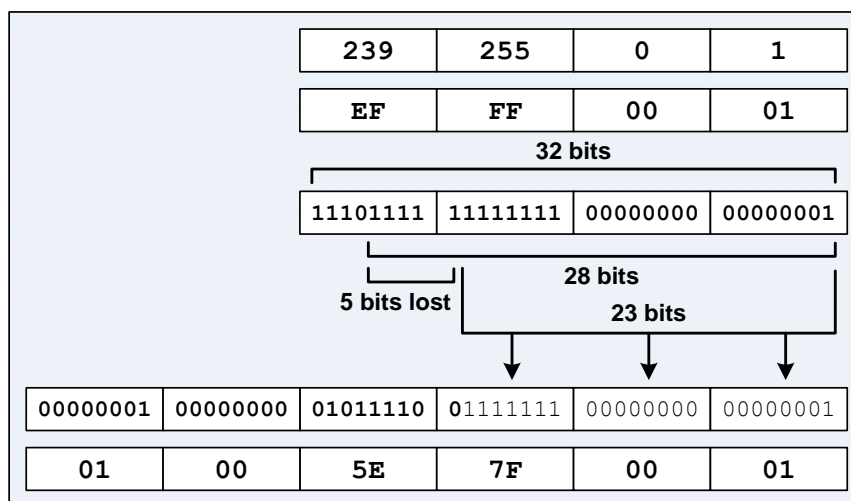


Figure 19-2: IEEE 802.3 MAC Address Format

- Take note upon the location of the most significant byte and most significant bit in each byte. IEEE documentation specifies Ethernet MAC addresses with the most significant byte on the left; inside each byte, the leftmost and rightmost bits are the least and most significant bits respectively. This bit order is referred to as **canonical** or **little-endian**.  
**Note:** When overriding the MAC address to use a local address, the device or device driver often does not enforce the setting of the U/L bit to a value of 1.
- The IANA owns a block of Ethernet MAC addresses that start with hexadecimal 01:00:5E. The lower half of this block (0100.5E00.0000 – 0100.5E7F.FFFF) is allocated for multicast MAC addresses. The translation between L3 IP multicast and L2 MAC multicast addresses is achieved by mapping the low-order 23 bits of the L3 IP multicast address into the low-order 23 bits of the L2 MAC address.



**Figure 19-3: IP Multicast to Ethernet MAC Multicast Address Mapping**

- As there are 28 bits (32 – 4 Class D prefix) of unique address space for an IP multicast address and only 23 bits are mapped into the IEEE MAC address, there are 5 bits of overlap (28 – 23). These 5 bits represent  $2^5 = 32$  addresses. Therefore, there is a 32:1 overlap of IP addresses to MAC addresses – 32 IP multicast addresses are mapped to the same MAC multicast address. As an example, below lists all the IP multicast addresses that are mapped to the same MAC multicast address 01-00-5E-7F-00-01.

224.127.0.1	225.127.0.1	226.127.0.1	227.127.0.1	228.127.0.1
229.127.0.1	230.127.0.1	231.127.0.1	232.127.0.1	233.127.0.1
234.127.0.1	235.127.0.1	236.127.0.1	237.127.0.1	238.127.0.1
239.127.0.1	224.255.0.1	225.255.0.1	226.255.0.1	227.255.0.1
228.255.0.1	229.255.0.1	230.255.0.1	231.255.0.1	232.255.0.1
233.255.0.1	234.255.0.1	235.255.0.1	236.255.0.1	237.255.0.1
238.255.0.1	239.255.0.1			

- Since one multicast MAC address can actually represent 32 multicast IP addresses, a host must receive and examine every frame destined for the multicast group that it has joined, regardless of the destination multicast IP address that the frame is destined for; followed by examine the IP address in the frame to verify the multicast group that the frame is destined for.
- **Note:** Switches forward frames based on MAC addresses. If a switch is configured for Layer 2 multicast snooping, it will forward frames to all members belong to other multicast groups with the same MAC address mapping, even if the frames belong to a different multicast group. Most Layer 2 switches flood all multicast frames that falls within the MAC address range of 0100.5E00.00xx out to all switch ports even if IGMP snooping is enabled.

- Whenever a multicast application is started on a receiver, the application must learn about the available multicast sessions or streams, which are mapped to one or more IP multicast group. The application may then request to join the corresponding multicast groups.
- Below lists the available methods for multicast applications to learn about the multicast sessions:
  - The application may join a well-known, predefined group in which another application sends announcements about the available sessions.
  - The application may contact the appropriate directory servers if they are available.
  - The application may be launched from a web page on which the sessions are listed as URLs.
  - The application may join a session upon clicking the email link that announced the session.
- Another option is the use of the application which called **Session Discovery (sd)** that acts like a TV program guide to display the multicast contents. A client directory application uses either the **Session Description Protocol (SDP)** or **Session Announcement Protocol (SAP)** to learn about the available contents. **Note:** SDP/SAP is referred to as **sd**r in Cisco documentation.
- The initial **sd** tool was revised and become the **sd**r session directory application that provides functions that announce available sessions along with their descriptions, and create new sessions. When SDR is used at the sender side, it creates new sessions and avoids address conflicts. Senders consult their SDR caches upon session creation to choose an unused multicast address. When a new session is created, the sender announces it with all the information that is necessary for receivers to join the session. When SDR is used at the receiver side, it learns about the available sessions. A join to a particular multicast group is initiated when the user selects a multicast channel listed in SDR.
- RFC 4566 – Session Description Protocol (SDP) defines the variables that describe the sessions. Most of the variables were inherited from the **sd**r tool. The RFC does not define the transport of the SDP packets. Below lists the mechanisms to transmit the packets that describe the sessions:
  - RFC 2974 – Session Announcement Protocol (SAP) that broadcasts multicast session info.
  - RFC 3261 – Session Initiation Protocol (SIP) that defines a signaling protocol for Internet conferencing, telephony, telepresence, event notification, instant messaging, etc.
  - RFC 2326 – Real Time Streaming Protocol (RTSP) that serves mainly as a control protocol in multimedia environments. RTSP allows videocassette recorder (VCR)-like controls, eg: select, forward, rewind, pause, stop; and also carries the info of a session.
  - Multipurpose Internet Mail Exchange (MIME)-format emails that may carry SDR packets describing multicast sessions.
  - Web pages that may provide session descriptions in standardized SDR format.
- Cisco IP/TV is an example of IP multicast applications. Cisco IP/TV generally has 3 components – the server (the source), the content manager (the directory server), and the viewer (the receiver). Viewers may contact the content manager directly via unicast and request the list of available programs (sessions or streams) from it. Viewers may also listen to periodic SAP announcements. Cisco IP/TV uses SAP to transport the SDR sessions to the viewers using standard SDR formation.
- The term **Internet Protocol Television (IPTV)** first appeared in 1995 with the founding of Precept Software, which was acquired by Cisco Systems in 1998. The IP/TV technology was then integrated into various Cisco product lines, eg: Cisco IP/TV Software, Cisco Application and Content Networking System (ACNS) Software, Cisco Content Delivery Engine (CDE), Cisco Wide Area Application Engine (WAE), etc.



## Internet Group Management Protocol (IGMP)

- Hosts use **Internet Group Management Protocol (IGMP)** to register with their local routers to join and leave specific multicast groups. The routers are then aware that they need to forward the multicast data stream destined to the registered hosts of a specific multicast group. Understanding the IGMP multicast group membership join and leave process is important for configuring and troubleshooting IP multicasting.
- RFC 1112 – Host Extensions for IP Multicasting specifies IGMPv1 specifies that multicast routers periodically send Membership Queries (every 60 to 120 seconds) to the All-Hosts multicast address group 224.0.0.1. Hosts that are interested to receive traffic for specific multicast group send Membership Reports to the multicast address of the group they want to join. Hosts send Membership Reports either when they want to first join a multicast group or in response to Membership Queries originated by multicast routers (known as IGMPv1 queriers) to indicate that they want to continue receive traffic destined for the group that they have joined.  
**Note:** The source does not necessarily have to join a multicast group; it only needs to know the multicast address to send traffic to the receivers on the group.
- The **report suppression** process is in which only one member per group responds to a query on each subnet in order to conserve bandwidth on the subnet and minimize processing by hosts.
- IGMPv1 specifies that there must be at least an active member of a multicast group on a segment if multicast traffic is to be forwarded to that segment.
- IGMPv1 does not define a mechanism for hosts to leave a multicast group; therefore IGMPv1 hosts leave a multicast group silently anytime without notifying the local router. It is OK when there are multiple members remain on a segment, as the multicast traffic must still be delivered to the segment. However, when the last member on a segment leaves the multicast group, the router would still forward multicast traffic into a segment unnecessarily for a period of time (3 minutes) until it timeout the group after 3 consecutive queries without a report from a host. This approach is inefficient when there are many groups or there is a lot of traffic in the groups.
- RFC 2236 – Internet Group Management Protocol, Version 2 was introduced to overcome the limitations and restrictions discovered in IGMPv1. IGMPv2 is backward compatible with IGMPv1. IGMPv2 focus upon the issues of leave and join latencies as well as the ambiguities of IGMPv1.
- Below summarizes the important changes in IGMPv2:
  - **Group-Specific Query** that allows a router to query membership for only a single group instead of all groups; an optimized method to quickly find out whether any members are left in a particular group without asking all groups to report. The difference between the General Query and Group-Specific Query is that a General Query is multicast to the All-Host address 224.0.0.1; whereas a Group-Specific Query for a specific group is multicast to the multicast address of the group.
  - **Leave Group message** that allows hosts to inform their local routers when leaving a group. This reduces the leave latency for the group when the last member is leaving the group. The specification mentions the timing of when Leave Group message must be sent.
  - **Querier Election mechanism** that specifies that when there are multiple IGMP routers on the same segment (broadcast domain), the router with the highest IP address is elected as the designated querier.
  - **Query intervals and response times** that are specified in a general or group-specific query and are used to control the burstiness of membership reports. The query indicates the response time that the members must respond to a query by issuing a report.

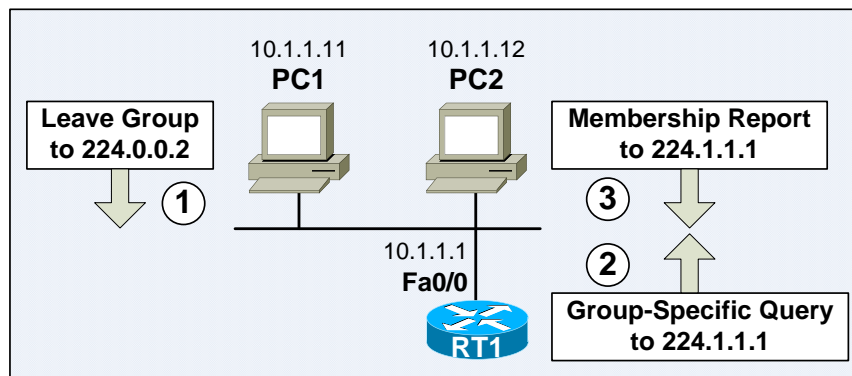
- IGMPv2 members do not need to wait for a query to join a multicast group; they send an unsolicited Membership Report to indicate the interest to join a group. This process reduces the join latency when there is no other members exist in the group.
- The output of the **show ip igmp groups** EXEC command below indicates that the multicast group 224.1.1.1 has been active on the Fa0/0 interface for 49 seconds, the multicast group expires and will be deleted in 2 minutes and 51 seconds if an IGMP Host Membership Report for the group is not received within that period, and the last host that reported its membership was PC2 (10.1.1.12) when it joined the multicast group.

```

RT1#sh ip igmp groups
IGMP Connected Group Membership
Group Address      Interface          Uptime    Expires    Last Reporter
224.1.1.1         FastEthernet0/0   00:00:51  00:02:41   10.1.1.12
224.0.1.40        FastEthernet0/0   00:01:20  00:02:10   10.1.1.1
RT1#

```

- IGMPv1 hosts leave group passively and silently; they just stop reporting their membership upon the membership queries. IGMPv2 introduces explicit Leave Group messages. When an IGMPv2 router receives a Leave Group message, it sends a Group-Specific Query for the multicast group to find out whether there are other hosts that are still interested to receive traffic for the group. This process reduces the leave latency.



**Figure 19-4: IGMPv2 – Leaving a Multicast Group**

- PC1 and PC2 are members of the multicast group 224.1.1.1. Host PC1 leaves the group and by sending an IGMPv2 Leave Group message to the All-Routers multicast address 224.0.0.2. RT1 receives the Leave Group message and sends a Group-Specific Query to find out whether there are other members are still present in the group. PC2 that has not left the multicast group 224.1.1.1 yet responds with a Membership Report message to inform RT1 that it is still interested to receive traffic destined for 224.1.1.1 and request RT1 to forward traffic for the group. Below shows the output of the **show ip igmp groups** EXEC command after PC1 has left the group.

```

RT1#sh ip igmp groups
IGMP Connected Group Membership
Group Address      Interface          Uptime    Expires    Last Reporter
224.1.1.1         FastEthernet0/0   00:01:41  00:02:20   10.1.1.12
224.0.1.40        FastEthernet0/0   00:02:09  00:02:23   10.1.1.1
RT1#

```

- When PC2 leaves the multicast group by sending an IGMPv2 Leave Group message, RT1 sends a Group-Specific Query and expect a report from other active group members. However, since PC2 is the last member of the multicast group 224.1.1.1, RT1 will not receive any Membership Report for 224.1.1.1, and the group time out and RT1 stops forwarding multicast traffic for the multicast group into the segment (broadcast domain).
- RFC 3376 – Internet Group Management Protocol, Version 3 introduced the support for multicast source filtering, in which hosts can determine to receive multicast traffic only from particular sources within a multicast group, results in efficient utilization of routing resources.
- A host sends an IGMPv3 Membership Report that may specify a source list that is used for source filtering to 224.0.0.22 (the IGMP multicast address) upon joining a multicast group. A source list is used to indicate the multicast sources that the host will accept (INCLUDE) or will not accept (EXCLUDE) packets from. Multicast routers can now avoid delivering multicast packets from specific sources to networks where there are no interested receivers.
- IGMPv3 multicast routers refresh the group membership state of members on a segment by sending General Queries periodically to request group membership information; and all IGMPv3 members respond with IGMPv3 Membership Reports that contain Current-State Group Records.
- IGMPv3 is mainly used for **Source-Specific Multicast (SSM)** that is based on PIM sparse mode, which uses a separate source-distribution tree for each source within each multicast group.
- The **show ip igmp interface [type num]** EXEC command displays multicast-related information for an interface, eg: the IGMP version that is active on an interface.

```

RT1#sh ip igmp interface
FastEthernet0/0 is up, line protocol is up
  Internet address is 10.1.1.1/24
  IGMP is enabled on interface
  Current IGMP host version is 2
  Current IGMP router version is 2
  IGMP query interval is 60 seconds
  IGMP querier timeout is 120 seconds
  IGMP max query response time is 10 seconds
  Last member query count is 2
  Last member query response interval is 1000 ms
  Inbound IGMP access group is not set
  IGMP activity: 2 joins, 0 leaves
  Multicast routing is enabled on interface
  Multicast TTL threshold is 0
  Multicast designated router (DR) is 10.1.1.1 (this system)
  IGMP querying router is 10.1.1.1 (this system)
  Multicast groups joined by this system (number of users):
    224.0.1.40(1)
RT1#

```

## IGMP Snooping and Cisco Group Management Protocol (CGMP)

- Usually hosts are not directly connected to routers but are connected to switches instead. IGMP is a L3 network layer protocol, and therefore switches that operate at the L2 data link layer do not understand nor participate in IGMP and hence are not aware of which directly connected hosts are members of which particular multicast groups. L2 switches flood multicast frames out to all ports within a VLAN by default (except the port from which the frame is originated); the same way that unknown unicast frames are flooded – even if only a host on a port required the multicast data stream.
- A Cisco Catalyst switch can be configured to manually associate a multicast MAC address with multiple ports in which it only forwards multicast frames destined for a particular multicast group. However, this method is not scalable as IP multicast hosts dynamically join and leave groups.
- **IGMP snooping** allows a switch to eavesdrop the IGMP Membership Reports that are originated from hosts to the local multicast routers when they are joining multicast groups, and update its MAC address table or Content Addressable Memory (CAM) table accordingly. The switch must understand IGMP to process IGMP Membership Reports and IGMP Leave Group messages. Catalyst switches that support IGMP snooping always have specific built-in ASIC hardware that enhances the processing of IGMP messages, which in turn directly affect the overall cost.
- **Note:** IGMP snooping is not supported on all Cisco Catalyst switch platforms. Google about **Multicast Catalyst Switches Support Matrix** for the support of IGMP Snooping and CGMP.
- CGMP was developed for Cisco Catalyst switch platforms that do not support IGMP snooping. CGMP is a Cisco-proprietary protocol that operates between a multicast router and a switch. A multicast router informs all its directly connected switches regarding IGMP registrations that were received from hosts through the switches, which in turn forwards multicast frames only to ports on which the requesting hosts are connected rather than flooding the frames out to all ports. **Note:** A multilayer switch configured for multicast routing also can be configured for CGMP.
- CGMP messages are sent to the well-known address 0100.0cdd.dddd that is flooded everywhere as a special case so that CGMP messages can be propagated across non-CGMP switches. A CGMP that is originated from a multicast router contains the request type (join or leave), the L2 multicast MAC address, and the actual MAC address of the host.

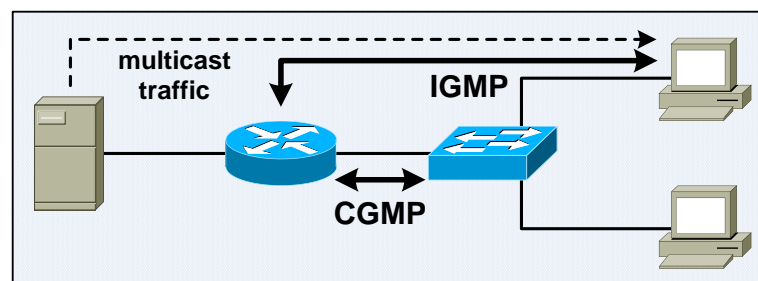


Figure 19-5: IGMP and CGMP

- IGMP is used between a host and its local router; while CGMP is used between a router and a switch. CGMP is disabled on all interfaces on multicast routers by default. The **ip cgmp** interface subcommand enable CGMP on an interface. Only the multicast router must be configured for CGMP. By default, all Catalyst switches and automatically process CGMP messages from routers.

- IGMP snooping and CGMP are mutually exclusive – cannot be used at the same time on a switch. IGMP snooping is enabled by default for switches that have IGMP snooping capability; while CGMP is enabled by default for switches that do not support IGMP snooping. Always remember to enable CGMP on upstream multicast router or multilayer switches for networks that have legacy L2 switches for efficient flooding of multicast traffic.

## Protocol Independent Multicast (PIM)

- A router performs routing table lookup upon the destination address and forwards a unicast packet out the appropriate interface. However, a router may have to forward a multicast packet out to multiple interfaces. PIM is commonly implemented on multicast routers to dynamically build the distribution trees that determine the paths to deliver the multicast traffic to all receivers.
- Multicast routers consider both the source and destination addresses of a multicast packet and use the distribution tree to forward the packet away from the source toward the destination. Below describes the 2 types of distribution trees:
  - **Source Tree** – created for each source that is sending traffic to each multicast group. It has its root at the source and has branches throughout the network to the receivers. Source trees are also known as source-routed or shortest path trees (SPTs) as the tree takes a direct and shortest path from the source to its receivers.
  - **Shared Tree** – a single tree that is shared between all sources for each multicast group. A shared tree has a single common root known as the **Rendezvous Point (RP)**. Sources initially send their multicast packets to the RP, which in turn forwards data through a shared tree to the members of the group.

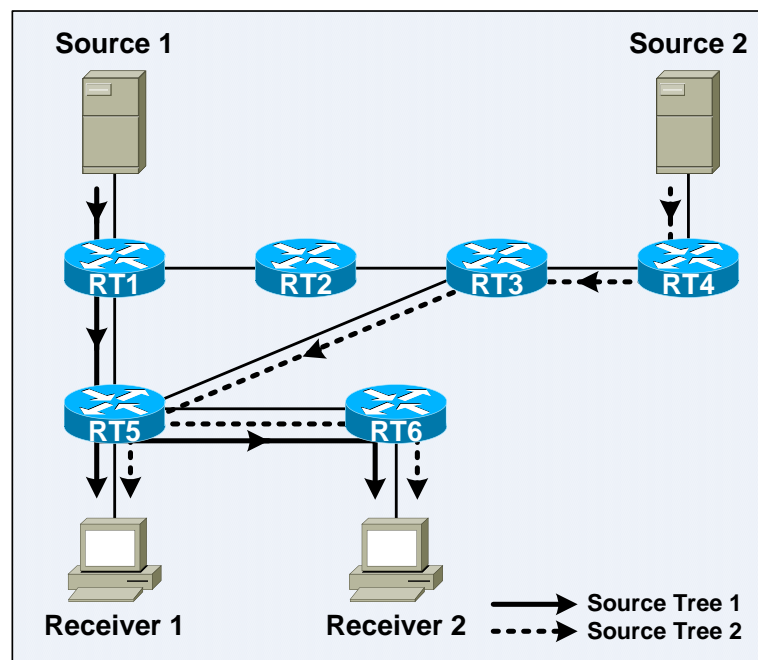


Figure 19-6: Source Distribution Tree

- The figure above shows 2 source trees between Source 1, Receiver 1, and Receiver 2; as well as between Source 2, Receiver 1, and Receiver 2. The path between the source and receivers is the path with the lowest cost. Packets are forwarded according to the source and group address pair along the tree. The forwarding state associated with the source tree is identified using the notation (S, G) (pronounced as “S comma G”), where S is the IP address of the source and G is the multicast group address. A separate unique tree is built for every source S sending to group G.

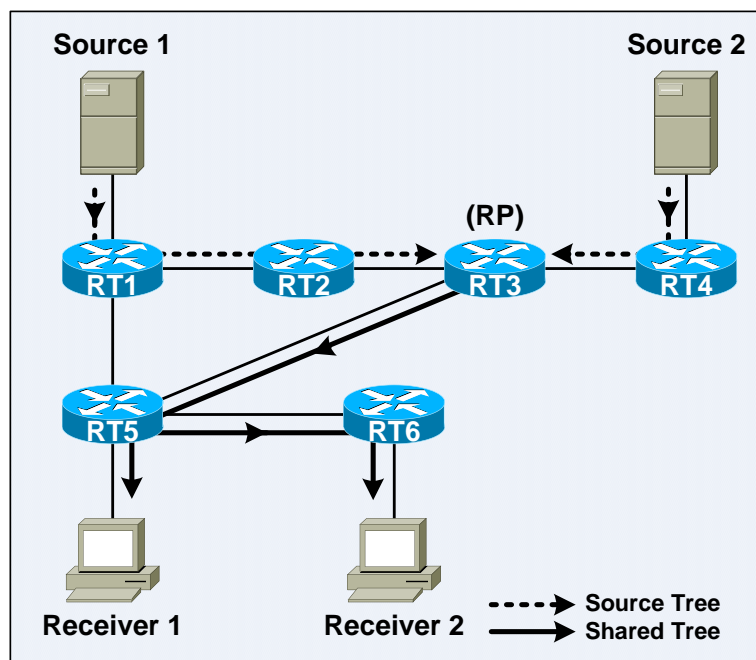


Figure 19-7: Shared Distribution Tree

- The figure above shows a shared distribution tree. RT3, the RP, is the root of the shared tree. The tree is built from RT3 to RT5 and RT6 toward Receiver 1 and Receiver 2. Source 1 and Source 2 send multicast packets toward the RP via source distribution trees; the packets are then forwarded from the RP to the receivers according to the shared distribution tree. The default forwarding state for the shared tree is identified using the notation (\*, G) (pronounced as “star comma G”); where \* is a wildcard entry meaning any source, and G is the multicast group address. The root is not necessarily the multicast source – it is a router that is centrally located in the network that is called the **Rendezvous Point (RP)**.
- Think that the multicast forwarding paths as a tree structure. The source resides at the root of the tree and blindly sending IP packets to a multicast address. The source is never aware of the recipients that are members of a multicast group. The source depends upon multicast routers and switches to deliver the multicast packets. The multicast routers or switches reside at a branch of the tree replicate the multicast packets upon interfaces that have downstream recipients.
- **Reverse Path Forwarding (RPF)** is the concept of forwarding multicast traffic away from the source, rather than toward the receiver – opposite of the normal unicast packet forwarding. RPF ensures that multicast packets are not being replicated back into the network in order to avoid routing loops. In multicasting, the source IP address indicates the known source, while the destination IP address indicates a group of unknown receivers. “toward the destination” and “away from the source” sound like the same thing, but they are not!
- Multicast routers uses the unicast routing table to determine the upstream (toward the source) and downstream (away from the source) neighbors and ensures that only one router interface is considered as the incoming interface for a specific multicast source – verify that the packet is received upon the same interface used to reach the source. If this is true, the packet can be forwarded or replicated toward the multicast recipient; if it is not true, the packet is discarded. Packet received on an interface and forwarded out another interface might be replicated around the network and forwarded back to the same router on a different interface. RPF ensures that this packet is not being forwarded again.

- **PIM Dense Mode (PIM-DM)** is defined in RFC 3973 – Protocol Independent Multicast – Dense Mode (PIM-DM): Protocol Specification (Revised). PIM-DM uses a **push** approach that uses source trees to flood multicast traffic to the entire network. Routers that do not need the data (because they are not connected to receivers that want the data or to other routers that want it) request to prune the tree so that they do not receive the multicast packets.

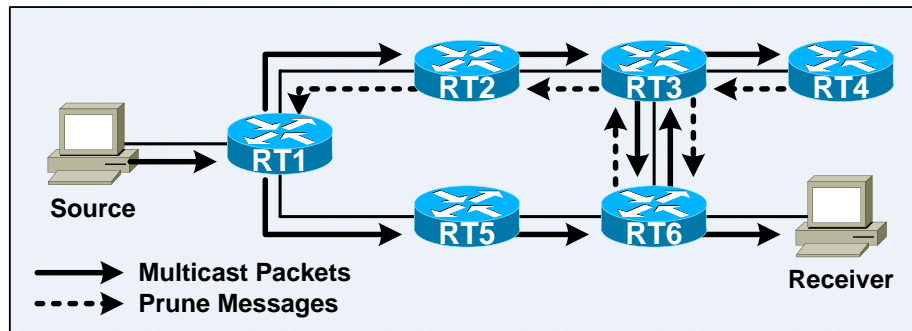


Figure 19-8: PIM-DM Initial Flooding and Pruning

- PIM-DM initially floods multicast traffic throughout the entire network. The traffic is sent out of all non-RPF interfaces where there is another PIM-DM neighbor or a directly connected member. As each router receives the multicast traffic via its RPF interface (the interface in the direction of the source), it forwards the multicast traffic to all its PIM-DM neighbors. The (S, G) state entry is created in every router in the network.
- The flooding may result in some traffic arriving upon a non-RPF interface as with RT3 and RT6. Packets arriving via the non-RPF interfaces are discarded. PIM-DM Prune messages are sent to stop unwanted traffic when there is no host registered for the multicast group using IGMP. Prune messages are sent out of an RPF interface when the router has no downstream receivers for multicast traffic from the specific source. Prune messages are also sent out of non-RPF interfaces to terminate the multicast traffic flow as it is arriving via an interface that is not on the shortest path to the source. PIM-DM Prune messages are sent to 224.0.0.13 – PIM.
- There is only one receiver in the scenario above, and therefore all other paths are pruned. Although the multicast traffic flow does not reach and pass through most routers in the network, the (S, G) state entry remains in all routers and will remain there until the source stops sending. In PIM-DM, all prune messages expire in 3 minutes. After that, the multicast traffic is flooded again to all routers. This periodic flood-then-prune operation or behavior is normal and must be taken into account when a network is intended to use PIM-DM.
- PIM-DM routers assume that the recipients of a multicast group are located on every subnet – the multicast group is densely populated across the network; few senders, but many receivers; there will be a great amount of multicast traffic; and the multicast streams will be constant.

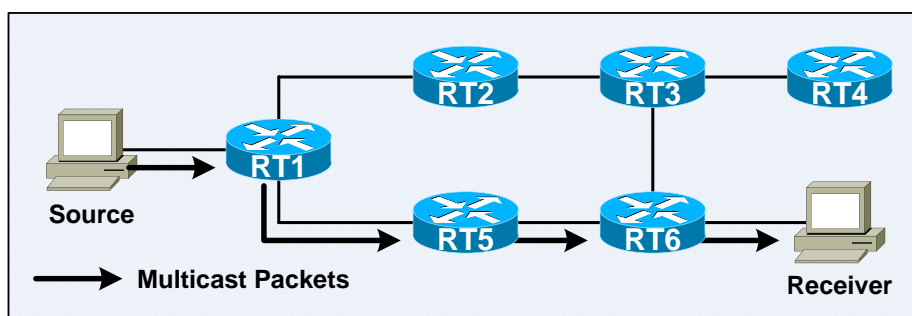


Figure 19-9: PIM-DM Multicast Traffic Flow after Pruning



- **PIM Sparse Mode (PIM-SM)** is defined in RFC 2362 – Protocol Independent Multicast – Sparse Mode (PIM-SM): Protocol Specification. PIM-SM uses a different **pull** approach to forward multicast traffic only to the portions of network that need it. It uses a shared tree and therefore required to define an RP. In sparse mode, sources register with the RP, multicast routers along the path from active receivers that have explicitly requested to join a specific multicast group would join the tree – the multicast tree is not extended to a router unless a host has joined the group. The multicast tree is built and grown in reverse by beginning with the group members at the end leaves and extended back toward the central root. Multicast routers calculate using the unicast routing table whether they have a better metric to the RP or to the source itself. They forward the join messages to the device with which they have the better metric.
- Sparse mode multicast flows are described as (\*, G) as the multicast tree allows any source to send to a group. As a receiver joins a multicast group via IGMP, the local router forwards the membership report toward the RP at the root of the tree. Each router along the way adds that branch to the shared tree. Pruning is performed only when a member leaves the group.

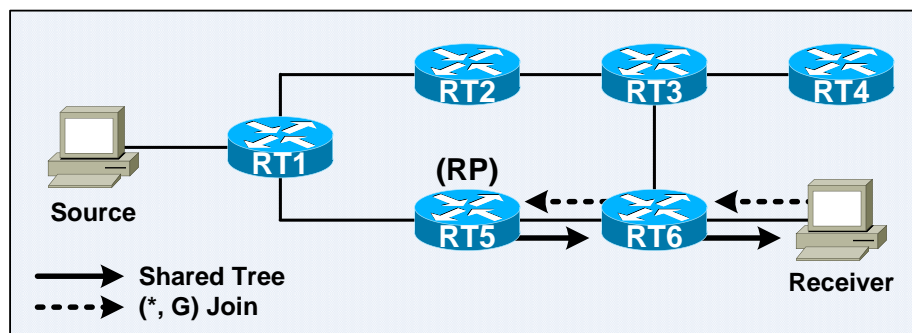


Figure 19-10: PIM-SM Shared Tree Join

- When an receiver attached to the leaf router RT6 joins the multicast group G, the last-hop router – RT6 which knows the IP address of the RP router for multicast group G sends a (\*, G) join for the group toward the RP. The PIM Join travels hop-by-hop toward the RP to build a branch of the shared tree that extends from the RP to the last-hop router directly connected to the receiver. The traffic of multicast group G may then flow down the shared tree to the receiver. The (\*, G) state entry is created only along the shared tree.
- Both PIM-DM and PIM-SM modes construct identical tree structures and therefore result in the same multicast traffic flow patterns. PIM-SM is appropriate for wide-scale deployment for both densely and sparsely populated groups in an enterprise network. PIM-SM is preferred over PIM-DM for all production networks regardless of size and membership density.
- **PIM Sparse-Dense Mode** allows a PIM router to operate in both sparse and dense modes on a per-group basis on the same router interface. Sparse mode is used if a group has an RP defined; otherwise, dense mode is used. PIM sparse-dense mode also supports automatic RP discovery. Multiple RPs can be implemented with each RP in an optimum location for maximum efficiency. Configuring, managing, and troubleshooting multiple RPs can be difficult if done manually. However, PIM sparse-dense mode supports automatic selection of RPs for each multicast source, eg: RT1 could be the RP for Source 1 and RT2 could be the RP for Source 2. If no RP is discovered for the multicast group or none is manually configured, PIM sparse-dense mode will operate in dense mode. Therefore, automatic RP discovery should be implemented with PIM sparse-dense mode.
- Cisco recommends PIM sparse-dense mode for IP multicast, as PIM-DM does not scale well and requires many router resources, and PIM-SM has limited RP configuration options. Additionally, it can use either statically defined RPs, Auto-RP, or BSR with the least configuration effort.



- Below are some extensions, optimizations, and enhancements upon PIM:
  - **Bidirectional PIM Mode**, which is designed for many-to-many applications – many hosts multicasting to each other.
  - **Source-Specific Multicast (SSM)**, which is a variant of PIM-SM that builds only source specific shortest path trees and does not need an active RP for source-specific groups in the address range 232.0.0.0/8.

- Comparison of PIM modes:

	<b>Multicast Flows</b>	<b>Tree Construction</b>	<b>Tree Refinements</b>
<b>Dense Mode</b>	(S, G)	Root to leaves. Source is the root. Receivers are the leaf nodes.	First flood, then prune.
<b>Sparse Mode</b>	(*, G)	Leaves to root. RP is the root. Source can be anywhere. Receivers are the leaf nodes.	Group extended from receivers toward RP. Pruning only when member leaves group.
<b>Sparse-Dense Mode</b>	(S, G) or (*, G)	Hybrid on a per-group basis	N/A

- The **ip pim dense-mode** interface subcommand configures PIM dense mode on an interface. The **ip pim sparse-mode** interface subcommand configures PIM sparse mode on an interface. The **ip pim sparse-dense-mode** interface subcommand configures PIM sparse-dense mode on an interface.
- PIMv1 RPs can be configured manually or using the dynamic auto-RP process. The **ip pim rp-address {ip-addr} [access-list] [override]** global configuration command manually identify an RP. An access list limits the range of multicast groups supported by the RP. The **override** keyword causes the RP to be preferred over any automatically determined RP. Because the RP does not advertise itself, its address and function must be defined on every router in the PIM domain, including the RP itself. Future changes upon the RP location are difficult as every router must be reconfigured with the new RP address.
- **Auto-RP** is a Cisco-proprietary process that automatically informs PIM-SM routers about the appropriate RP for a group by identifying a centrally located and well-connected router to function as the **mapping agent** that learns all the candidate RPs that are announced through the Cisco-RP-Announce multicast address 224.0.1.39, in which all PIM-SM routers must join by default.
- The **ip pim send-rp-discovery [intf-type intf-num] scope {ttl}** global configuration command configures a router as a RP mapping agent. The optional *intf-type intf-num* defines the interface type and number that is to be used as the source address of the RP mapping agent; and the optional *ttl* parameter specifies the Time-to-Live (TTL) value that limits the scope of the Auto-RP discovery messages – how many router hops away the information will reach and valid. The RP mapping agent sends Group-to-RP mapping information to all PIM routers over the Cisco-RP-Discovery multicast address 224.0.1.40.
- Each candidate RP router must then be explicitly defined with the **ip pim send-rp-announce {intf-type intf-num | ip-addr} scope {ttl} [group-list acl]** global configuration command. A router begins sending announcements to the RP mapping agent when it knows it can be an RP. The interface must be specified to indicate the advertised RP address and identifies where to reach the mapping agent. TTL limits the scope of Auto-RP announcements by the number of router hops. The router can also advertise itself as a candidate RP for the multicast groups permitted through the optional **group-list** access list. The default announcement interval is 60 seconds.

- PIMv2 also includes an industry-standard dynamic Group-to-RP mapping advertisement mechanism that is known as **Bootstrapping**, which is similar to the Cisco Auto-RP method.
- A **bootstrap router** (BSR) that learns about RP candidates for a group and advertises them to PIM routers must first be identified using the **ip pim bsr-candidate** *{intf-type intf-num<sup>[1]</sup> [hash-mask-length] [priority]}* global configuration command; followed by defining the candidate RP routers that advertise themselves to the BSR as PIMv2 candidate RPs using the **ip pim rp-candidate** *{intf-type intf-num<sup>[2]</sup> [ttl] [group-list acl] [priority priority]}* global configuration command. The priority value ranges from 0 to 255. The BSR or RP with the larger priority is preferred. The router with the higher IP address becomes the BSR or RP if the priority values are the same.
 

**Note:** The Cisco IOS implementation of PIM BSR which predates the draft-ietf-pim-sm-bsr IETF draft uses the value 0 as the default priority for candidate RPs and BSRs. Explicitly set the priority value to 192 to comply with the IETF draft that specifies 192 as the default priority value.

[1] – The IP address associated with this interface determines the candidate BSR address.  
 [2] – Advertises the IP address associated with this interface as the candidate RP address.
- Once the BSR and candidate RPs are configured, all other PIM routers will learn the appropriate RP from the BSR. The selection of RP for a group is based on a hashing function. The length of the hash mask controls the number of multicast groups that are being hashed to the same RP.
- The bootstrap messages are propagated throughout the entire PIM domain by default. The scope of the advertisements can be limited by defining PIMv2 border routers using the **ip pim border** global configuration command.
- A small network with only one or some L2 or L3 switches and without a multicast router always support multicast. When a host sends an IGMP Membership Report to join a multicast group, it does not know about multicast routers at all; it just sends out a request to join and hopes that it will start receiving traffic destined for the multicast group. Even if a multicast router is present, it does not send a reply to a host upon joining a multicast group. A multicast router only sends out Membership Queries periodically asking if the hosts still want to remain as a member of a group. In such small network, L2 switches simply flood the multicast traffic out all ports on a VLAN; CGMP is not in action to prune the multicast traffic. L3 switches can use IGMP snooping to constrain the flooding of multicast traffic.

*This page is intentionally left blank*

## IP Multicast Routing Lab

- IP multicast routing must first be enabled using the **ip multicast-routing** global configuration command regardless of the multicast routing protocol implemented.

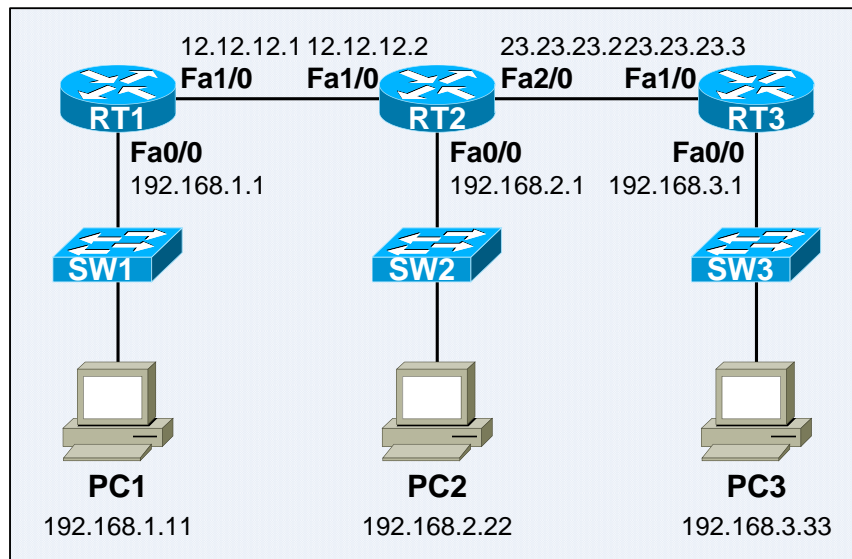


Figure 20-1: Sample IP Multicast Network

### PIM Dense Mode Configuration

RT1 Configuration:	RT2 Configuration:	RT3 Configuration:
<pre>! ip multicast-routing ! interface FastEthernet0/0  ip pim dense-mode ! interface FastEthernet1/0  ip pim dense-mode ! router eigrp 100  network 12.0.0.0  network 192.168.1.0  no auto-summary !</pre>	<pre>! ip multicast-routing ! interface FastEthernet0/0  ip pim dense-mode ! interface FastEthernet1/0  ip pim dense-mode ! interface FastEthernet1/0  ip pim dense-mode ! router eigrp 100  network 12.0.0.0  network 23.0.0.0  network 192.168.2.0  no auto-summary !</pre>	<pre>! ip multicast-routing ! interface FastEthernet0/0  ip pim dense-mode ! interface FastEthernet1/0  ip pim dense-mode ! router eigrp 100  network 23.0.0.0  network 192.168.3.0  no auto-summary !</pre>

Figure 20-2: PIM-DM Configuration on RT1, RT2, and RT3

- The following commands are often used in lab environments where no multicast servers and receivers are configured. These can cause multicast traffic to flow to a network segment.
  - Issue the **ip igmp join-group {group-address}** interface subcommand to configure a router to join and become a member of the specified group. Note that accepting multicast packets prevents the router from performing fast switching. Once the router become a multicast group member and supports the protocol that is being transmitted to the group, eg: ICMP Echo Request packets, it can respond to the requests addressed to the group.
  - Issue the **ip igmp static-group {group-address}** interface subcommand to configure a router as a statically connected member of the specified group. The router does not accept the packets destined for the group, but only forwards them; hence, allows fast switching. The outgoing interface appears in the IGMP cache, but the router itself is not a member, as evidenced by the lack of an L (local) flag in the multicast route entry, as shown in the **show ip mroute EXEC** command.

- Below shows the status of RT1 upon PC1 joining the multicast group 224.11.22.33:

```

RT1#sh ip igmp groups
IGMP Connected Group Membership
Group Address      Interface          Uptime    Expires    Last Reporter
224.0.1.40         FastEthernet0/0   00:07:16  00:02:53  192.168.1.1
RT1#
=====

PC1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
PC1(config)#int fa0/0
PC1(config-if)#ip igmp join-group 224.11.22.33
PC1(config-if)#do sh ip igmp groups
IGMP Connected Group Membership
Group Address      Interface          Uptime    Expires    Last Reporter
224.11.22.33       FastEthernet0/0   00:00:30  stopped    192.168.1.11
PC1(config-if)#
=====

RT1#
RT1#sh ip igmp groups
IGMP Connected Group Membership
Group Address      Interface          Uptime    Expires    Last Reporter
224.11.22.33       FastEthernet0/0   00:00:09  00:02:50  192.168.1.11
224.0.1.40         FastEthernet0/0   00:07:37  00:02:32  192.168.1.1
RT1#
RT1#sh ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
      L - Local, P - Pruned, R - RP-bit set, F - Register flag,
      T - SPT-bit set, J - Join SPT, M - MSDP created entry,
      X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
      U - URD, I - Received Source Specific Host Report,
      Z - Multicast Tunnel, z - MDT-data group sender,
      Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 224.11.22.33), 00:00:11/00:02:48, RP 0.0.0.0, flags: DC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet1/0, Forward/Dense, 00:00:11/00:00:00
    FastEthernet0/0, Forward/Dense, 00:00:11/00:00:00

(*, 224.0.1.40), 00:07:09/00:02:29, RP 0.0.0.0, flags: DCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet1/0, Forward/Dense, 00:06:03/00:00:00
    FastEthernet0/0, Forward/Dense, 00:07:09/00:00:00

RT1#

```

- Below shows that all members of the multicast group will reply the ICMP Echo Requests destined for the multicast group.

```

PC1#ping 224.11.22.33

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 224.11.22.33, timeout is 2 seconds:

Reply to request 0 from 192.168.1.11, 4 ms
Reply to request 0 from 192.168.2.22, 128 ms
Reply to request 0 from 192.168.3.33, 128 ms
PC1#

```

- Enabling PIM on an interface also enables IGMPv2 operation on the interface. The **ip igmp version {1 | 2 | 3}** interface subcommand changes the IGMP version for a particular interface.
- There are 2 versions of the PIM protocol – PIMv1 and PIMv2. PIMv2 is used on router interface by default. The **ip pim version {1 | 2}** interface subcommand changes the version.
- The **show ip mroute [group-addr] [summary] [count] [active [kbps]] EXEC** command that displays the IP multicast routing table is the most useful command for determining the state of multicast sources and groups from the router perspective. It represents a part of the multicast distribution tree with an incoming interface and a list of outgoing interfaces. The **summary** keyword displays a one-line, abbreviated summary of each entry in the IP multicast routing table. The **count** keyword displays statistics about the group and source, including number of packets, packets per second, average packet size, and bytes per second. The **active** parameter displays the rate at which active sources are sending to multicast groups; active sources are those sending at the rate specified as the *kbps* value or higher (the default value is 4 kbps).

```

RT1#sh ip mroute summary
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
      L - Local, P - Pruned, R - RP-bit set, F - Register flag,
      T - SPT-bit set, J - Join SPT, M - MSDP created entry,
      X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
      U - URD, I - Received Source Specific Host Report,
      Z - Multicast Tunnel, z - MDT-data group sender,
      Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 224.11.22.33), 00:01:11/stopped, RP 0.0.0.0, OIF count: 2, flags: DC
  (192.168.1.11, 224.11.22.33), 00:00:08/00:02:59, OIF count: 1, flags: T
  (192.168.2.22, 224.11.22.33), 00:00:06/00:02:59, OIF count: 1, flags: T
  (192.168.3.33, 224.11.22.33), 00:00:04/00:02:59, OIF count: 1, flags: T

(*, 224.0.1.40), 00:03:09/00:03:29, RP 0.0.0.0, OIF count: 2, flags: DCL

RT1#
RT1#sh ip mroute count
IP Multicast Statistics
5 routes using 3074 bytes of memory
2 groups, 1.50 average sources per group
Forwarding Counts: Pkt Count/Pkts per second/Avg Pkt Size/Kilobits per second
Other counts: Total/RPF failed/Other drops(OIF-null, rate-limit etc)

Group: 224.11.22.33, Source count: 3, Packets forwarded: 3, Packets received: 3
  Source: 192.168.1.11/32, Forwarding: 1/0/100/0, Other: 1/0/0
  Source: 192.168.2.22/32, Forwarding: 1/0/100/0, Other: 1/0/0
  Source: 192.168.3.33/32, Forwarding: 1/0/100/0, Other: 1/0/0

Group: 224.0.1.40, Source count: 0, Packets forwarded: 0, Packets received: 0
RT1#
RT1#sh ip mroute active
Active IP Multicast Sources - sending >= 4 kbps

Group: 224.11.22.33, (?)
  Source: 192.168.1.11 (?)
    Rate: 8 pps/6 kbps(1sec), 2 kbps(last 40 secs), 2 kbps(life avg)
RT1#

```

- Interpret the multicast forwarding entries in IP multicast forwarding table in the following way:
  - **(S, G)** – For the source S sending to the group G; traffic is forwarded from the source via the shortest path. These entries typically reflect a source tree, but may also appear on a shared tree. (S, G) entries consume more router CPU and memory resources as there is an entry for each source and group pair. The multicast traffic is sent over the optimal path to each receiver, hence minimizing the delay in packet delivery.
  - **(\*, G)** – For any source (\*) sending to the group G; traffic is forwarded via an RP for this group. These entries reflect a shared tree, but are also created for any existing (S, G) entry. (\*, G) entries consume less router CPU and memory resources, but may result in suboptimal paths from a source to receivers, hence introducing extra delay in packet delivery.
- The **show ip pim interface** [intf-type intf-num] [count | detail | stats] EXEC command displays information about interfaces configured for PIM. The optional **count** keyword displays the number of packets received upon and sent out from the interface. The optional **detail** keyword displays PIM details of each interface. The optional **stats** keyword displays multicast PIM interface octet counts.

```

RT1#sh ip pim interface

Address          Interface          Ver/   Nbr    Query  DR      DR
                  Mode              Count  Intvl  Prior
192.168.1.1      FastEthernet0/0   v2/D   0      30     1       192.168.1.1
12.12.12.1       FastEthernet1/0   v2/D   1      30     1       12.12.12.2
RT1#
RT1#sh ip pim interface count

State: * - Fast Switched, D - Distributed Fast Switched
       H - Hardware Switching Enabled
Address          Interface          FS    Mpackets In/Out
192.168.1.1      FastEthernet0/0   *     1/2
12.12.12.1       FastEthernet1/0   *    15/1
RT1#
RT1#sh ip pim interface stats

Interface          Mpackets In    Mpackets Out      Octets In      Octets Out
Fa0/0              1          2          100
Fa1/0              15         1          980
RT1#

```

**Note:** If all the routers on a multi-access link have the same priority (the default value is 1), highest IP address is the tiebreaker. Point-to-point links do not have DRs (shows 0.0.0.0). The PIM DR has 2 major responsibilities – send IGMP queries onto the LAN; and if PIM sparse mode is running, transmit PIM Join and Register messages to the RP.

- The **show ip pim neighbor** [intf-type intf-num] EXEC command displays the PIM neighbors discovered by PIMv1 Router-Query messages or PIMv2 Hello messages.

```

RT1#sh ip pim neighbor
PIM Neighbor Table
Mode: B - Bidir Capable, DR - Designated Router, N - Default DR Priority,
      S - State Refresh Capable
Neighbor          Interface          Uptime/Expires    Ver  DR
Address                               Prio/Mode
12.12.12.2        FastEthernet1/0    00:11:36/00:01:39 v2   1 / DR S
RT1#

```

- The **mrinfo** [*hostname* | *ip-addr*] EXEC command displays information about multicast routers that are peering with the local router (if no address is specified) or with the specified router.

```
RT1#mrinfo
192.168.1.1 [version 12.3] [flags: PMA]:
  192.168.1.1 -> 0.0.0.0 [1/0/pim/querier/leaf]
  12.12.12.1 -> 12.12.12.2 [1/0/pim]

RT1#
```

The flags indicate the following:

<b>P</b>	Prune capable
<b>M</b>	<b>mtrace</b> capable
<b>S</b>	SNMP capable
<b>A</b>	Auto-RP capable

- The Cisco IOS multicast traceroute tool – the **mtrace** {*source-name* | *source-addr*} [*dest-name* | *dest-addr* [*group-name* | *group-addr*]] EXEC command traces the path from a source to a destination branch for a multicast distribution tree. The trace request generated by the **mtrace** command is multicast to the multicast group to find the last hop router to the specified destination. The trace then follows the multicast path from destination to source by passing the **mtrace** request packet via unicast to each hop. Responses are unicast to the querying router by the first hop router to the source. This command can isolate multicast routing failures.

```
RT1#mtrace 192.168.1.11 192.168.1.11 224.11.22.33
Type escape sequence to abort.
Mtrace from 192.168.1.11 to 192.168.1.11 via group 224.11.22.33
From source (?) to destination (?)
Querying full reverse path...
 0 192.168.1.11
-1 192.168.1.1 PIM [192.168.1.0/24]
-2 192.168.1.11
RT1#
RT1#mtrace 192.168.1.11 192.168.2.22 224.11.22.33
Type escape sequence to abort.
Mtrace from 192.168.1.11 to 192.168.2.22 via group 224.11.22.33
From source (?) to destination (?)
Querying full reverse path...
 0 192.168.2.22
-1 12.12.12.2 PIM [192.168.1.0/24]
-2 12.12.12.1 PIM [192.168.1.0/24]
-3 192.168.1.11
RT1#
RT1#mtrace 192.168.1.11 192.168.3.33 224.11.22.33
Type escape sequence to abort.
Mtrace from 192.168.1.11 to 192.168.3.33 via group 224.11.22.33
From source (?) to destination (?)
Querying full reverse path...
 0 192.168.3.33
-1 23.23.23.3 PIM [192.168.1.0/24]
-2 23.23.23.2 PIM [192.168.1.0/24]
-3 12.12.12.1 PIM [192.168.1.0/24]
-4 192.168.1.11
RT1#
```



- The **mstat** {source-name | source-addr} [dest-name | dest-addr [group-name | group-addr]] EXEC command displays IP multicast packet rate and loss information.

```

RT1#mstat
VRF name:
Source address or name: 192.168.1.11
Destination address or name: 192.168.2.22
Group address or name: 224.11.22.33
Multicast request TTL [64]: [Enter]
Response address for mtrace: [Enter]
Type escape sequence to abort.
Mtrace from 192.168.1.11 to 192.168.2.22 via group 224.11.22.33
From source (?) to destination (?)
Waiting to accumulate statistics.....
Results after 10 seconds:

      Source          Response Dest    Packet Statistics For          Only For Traffic
192.168.1.11      192.168.1.1    All Multicast Traffic          From 192.168.1.11
      |              / rtt 27  ms    Lost/Sent = Pct Rate          To 224.11.22.33
      v              / hop 2435 ms    -----
--
192.168.1.1
12.12.12.1        ?
      |              ^      ttl  0
      v              |      hop  0  ms    -3/0 = --%    0 pps    0/0 = --%  0 pps
12.12.12.2        ?
      |              \     ttl  1
      v              \     hop -2  s      0          0 pps          0    0 pps
192.168.2.22      192.168.1.1
Receiver          Query Source

RT1#

```

## PIM Sparse Mode Configuration

RT1 Configuration:	RT2 Configuration:	RT3 Configuration:
<pre> ! ip multicast-routing ip pim rp-address 192.168.2.1 ! interface FastEthernet0/0 ip pim sparse-mode ! interface FastEthernet1/0 ip pim sparse-mode ! router eigrp 100 network 12.0.0.0 network 192.168.1.0 no auto-summary ! </pre>	<pre> ! ip multicast-routing ip pim rp-address 192.168.2.1 ! interface FastEthernet0/0 ip pim sparse-mode ! interface FastEthernet1/0 ip pim sparse-mode ! interface FastEthernet1/0 ip pim sparse-mode ! router eigrp 100 network 12.0.0.0 network 23.0.0.0 network 192.168.2.0 no auto-summary ! </pre>	<pre> ! ip multicast-routing ip pim rp-address 192.168.2.1 ! interface FastEthernet0/0 ip pim sparse-mode ! interface FastEthernet1/0 ip pim sparse-mode ! router eigrp 100 network 23.0.0.0 network 192.168.3.0 no auto-summary ! </pre>

Figure 20-3: PIM-SM Configuration on RT1, RT2, and RT3

- Below shows the IP multicast routing table on RT1 after PC1, PC2, and PC3 have ping 224.11.22.33.

```

RT1#sh ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 224.11.22.33), 00:09:11/stopped, RP 192.168.2.1, flags: SJCF
  Incoming interface: FastEthernet1/0, RPF nbr 12.12.12.2
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:09:11/00:02:55

(192.168.1.11, 224.11.22.33), 00:00:18/00:03:24, flags: FT
  Incoming interface: FastEthernet0/0, RPF nbr 0.0.0.0, Registering
  Outgoing interface list:
    FastEthernet1/0, Forward/Sparse, 00:00:18/00:03:20, A

(192.168.2.22, 224.11.22.33), 00:00:48/00:02:54, flags: JT
  Incoming interface: FastEthernet1/0, RPF nbr 12.12.12.2
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:00:48/00:02:55

(192.168.3.33, 224.11.22.33), 00:00:39/00:02:24, flags: JT
  Incoming interface: FastEthernet1/0, RPF nbr 12.12.12.2
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:00:39/00:02:55

(*, 224.0.1.40), 00:10:28/00:02:53, RP 192.168.2.1, flags: SJCL
  Incoming interface: FastEthernet1/0, RPF nbr 12.12.12.2
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:10:28/00:02:53

RT1#

```

- The configuration above uses the **ip pim rp-address** {rp-addr} global configuration command to statically define RT2 as the RP for the multicast groups operating in PIM sparse mode. Use the **ip pim send-rp-announce** {intf-type intf-num | ip-addr} **scope** {ttl} [**group-list acl**] global configuration command to use the Auto-RP mechanism to define a router as an RP and distribute Group-to-RP mappings. This command causes the router to send an Auto-RP announcement message to the well-known multicast group Cisco-RP-Announce (224.0.1.39) to announce the router as a candidate RP for the groups in the range described by the access list. Use the **ip pim send-rp-discovery** [intf-type intf-num] **scope** {ttl} global configuration command to configure a router as an RP mapping agent. An RP mapping agent listens to the 224.0.1.39 multicast group to receives Auto-RP announcement messages, stores them into its local Group-to-RP mapping cache, uses the information in the Auto-RP announcement messages to elect the RP (highest IP address), and sends Auto-RP discovery messages to the 224.0.1.40 Cisco-RP-Discovery multicast group. Other PIM routers that join the 224.0.1.40 multicast group by default automatically discover the RP from the RP mapping agent and store the information about the RP in their local Group-to-RP mapping caches.

- The **ip pim spt-threshold** {rate | infinity} [group-list acl] global configuration command configures when a PIM leaf router should switchover from the shared tree to the shortest path source tree in PIM sparse mode. When this command is not used, the PIM leaf router joins the shortest path source tree immediately after the first packet arrives from a new source. The **infinity** keyword indicates that the switchover will never occur – will use the shared tree. If a source sends at a rate  $\geq$  the *rate* value (in kbps), a PIM join message is triggered toward the source to construct a source tree. When the traffic rate from the source drops below the threshold traffic rate, the leaf router will switch back to the shared tree and send a prune message toward the source. Specify an access list to define the multicast groups to which the threshold applies.
- The RP for a multicast group operating in PIM sparse mode must be reachable and from the router. The **show ip pim rp** [group-name | group-addr] | mapping] EXEC command is often used to troubleshoot RP in addition to the standard tools, eg: unicast **ping** to check the RP reachability. This command displays RP information about active groups or as specified with the group name or group address. The **mapping** keyword displays the contents of the Group-to-RP mapping cache indicating which RP is active for which group range. The mapping cache is populated via the static RP assignments, Auto-RP, or BSR mechanisms. It contains detailed information such as the IP address of the router that distributed the information, or local when the source of the information is the local router that either has manual RP configuration or is a source of automatically distributed information, the mechanism by which the information was determined – Static, Auto-RP, or BSR; and whether the router is operating as a candidate RP, mapping agent, or BSR.

```

RT2#sh ip pim rp
Group: 224.11.22.33, RP: 192.168.2.1, next RP-reachable in 00:00:59
Group: 224.0.1.40, RP: 192.168.2.1, next RP-reachable in 00:00:59
RT2#
RT2#sh ip pim rp mapping
PIM Group-to-RP Mappings

Group(s): 224.0.0.0/4, Static
        RP: 192.168.2.1 (?)
RT2#

```

- The **show ip rpf** {ip-addr | name} EXEC command displays RPF information for the RP or any source specified. The specified address is not necessarily to be a currently active source. Specifying the address of the RP is useful in determining the RPF information for the shared tree.

```

RT2#sh ip rpf 192.168.2.1
RPF information for ? (192.168.2.1)
  RPF interface: FastEthernet0/0
  RPF neighbor: ? (192.168.2.1) - directly connected
  RPF route/mask: 192.168.2.0/24
  RPF type: unicast (connected)
  RPF recursion count: 0
  Doing distance-preferred lookups across tables
RT2#
RT2#sh ip rpf 192.168.1.11
RPF information for ? (192.168.1.11)
  RPF interface: FastEthernet1/0
  RPF neighbor: ? (12.12.12.1)
  RPF route/mask: 192.168.1.0/24
  RPF type: unicast (eigrp 100)
  RPF recursion count: 0
  Doing distance-preferred lookups across tables
RT2#

```

- The RPF interface is the interface in the direction of the source or RP; while the RPF neighbor is the address of the next-hop router in the direction of the source or RP. The RPF type indicates the source of RPF information; unicast indicates that the information was derived from the unicast routing table; other RPF types include DVMRP, Multiprotocol Border Gateway Protocol (MBGP) Extensions for IP Multicast, or static. RPF information is essential in multicast routing, and requires attention when inspecting the PIM-SM information because of the possible coexistence of shared and source trees.

## IGMP Snooping

- IGMP snooping is enabled on all switch ports and VLAN interfaces by default on Cisco Catalyst switch platforms that support it. The **[no] ip igmp snooping** global configuration command enable or disable IGMP snooping respectively.
- Below shows the output of the **show ip igmp snooping** EXEC command on a Catalyst 2950. It displays the IGMP snooping configuration of the switch.

```
Switch#sh ip igmp snooping
Global IGMP Snooping configuration:
-----
IGMP snooping                : Enabled
IGMPv3 snooping (minimal)    : Enabled
Report suppression           : Enabled
TCN solicit query            : Disabled
TCN flood query count        : 2
Last member query interval   : 1000

Vlan 1:
-----
IGMP snooping                : Enabled
Immediate leave               : Disabled
Multicast router learning mode : pim-dvmrp
Source only learning age timer : 10
Last member query interval    : 1000
CGMP interoperability mode    : IGMP_ONLY

Switch#
```

*This page is intentionally left blank*

## General Operating System Concepts

- Modern operating systems provide 2 primary functions:

<b>Hardware Abstraction</b>	Provides software developers a common interface between their applications and the hardware, which shields the programmers from the complexity of the hardware. The hardware-specific codes are developed once in the OS and shared by everyone.
<b>Resource Management</b>	Managing the hardware resources (eg: CPU, memory, disk space) so that they can be shared efficiently among multiple applications. Like hardware abstraction, the resource management codes are developed in the OS to keep application programmers from reinventing the wheels – writing the resource management codes.

- **Multitasking** is referred to as running multiple programs at once. Applications written for multitasking OSes often contain multiple **independent** and **concurrent** tasks, called **threads**. Each thread has its own set of CPU register values, but can share the same memory address space with other threads belong to the same application (or process). A **process** is a group of threads that share a common memory space and have a common purpose. On OSes and CPUs that support virtual memory, each process runs in a separate address space that is protected from other processes.
- A processor can execute instruction for only one program at a time. **Scheduling** decides which process or thread should run. This function is usually performed by the core of the OS – **kernel**. An OS can use one of the several scheduling methods to schedule threads, depends on the type of applications (eg: batch, interactive, real-time). Different types of applications have different CPU utilization characteristics, and the overall performance is affected by the scheduling method used.
- **FIFO with run-to-completion** is the simplest scheduling method where each thread is assigned to the processor in the order its run request is received and let all threads run to completion. It is easy to implement, has very low overhead, and fair – all threads are treated equally. First come, first served.
- FIFO with run-to-completion scheduling is good for batch applications that perform serial processing and then exit, but it does not work well for interactive and real-time applications, which require fast and short duration access to the CPU to provide quick response to users or other devices. A possible scheduling method for these applications is **priority scheduling**, where priorities are assigned to different types of threads – critical threads, eg: real-time threads are assigned with a higher priority than other less critical threads, eg: batch threads. Multiple threads with the same priority are then processed in the order which they were received (FIFO).
- Even though priority scheduling is an improvement over FIFO, it still has one drawback that makes it unsuitable for interactive and real-time applications – high-priority threads can get stuck behind a long running low-priority thread. A method that can temporarily suspend or preempt a running thread to allow other threads to access the CPU is required to solve this problem.

- **Thread Preemption** is the ability of an OS to involuntarily suspend a running thread to allow another higher priority thread to access the CPU resources. Preemptive multitasking OSes employ preemptive scheduling methods that utilize preemption instead of run-to-completion.
- Preemption relies on the kernel to periodically change the current thread via a **context switch**, which can be triggered with either a system timer (each thread is assigned a time slice) or a function call to the kernel. When a context switch is triggered, the kernel selects the next thread to run and queue the preempted thread in a list for it to run again at its next opportunity – the computer actually changes the task that it is currently working on.
- Context switching can be quite expensive in term of system performance as all processor registers for the thread that is being taken off must be saved, and all processor registers for the thread that is being granted access to the CPU must be restored.

- Below describes the advantages and disadvantages of preemptive multitasking:

Advantages	Disadvantages
<b>Predictable.</b> A thread can be set up to run once a second and the programmer can be reasonably certain that the thread will be scheduled to run at that interval.	<b>Less efficient.</b> It tends to switch contexts more often, and the CPU spends more time for scheduling and switching between threads than run-to-completion approach.
<b>Difficult to break.</b> No single thread can monopolize the CPU for long period and stop other threads from running.	<b>Adds complexity to applications.</b> A thread can be interrupted anywhere. Applications must be well designed and written to protect critical data structures from being changed by other threads when they are being preempted.

- OSes also manage the memory resources by dividing them into various sections for storing actual application instruction codes, variables, and heap. The heap is a section of memory from which processes can dynamically allocate and free the memory resources.
- **Virtual memory** is found in most modern OSes to provide more memory than the available physical RAM size, and is transparent to processes. The computer memory is expanded with secondary storage, eg: harddisk drive. Virtual memory is created using a hardware feature that available on some CPUs – **memory map unit (MMU)**. MMU automatically remaps memory address requests to either physical memory (RAM) or secondary storage (harddisk) depends on where the contents actually reside. MMU can be programmed to create separate address spaces for all processes to prevent them from accessing the memory address space of other processes.
- **Segmentation** → A running process is restricted to use certain parts of memory called segments. If a process read, or write data outside the permitted memory address space or virtual memory (paging) allocated for it, a **general protection fault** or **page fault** will occur respectively.
- Although virtual memory has many benefits, but there are resource requirements and performance penalties. As a result, IOS is not implemented with a full virtual memory scheme.
- OSes usually support interrupts, a hardware feature that cause the CPU to temporarily suspend its current instruction sequence and transfer control to a special program called **interrupt handler**. OSes usually provide a set of interrupt handlers for all possible interrupt types.

## Cisco IOS Architecture

- The main challenge for IOS is to switch or forward packets as quickly and efficiently as possible.
- IOS is a specialized embedded operating system tightly coupled to the underlying hardware. Therefore always consider the hardware architecture when discussing about the software.
- Cisco IOS has a **monolithic** architecture (a single large program), which means it runs as a single image and all processes share the same memory address space. There is no memory protection mechanism exists due to the CPU and memory overhead they introduce – a bug in a process can potentially corrupt the data and/or memory of another process.
- Cisco IOS has a run-to-completion scheduler – the kernel does not preempt a running process. A process must make a kernel call to allow other processes to run. For Cisco routers that require very high availability and response time, eg: Cisco CRS-1, these limitations are not acceptable. Competitive network OSES (eg: Juniper JUNOS) were designed not to have these limitations.
- A new version of Cisco IOS – IOS-XR was developed to offer modularity, memory protection, lightweight thread, and preemptive scheduling.
- **IOS software modularity** is a new IOS feature that supports **individually restartable process**. Past patches would require a complete IOS update and system reboot, whereas the new modular kernel allows the affected processes to be **patched independently**. After patching, the particular processes can be **restarted independently** from other processes, hence reduces system downtime.
- **Microcode** is a set of processor-specific software instructions that enables and manages the features and functions of a specific processor type. A router loads the microcode for each processor type present in the system upon system startup or reload. The latest available microcode image is bundled and distributed along with the system software image.

## Memory Organization

- IOS maps the entire physical memory into a large flat virtual address space. Since the kernel does not perform any memory paging or swapping, the virtual address space is limited to the available physical memory. IOS divides the address space into various areas called **regions** that correspond to the various types of physical memory, eg: SRAM and DRAM. Classifying memory into regions allows IOS to group various types of memory and therefore the software does not need to know about specific types of memory on every platform.
- Memory regions can also be nested in a parent-child relationship. Although there is no imposed limit upon the depth of nesting, only one level is really used. Nested regions form **subregions** of the parent region. Subregions are denoted by a name with the **:** separator and no parentheses.



- Memory regions are classified into the following 8 categories:

Memory Region Class	Characteristics
Local	Normal run-time data structures and local heaps. Often in DRAM.
Iomem	Shared input / output memory that is visible to both the CPU and controllers of network interfaces. Often is DRAM.
Fast	Fast memory, eg: SRAM for speed-critical tasks and special purpose.
IText	A region for the code currently executed by the IOS.
IData	Storage for the initialized variables.
IBss	Storage for non-initialized variables. BSS – Block Storage Section.
PCI	Peripheral Component Interconnect (PCI) memory accessible to all devices on the PCI bus.
Flash	Flash memory. Often used to store configuration backups, the binaries of IOS images, and other data, eg: crash info. A file system is typically built in the Flash memory region.

- The **show region** privileged command displays the regions defined on a system.

```

Router#sh region
Region Manager:

      Start          End          Size (b)  Class  Media  Name
0x0B000000  0x0BFFFFFF  16777216  Iomem  R/W    iomem
0x60000000  0x6AFFFFFF  184549376 Local  R/W    main
0x600089A4  0x643611BF  70617116  IText  R/O    main:text
0x64362000  0x6621F9BF  32233920  IData  R/W    main:data
0x6621F9C0  0x66A471BF  8550400   IBss   R/W    main:bss
0x66A471C0  0x67A471BF  16777216  Local  R/W    main:heap
0x67A47218  0x68A47217  16777216  Local  R/W    main:heap
0x6A000000  0x6AFFFFFF  16777216  Local  R/W    main:heap
0x7B000000  0x7BFFFFFF  16777216  Iomem  R/W    iomem:(iomem_cwt)
0x80000000  0x8AFFFFFF  184549376 Local  R/W    main:(main_k0)
0xA0000000  0xAAFFFFFF  184549376 Local  R/W    main:(main_k1)

Free Region Manager:

      Start          End          Size (b)  Class  Media  Name
0x68A47270  0x69FFFA7    22777144  Local  R/W    heap

Router#

```

- The gaps between the address space, eg: **iomem** ends at 0x0BFFFFFF and **main** begins at 0x60000000, are intentional to allow for expansion of the regions and provide protection against errant threads. If a runaway thread is advancing through memory and writing garbage into a free unallocated memory region (the gap), it will be stopped.
- The entire DRAM area from 0x60000000 to 0x6AFFFFFF is classified as a **local** region and further divided into subregions that correspond to the various parts of the IOS image itself (text, BSS, and data) and the heap. The **heap** is the entire free Local memory that left over after the IOS image was loaded into it.

- Below shows that some regions appear to be duplicates with different address ranges.

0x0B000000	0x0BFFFFFF	16777216	Iomem	R/W	iomem
0x7B000000	0x7BFFFFFF	16777216	Iomem	R/W	iomem:(iomem_cwt)

These duplicate regions are called **aliases**. Some Cisco platforms have multiple physical address ranges that point to the same block of physical memory. These different ranges are used to provide alternate data access method or automatic data translation in hardware.

For example, one address range might provide cached access to an area of physical memory while another might provide uncached access to the same memory.

- The duplicate ranges are mapped as alternate views during system initialization and IOS creates alias regions for them. Aliased regions do not count toward the total memory on a platform, as they are not really separate memory! They allow IOS to provide a separate region for alternate memory views without artificially inflating the total memory calculation.

### IOS Processes

- An IOS process is equivalent to a single thread in other OSes – IOS processes have only one thread each. Each process has its own memory block (stack) and CPU context (eg: registers), and can control resources such as memory and a console device. IOS does not implement virtual memory protection between processes in order to minimize the overhead of paging or swapping. No memory management is performed during context switches; therefore, although each process receives its own memory allocation, other processes can freely access the same memory space – nothing can stop a process from intruding into a memory block of another process! Cisco has sacrificed both stability and security features in the IOS architecture design to achieve higher productivity and reduce resource consumptions.
- Processes can be created and terminated anytime by the kernel (during IOS initialization) or by another running process while IOS is operating except during an (hardware) interrupt. When the CPU is interrupted, it temporarily suspends the execution of instructions of the current thread and begins to run an interrupt handler function. New processes cannot be created while the CPU is running the interrupt handler.
- The **parser** is responsible for creating many of the IOS processes. The parser is a set of functions that interpret IOS configuration and EXEC commands. It is invoked by the kernel during IOS initialization and EXEC processes that are providing a CLI to the console and Telnet sessions. Anytime a command is entered by a user or a configuration line is read from a file, the parser interprets the text and takes immediate action. Some commands result in setting of a value, eg: an IP address; while others enable complicated functionality, eg: routing or event monitoring.
- Some commands result in starting a new process, eg: when the **router eigrp** command is entered via the CLI, the parser starts a new process called ipigrp if it has not already been started.

## Device Drivers

- A primary OS function is provides hardware abstraction between platform hardware and the programs that run on it. Hardware abstraction typically occurs in device drivers that are part of the OS, making them an integral part of the system. IOS contains device drivers for a range of platform hardware devices, eg: Flash cards and NVRAM, but most notable are the device drivers for network interfaces.
- IOS network interface device drivers provide the primary intelligence for packet operations in and out of interfaces. Each driver consists of a control component and a data component. The control component is responsible for managing the state and status of the device, eg: shutting down an interface; while the data component is responsible for all data flow operations through the device and contain logic that assists with the packet switching operations. IOS device drivers are very tightly coupled with the packet switching algorithms.
- IOS device drivers interface with the rest of the IOS system via a special control structure called **Interface Descriptor Block** (idB). The idB contains entry points into the functions of a device driver and data about the state and status of a device, eg: the IP address, the interface state, and the packet statistics are some of the fields in the idB. IOS maintains an idB for each interface present on a platform and maintains an idB for each subinterface.

## Appendix 2

# Cisco IOS Packet Switching Architectures

- Cisco IOS packet switching algorithms (or paths) are the most foundation components of Cisco routers. Different switching methods and structures affect how a router performs its primary task.
- In Ethernet, switching is the process of forwarding frames based on destination MAC addresses. In routers, switching is the process of forwarding packets between interfaces within a router.
- Packet switching is responsible for moving packets across a router. Its most basic operations are:
  - i) A packet is received from an inbound interface.
  - ii) The packet's destination address is inspected and compared against the routing table.
  - iii) If there is a match, the packet is forwarded out to the appropriate outbound interface.
  - iv) If there is no match, the packet is discarded or dropped.
- The main concern of IOS is not much on how to switch packets, but how to switch them **quickly**. Packet switching is a **data-intensive** operation as opposed to a computation-intensive operation. Speeding up the operation is not as simple as using a faster CPU. Other factors, eg: I/O bus and memory technology, can also have big impact on the packet switching performance.
- The challenge of IOS developers is to create the highest possible performance based on the limitations of the given CPU, I/O bus, and memory technology.
- Process switching was the only switching method when IOS was first developed. Later IOS releases introduced newer improved switching methods. Some methods rely on **hardware-specific optimizations** while other use **software techniques** that work well on many platforms.
- Below lists the switching methods developed and available as of Cisco IOS release 12.0:
  - i) Process Switching
  - ii) Fast Switching
  - iii) Autonomous Switching
  - iv) Silicon Switching Engine (SSE) Switching
  - v) Optimum Switching
  - vi) Distributed Fast Switching
  - vii) Cisco Express Forwarding (CEF)
  - viii) Distributed Cisco Express Forwarding (dCEF)
- This chapter covers only 4 of these methods in detail – process switching, fast switching, optimum switching, and Cisco Express Forwarding. Autonomous and SSE switching are platform-specific and aren't commonly used in today's networks. Distributed fast switching is actually an implementation of optimum switching on intelligent line cards.

## Process Switching

- Process switching is available on every version of IOS, and every platform. Its main disadvantage against other switching architectures is its **speed**, as it is required to perform routing table lookup for every packet and has the least amount of performance optimizations, and consumes large amounts of CPU resources. However, it has the advantage of being platform-independent and provides some load balancing capabilities not found in other switching methods.

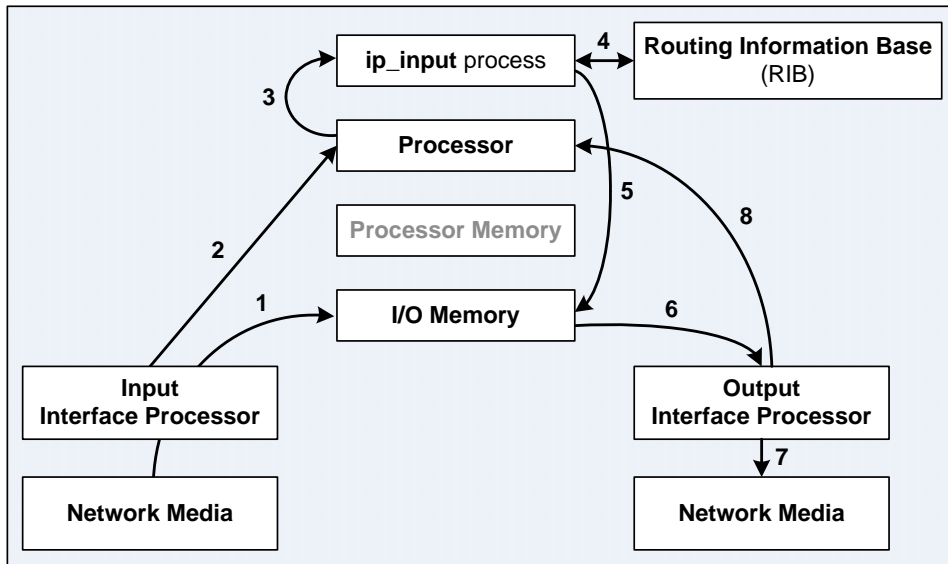
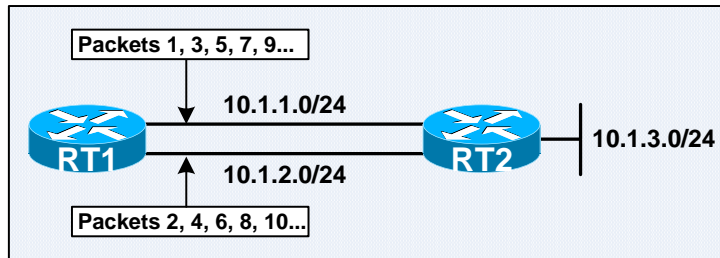


Figure A2-1: Process Switching

- Below lists the steps for process switching:
  - 1) The network interface processor detects there is a packet on the network media that needs to be processed. It receives and transfers the packet to the I/O memory.
  - 2) The interface processor generates a **receive interrupt** which interrupts the processor and inform it that there is a received packet waiting in I/O memory that needs to be processed. The IOS interrupt handler inspects the packet's header information (eg: encapsulation type, network layer header) to determine packet type, and copies it into the processor memory if necessary (platform dependant). The processor then places the packet on input queue of the appropriate switching process and releases the interrupt. The switching process for IP packets is called **ip\_input**.
  - 3) During the next time the scheduler runs, it notices the presence of a packet in the input queue for the **ip\_input** process, and schedules the process for execution.
  - 4) The actual packet forwarding operation begins when the **ip\_input** process runs. It consults the RIB (routing table) to determine the next-hop address and the output interface, and then consults the ARP cache to retrieve the corresponding physical layer address for the next-hop.
  - 5) The **ip\_input** process rewrites the MAC header of the packet with the appropriate addresses, and places the packet on the output queue of the appropriate outbound network interface. The packet is queued for transmission.
  - 6) The packet is moved from the output queue of the outbound interface to the transmit queue of the outbound interface. Any outbound QoS operation takes place between these two queues.
  - 7) The output interface processor detects a packet in its transmit queue and waiting for transmission. It dequeues the packet and transmits it onto the network media.
  - 8) After the output interface processor transmitted the packet, it interrupts the processor to inform that the packet has been transmitted. IOS then updates the outbound packet counters and frees or releases the space in I/O memory formerly occupied by the packet.
  
- The key points in the process that make it slow is that the processor waits for the next scheduled execution of the **ip\_input** process; and when the **ip\_input** process finally runs, it refers the RIB – a very slow process. The **ip\_input** process is run at the same priority level as other processes, eg: routing protocols, and the HTTP web server interface. Process switching should never be used as the switching method of choice. Other methods produce significantly better performance.
  
- Packet sourced from or destined to the router itself, eg: SNMP traps from the router and Telnet packets destined for the router, are always process-switched.

## Load Balancing with Process Switching

- One of the advantages of process switching is **per-packet load balancing**, which provides a simple way to route traffic over parallel paths to a destination – packets are distributed among the available paths based on the **routing metric** (or **path cost**) assigned to each path. The metric or cost is used to calculate a **load share** counter, which actually determines which path to take.
- **Note:** Some applications, eg: VoIP, cannot tolerate per-packet load balancing because packets may arrive out of order and affect the processing of the data. Always ensure that load balancing is performed on a per-destination basis when the network is supporting such applications.

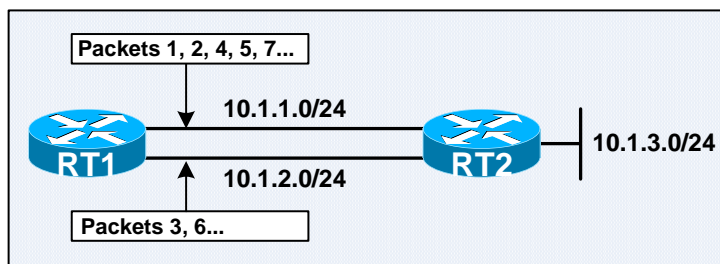


**Figure A2-2: Equal Cost Path Load Balancing**

- Below shows that RT1 has 2 equal cost paths to the 10.1.3.0/24 network. The asterisk character (\*) next to one of the paths indicates the path that will be used for the next packet switched toward 10.1.3.0/24. The traffic share count on both paths is 1, which means packets are switched out each path in a round-robin basis.

```
RT1#sh ip route 10.1.3.0 255.255.255.0
Routing entry for 10.1.3.0/24
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
  * 10.1.1.2
    Route metric is 0, traffic share count is 1
  10.1.2.2
    Route metric is 0, traffic share count is 1
RT1#
```

- IGRP and EIGRP can install unequal cost paths into the routing table. Figure A2-3 below shows a modified setup in Figure A2-2, with the upper path has about twice the bandwidth of the other.



**Figure A2-3: Unequal Cost Path Load Balancing**

- Below shows RT1 has 2 unequal cost paths to the 10.1.3.0/24 network. The lower-cost path through 10.1.1.1 has a traffic share count of 2, while the higher-cost path through 10.1.2.1 has a traffic share count of 1. In such a way, for every 2 packets switched out the higher-cost path, 1 packet is switched out the lower-cost path, as indicated by the packet numbers in the figure.

```

RT1#sh ip route 10.1.3.1
Routing entry for 10.1.3.0/24
  Known via "eigrp 100", distance 90, metric 1538560, type internal
  Redistributing via eigrp 100
  Last update from 10.1.2.2 on Serial0/1, 00:02:15 ago
  Routing Descriptor Blocks:
  * 10.1.1.2, from 10.1.1.2, 00:02:15 ago, via Serial0/0
    Route metric is 1538560, traffic share count is 2
    Total delay is 20100 microseconds, minimum bandwidth is 2500 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 1
  10.1.2.2, from 10.1.2.2, 00:02:15 ago, via Serial0/1
    Route metric is 3074560, traffic share count is 1
    Total delay is 20100 microseconds, minimum bandwidth is 1000 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 1

RT1#

```

- Although per-packet load balancing gives the most even distribution across parallel paths, it has a significant drawback – it **does not preserve** packet order, packets can arrive out of order due to the varying latency and delay factors throughout the various routing paths. Out-of-order packet processing can significantly degrade system performance, reduce TCP session throughput, loss of data in some UDP-based protocols (eg: SNA or NetBIOS Fast Sequenced Transport (FST), or Voice-over-IP – VoIP), and might be even interpreted as attacks by some firewalls.
- As mentioned earlier, the main disadvantage of process switching is its **speed**. Process switching requires routing table and ARP cache lookup for every packet, the time required to perform a lookup grows along with the sizes of the routing table and ARP cache. Recursive routes also require additional lookups in the routing table.
- Another factor that affects the speed of process switching is **in-memory data transfer time**. On some platforms, packets must be copied between I/O memory and processor memory before and after the switching process. Memory data copy operations are very CPU intensive.
- **Fast Cache** was deployed to overcome the speed limitation of process switching. It allows IOS to have smaller lookup tables, switches packets during the packet receive interrupt, and removes the need for in-memory data copying.

## Interrupt Context Switching

- Interrupt context switching is another switching method that is much faster than process switching. Below are the main differences between interrupt context switching and process switching:
  - i) The current process is interrupted to perform packet switching. Packets are switched during the packet receive interrupt, rather than the scheduled **ip\_input** process. The **ip\_input** process is rarely called, and the processor no longer has to wait for a process to complete.
  - ii) The processor uses route cache to find all the information needed to switch the packet. The type of route cache being used and the contents in the cache depend on the router hardware and the type of interrupt content switching – fast switching, optimum switching, or Cisco Express Forwarding (CEF).
  - iii) Cache entries are built by the **ip\_input** process while packets are first process switched. The first packet sent to a destination will always be process switched. Subsequent packets to the same destination can be fast switched and the scheduled **ip\_input** process does not get involved in switching the packets as long as the cache entry exists.
  
- A **cache** is a special high-speed storage mechanism used to store some frequently access data. A cache has smaller size or capacity, higher speed, and more expensive than normal main memory. It is being used to achieve higher performance as applications can access frequently access data in a faster manner compared to accessing the slower main memory.

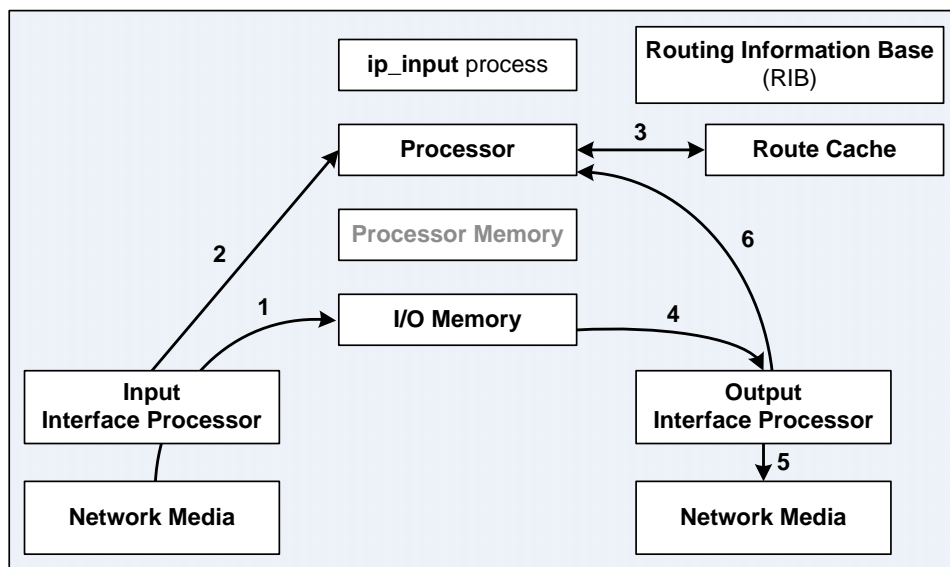


Figure A2-4: Interrupt Context Switching

- Below are the steps of interrupt context switching and how cache is being used in the switching process:
  - 1) The network interface processor detects there is a packet on the network media that needs to be processed. It receives and transfers the packet to the I/O memory.
  - 2) The interface processor generates a **receive interrupt**. During the interrupt, the IOS interrupt handler determines the packet type, and then begins the switching process.
  - 3) The process consults the route cache to determine the next-hop address, the output interface, and the MAC header for the next-hop (used for rewriting the MAC header of the packet). **Note:** A cache entry is built by the **ip\_input** process during the first lookup process.
  - 4) The packet is copied to either the output or transmit queue of the outbound interface depends upon various factors. The receive interrupt is ended, and the originally running process continues.
  - 5) The output interface processor detects a packet in its transmit queue and waiting for transmission. It dequeues the packet and transmits it onto the network media.



6) After the output interface processor transmitted the packet, it interrupts the processor to indicate that the packet has been transmitted. IOS then updates the outbound packet counters and frees or releases the space in I/O memory formerly occupied by the packet.

- Notice that the **ip\_input** process never gets involved when switching the packet. The currently running process is interrupted, instead of waiting for the next scheduled execution of the **ip\_input** process. No scheduled process is involved when a packet is fast switched as long as a cache entry exists. IOS can now perform the entire packet switching operation with very short period of an interrupt using Fast Cache! Caching allows IOS to separate the resource-intensive task of making a routing decision from the relatively easy task of forwarding a packet. Fast switching introduced the concept of “route once, forward many times”.
- The **show ip cache verbose EXEC** command displays the routing table cache for fast switching.

```
RT1#sh ip cache verbose
IP routing cache 2 entries, 344 bytes
  5 adds, 3 invalidates, 0 refcounts
Minimum invalidation interval 2 seconds, maximum interval 5 seconds,
  quiet interval 3 seconds, threshold 0 requests
Invalidation rate 0 in last second, 0 in last 3 seconds
Last full cache invalidation occurred 00:00:32 ago
```

Prefix/Length	Age	Interface	Next-hop
10.1.3.1/32-24	00:00:30	Serial0/1	10.1.2.2
	4 0F000800		
192.168.1.2/32-24	00:00:30	FastEthernet1/0	192.168.1.2
	14 CC020F400000CC000F4000100800		

```
RT1#
```

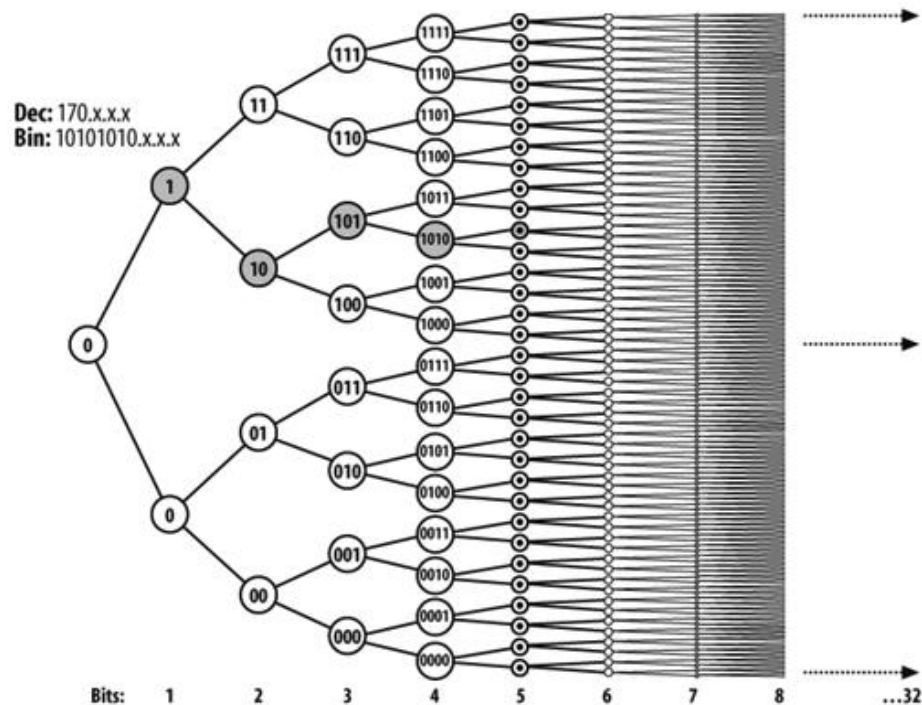
```
RT1#sh arp
```

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
Internet	192.168.1.1	-	cc00.0f40.0010	ARPA	FastEthernet1/0
Internet	192.168.1.2	1	cc02.0f40.0000	ARPA	FastEthernet1/0

```
RT1#
```

- Unlike the master tables that used to build the caches, which are basically just big lists, caches are implemented with well organized data structures to achieve fast retrieval of a particular entry with minimal search operations. The searching time increases proportionately with the size of the table for the master tables; while the searching time for the caches can be kept small and fairly constant regardless of the total number of entries.

- The IP Fast Cache was first implemented as a data structure called **hash table**. In Cisco IOS Release 10.2, it was replaced with another data structure called **2-way radix tree**, a form of binary tree. A radix tree stores the forwarding information and MAC headers based on the binary representation of the key (the unique field that uniquely identifies each set of data). The binary tree can have up to 32 levels, corresponding to the bits within an IP address.
- Below shows the steps of searching a number throughout a binary tree:
  - i) Start from the left (the most significant bit) of the binary number to be searched for.
  - ii) Begin at the root of the tree, branch left or right in the tree based on that digit.
  - iii) Compare the next digit until reaching an end node.



**Figure A2-5: The Fast Switch Binary Tree**

- For visualizing purpose, the nodes that marked in grey in the figure above match an IP address of 170.x.x.x (the binary value of 170 is 10101010).
- The benefit of this design is speed when compared with searching the RIB, as information regarding the next-hop and MAC address changes is stored within each end node, and finding specific entries is very quick since the tree is very deterministic.
- The route cache is not directly related to the routing table, and it is updated only when packets are being process-switched. To keep the entries in the route cache current and from losing their synchronization with the routing table and ARP cache, as well as to keep unused entries in the fast cache from unduly consuming memory on the router, 1/20th of the route cache is aged out or discarded randomly every minute. Those entries must then be rebuilt through process switching.
- And since there is no correlation between the MAC headers (used for rewrites) in the ARP cache and the structure of the fast cache, when the ARP table changes, some portion of the fast cache must be invalidated and rebuilt through process switching.

## Traffic Load Sharing with Fast Switching

- Fast switching does not support load sharing on a per-packet basis, which could lead to uneven link utilization in some situations where multiple paths exist between 2 destinations. This lack of a deterministic load balancing scheme is a concern for many network designers. Newer switching algorithms, eg: Cisco Express Forwarding (CEF), now support load balancing schemes that overcome this problem.
- Load sharing with fast switching occurs on a per-destination basis. When there are multiple equal cost paths for a particular destination network, fast cache has an entry for each host reachable within that network, and all traffic destined for a particular host follows a particular link.

## Optimum Switching

- Optimum switching is basically fast switching with some caching optimizations. As like fast switching, optimum switching performs the entire packet switching process during a single packet receive interrupt. The main difference between fast and optimum switching is the way that the route cache is accessed. The optimum switching code is also developed to take advantages of certain processor architectures; while the fast switching code is generic, and is not optimized for any specific processor. Unlike fast switching that utilizes binary tree that is designed to support any type of value, optimum switching is available only for the IP protocol.
- Optimum switching utilizes a **256-way multiway tree** instead of a binary tree to record and retrieve information in the route cache. For IP addresses, a multiway tree is much faster because each octet can only be one of 256 values.

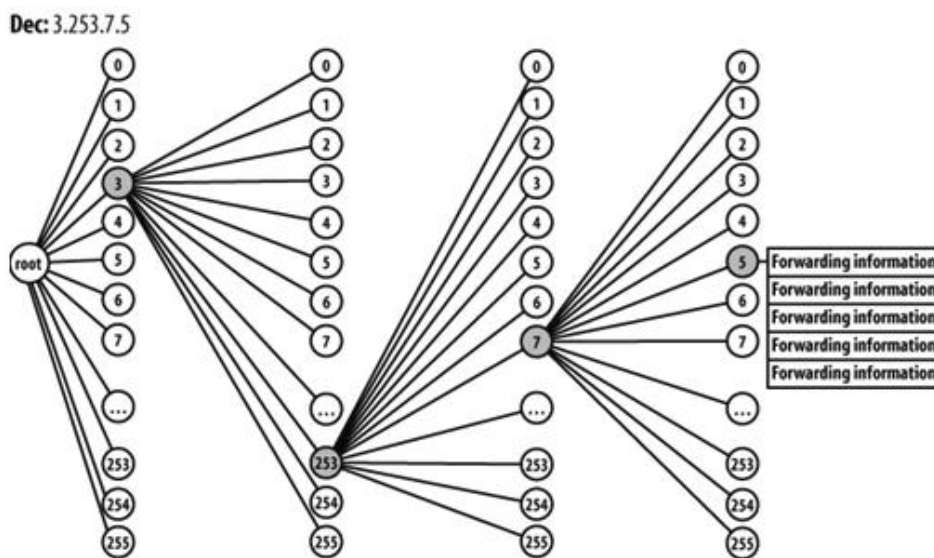


Figure A2-6: Optimum Switching 256-way Multiway Tree

- The root branches into 256 nodes numbered 0 to 255. Each node then branches into an additional 256 nodes. This pattern continues for 4 levels, one for each octet in the IPv4 addressing. The nodes in grey show how an IP address (3.253.7.5) would be matched. This type of tree is much faster for IP address lookups than a binary tree because the table is optimized for the IP address format. Information for each route (prefix) or IP address is stored in the final node. Since the size of this information is variable and each node may or may not contain information, the overall table size is also variable; and this is a drawback – searching the tree is not as efficient as it might be if every node were of a known static size.

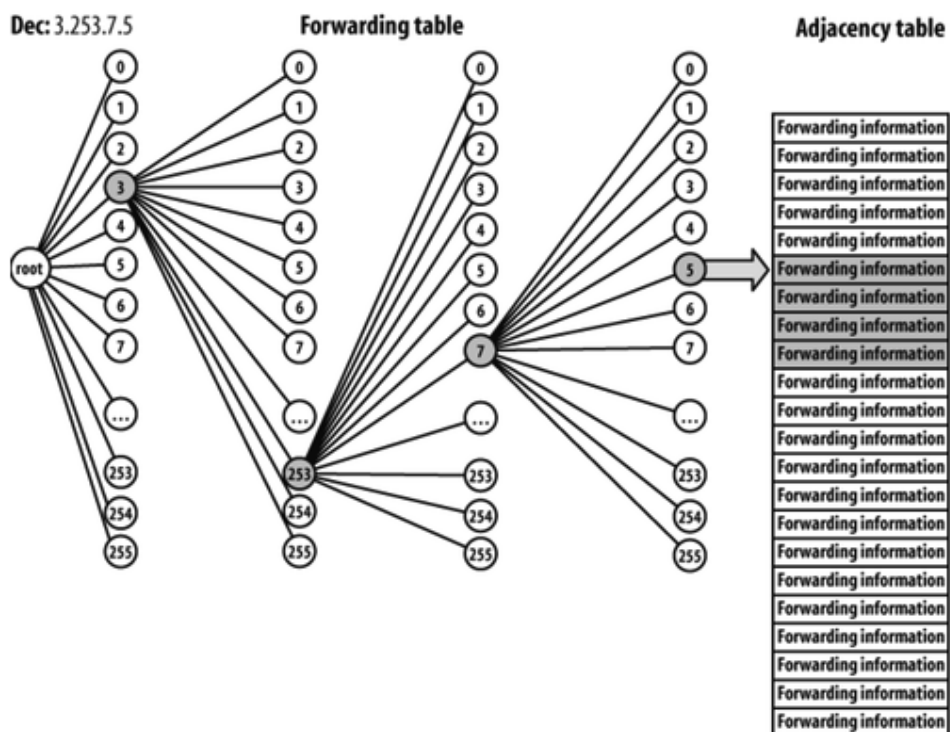
- Since the information stored in the nodes has no direct relationship to the RIB and ARP cache, the entries are also aged out and rebuilt through process switching as like with fast switching.
- Optimum switching is available only on high-end routers, eg: Cisco 7500 and 7600 series.

## Cisco Express Forwarding (CEF)

- Cisco Express Forwarding (CEF) is the preferred switching path on any router that supports it and is the default switching path on all modern routers.
- CEF introduces the concept of a **trie** (the **forwarding table – FIB**) upon the optimum switching multiway tree, in which the data is not stored within the nodes, but rather each node that is the same size becomes a pointer to another table – the **adjacency table**, that contains the data, eg: next-hop information and MAC header, that are needed to perform packet switching. The forwarding table is built from the routing table; while the adjacency table is built from the ARP table, Frame Relay map table, or other L2 table types.

**Note:** The term *trie* comes from the word *retrieve*, and is pronounced like *tree* or *try*.

**Note:** The terms CEF forwarding table, Forwarding Information Base (FIB) and CEF table are being used interchangeably for many CEF-related texts.



**Figure A2-7:** CEF Forwarding Table and Adjacency Table

- One of the biggest advantages of CEF is that the forwarding and adjacency tables are built without process switching – without waiting for a packet to be sent to a destination, in fact the tables are built before any packets are switched. Additionally, the forwarding table is built separately from the adjacency table – an error in one of the tables does not cause the other to become stale. When the ARP cache changes, only the adjacency table changes, therefore aging and invalidation of the forwarding table is no longer necessary. The CEF entries in the forwarding table never age, as they are linked directly to the routing table, changes upon the dynamic routing table are immediately propagated to the forwarding table.

- The adjacency table contains information other than MAC header and outbound interface as below:
  - **cached** – a prebuilt MAC header rewrite string and outbound interface used to reach a particular adjacent host or router.
  - **receive** – Packets destined to this IP address should be received by the router, which includes broadcast addresses and IP addresses configured on the router itself.
  - **drop** – Packets destined to this IP address should be dropped, which includes traffic denied by an access list, or routed to the NULL interface.
  - **punt** – Packets that Cisco Express Forwarding cannot switch and will be passed to the next best switching algorithm for processing – normally fast switching.
  - **glean** – The destination is directly connected, but there is no prebuilt MAC header rewrite string currently available. An ARP request must be sent to build the MAC header.
- Below shows how CEF build the appropriate adjacency table entry from the ARP cache information for a directly connected destination – 192.168.1.2 that is resolved to a glean adjacency.

```

RT1#sh ip cef detail
IP CEF with switching (Table Version 16), flags=0x0
  6 routes, 0 reresolve, 0 unresolved (0 old, 0 new), peak 2
  3 instant recursive resolutions, 0 used background process
  9 leaves, 11 nodes, 12664 bytes, 18 inserts, 9 invalidations
  0 load sharing elements, 0 bytes, 0 references
  universal per-destination load sharing algorithm, id B02A5381
  2(0) CEF resets, 0 revisions of existing leaves
  Resolution Timer: Exponential (currently 1s, peak 1s)
  1 in-place/0 aborted modifications
  refcounts: 3078 leaf, 3072 node

Table epoch: 0 (9 entries at this epoch)

0.0.0.0/0, version 0, epoch 0, attached, default route handler
0 packets, 0 bytes
  via 0.0.0.0, 0 dependencies
  valid no route adjacency
0.0.0.0/32, version 0, epoch 0, receive
192.168.1.0/24, version 5, epoch 0, attached, connected
0 packets, 0 bytes
  via FastEthernet1/0, 0 dependencies
  valid glean adjacency
192.168.1.0/32, version 3, epoch 0, receive
192.168.1.1/32, version 2, epoch 0, receive
192.168.1.255/32, version 4, epoch 0, receive
224.0.0.0/4, version 1, epoch 0
0 packets, 0 bytes, Precedence routine (0)
  via 0.0.0.0, 0 dependencies
  next hop 0.0.0.0
  valid drop adjacency
224.0.0.0/24, version 2, epoch 0, receive
255.255.255.255/32, version 1, epoch 0, receive
RT1#
RT1#debug ip cef table
IP CEF table debugging is on
RT1#
RT1#debug arp
ARP packet debugging is on
RT1#

```

## RT1#ping 192.168.1.2

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 192.168.1.2, timeout is 2 seconds:

```
00:10:10: IP ARP: creating incomplete entry for IP address: 192.168.1.2
interface FastEthernet1/0
00:10:10: IP ARP: sent req src 192.168.1.1 cc00.0550.0010,
dst 192.168.1.2 0000.0000.0000 FastEthernet1/0
00:10:10: IP ARP: rcvd rep src 192.168.1.2 cc02.0550.0000, dst 192.168.1.1
FastEthernet1/0
00:10:10: CEF-IP: Checking dependencies of 192.168.1.0/24
00:10:10: CEF-Table: Adjacency-prefix 192.168.1.2/32 add request --
succeeded.
00:10:12: IP ARP: rcvd req src 192.168.1.2 cc02.0550.0000, dst 192.168.1.1
FastEthernet1/0
00:10:12: IP ARP: sent rep src 192.168.1.1 cc00.0550.0010,
dst 192.168.1.2 cc02.0550.0000 FastEthernet1/0.!!!
Success rate is 60 percent (3/5), round-trip min/avg/max = 16/25/40 ms
RT1#
```

## RT1#sh ip cef detail

IP CEF with switching (Table Version 17), flags=0x0

```
7 routes, 0 reresolve, 0 unresolved (0 old, 0 new), peak 2
3 instant recursive resolutions, 0 used background process
10 leaves, 11 nodes, 12800 bytes, 19 inserts, 9 invalidations
0 load sharing elements, 0 bytes, 0 references
universal per-destination load sharing algorithm, id B02A5381
2(0) CEF resets, 0 revisions of existing leaves
Resolution Timer: Exponential (currently 1s, peak 1s)
1 in-place/0 aborted modifications
refcounts: 3080 leaf, 3072 node
```

Table epoch: 0 (10 entries at this epoch)

Adjacency Table has 1 adjacency

0.0.0.0/0, version 0, epoch 0, attached, default route handler

0 packets, 0 bytes

via 0.0.0.0, 0 dependencies

valid no route adjacency

0.0.0.0/32, version 0, epoch 0, receive

192.168.1.0/24, version 5, epoch 0, attached, connected

0 packets, 0 bytes

via FastEthernet1/0, 0 dependencies

valid glean adjacency

192.168.1.0/32, version 3, epoch 0, receive

192.168.1.1/32, version 2, epoch 0, receive

**192.168.1.2/32, version 16, epoch 0, connected, cached adjacency 192.168.1.2**

**0 packets, 0 bytes**

**via 192.168.1.2, FastEthernet1/0, 0 dependencies**

**next hop 192.168.1.2, FastEthernet1/0**

**valid cached adjacency**

192.168.1.255/32, version 4, epoch 0, receive

224.0.0.0/4, version 1, epoch 0

0 packets, 0 bytes, Precedence routine (0)

via 0.0.0.0, 0 dependencies

next hop 0.0.0.0

valid drop adjacency

224.0.0.0/24, version 2, epoch 0, receive

255.255.255.255/32, version 1, epoch 0, receive

RT1#

- The following packets are sent or punted to the CPU for process switching:
  - ARP requests and replies
  - IP packets that require a response from a router, eg: TTL has expired, MTU is exceeded, fragmentation is needed, etc.
  - IP broadcasts that will be relayed as unicast, eg: DHCP requests, IP helper address functions.
  - Routing protocol updates.
  - Cisco Discovery Protocol packets.
  - Packets requiring encryption.
  - Packets triggering Network Address Translation.
  - Other non-IP protocol packets, eg: IPX, AppleTalk, DECnet, etc.
  
- The version number indicates the number of times the CEF entry has been updated since the CEF table was generated. The epoch number indicates the number of times the CEF table has been flushed and regenerated as a whole.
  
- CEF supports load balancing over equal-cost paths. Load balancing at the switching level is far superior by load balancing via routing protocols, as routing protocols operate at a higher level. CEF load balancing is performed on a per-destination basis by default – all packets for a destination will use the same link. CEF can be configured to perform load balancing on a per-packet basis, for scenarios such as there is only a single host on the other end of the links, or there is a single server that consumes the majority of the bandwidth for any single link.
  
- Issue the **ip load-sharing [per-packet | per-destination]** interface subcommand to enables per-packet or per-destination CEF load balancing for an interface. When enabling per-packet load balancing to a particular destination, all interfaces that can forward traffic to the destination must be enabled for per-packet load balancing.  
**Note:** The **ip load-sharing per-destination** interface subcommand is the default configuration and is not shown in the configuration files.

## Cisco IOS Image Naming Convention, Packaging, and Deployment

- Cisco IOS software is a network system software that supports a broad range of Internet and enterprise network devices. The Internet was and is a driving force for the growth of Cisco IOS. Currently, Cisco IOS supports a broad area of networking technology, eg: IP, security, voice, VoIP, optical, wireless, content networking, and storage networking.

### Cisco IOS Trains

- A **Cisco IOS Train** is used to deliver Cisco IOS releases that evolve from a common code base.
- In recent years, with the addition of thousands of new features, hundreds of new applications, and a wide array of platforms, Cisco IOS software has diversified from one train to multiple trains of releases in order to support different feature sets for different customer needs.
- Below lists the types of Cisco IOS release trains:

<b>Train / Release</b>	<b>Description</b>	<b>Examples</b>
<b>Mainline</b>	Consolidates releases and fixes defects. Inherits features from previous generation T train, and does not add additional features.	12.2, 12.3, 12.4.
<b>T (Technology)</b>	Introduces new features and fixes defects. Provides support for new hardware.	12.2T, 12.3T.
<b>S</b>	Supports high-end backbone routing, and fixes defects.	12.0S, 12.2S.
<b>SB</b>	Targets at SP edge networks. Supports broadband and leased-line aggregation, and MPLS Provider Edge (PE).	12.2SB.
<b>SE, SG.</b>	Supports mid-range and low-end Ethernet LAN switching for Enterprise access and distribution networks, as well as mid-range and low-end SP Metro Ethernet networks.	12.2SE, 12.2SG.
<b>SR</b>	Targets at high-end SP Carrier Ethernet, Broadband Aggregation and Subscriber Services, Metro Ethernet, and MPLS Provider Edge (PE) networks.	12.2SR.
<b>SX</b>	Designed for at core and data center networks, enterprise campus, and service provider edge networks that require world-class IP and MPLS services. Supports high-end Ethernet LAN switching for access, distribution, core, and data center networks.	12.2SX.
<b>E</b>	Targets at enterprise core and SP edge networks. Supports advanced QoS, voice, security, and fixes defects.	12.1E.
<b>B</b>	Supports broadband features and fixes defects.	12.2B, 12.3B.

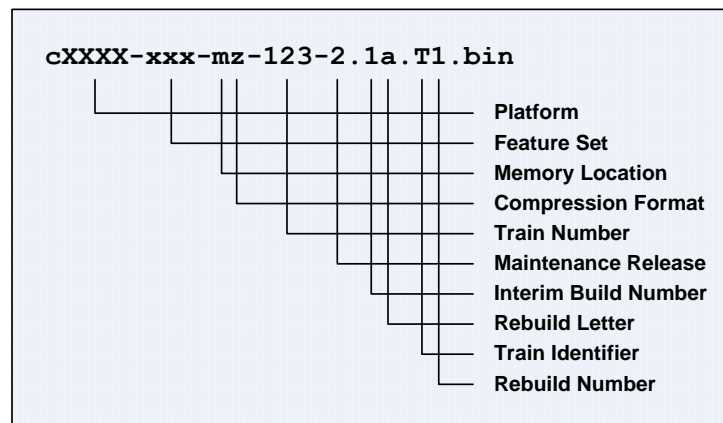


## Mainline and T Train Relationship

- Cisco uses at least the following 2 trains for releasing Cisco IOS software. With 2 trains, new features can be introduced in a release without affecting the code base of the mainline train.
  - i) Mainline train – A train for mainline bug fixes. Does not incorporate new features.
  - ii) T train – A train for synchronized mainline bug fixes and new features.

**Note:** Bug fixes to maintenance releases on a mainline train are synchronized with subsequent maintenance releases on the child T train.
- Mainline trains (eg: 12.2) are built from previous generation T trains (eg: 12.1T). Mainline trains (eg: 12.2) are parents of other T trains (eg: 12.2T). Ex: 12.1T → 12.2 → 12.2T → 12.3 → 12.3T.

## Cisco IOS Image Naming Convention



**Figure A3-1:** Cisco IOS Naming Convention

- Below lists the **Memory Location** codes:

Code	Description
f	The image runs from Flash memory.
m	The image runs from RAM.
r	The image runs from ROM.
l	The image is relocatable.

- Below lists the **Compression Format** codes:

Code	Description
z	The image is compressed in zip format.
x	The image is compressed in mzip format.

- **Interim Builds** are software from the Cisco IOS engineering build processes that the Cisco TAC released to customers to address a specific issue on temporary basis. Customers who have an interim build running on their network devices should contact Cisco TAC for assistance for replacing the interim build.
- An interim build name has a train number plus a maintenance release number and **decimal number in parentheses**, eg: 12.2(3.1).
- Unlike conventional computer software versioning rules, a 12.3 IOS release is not necessary newer than a 12.2 IOS release, eg: a maintenance release of 12.2T can be released few months later than a maintenance release of 12.3.

- Cisco IOS binary image name examples:

Image Name	Description
c3725-entbase-mz-123-2.T.bin	12.3(2)T release with the Enterprise Base Feature set for the Cisco 3725 platform.

## Cisco IOS Packaging

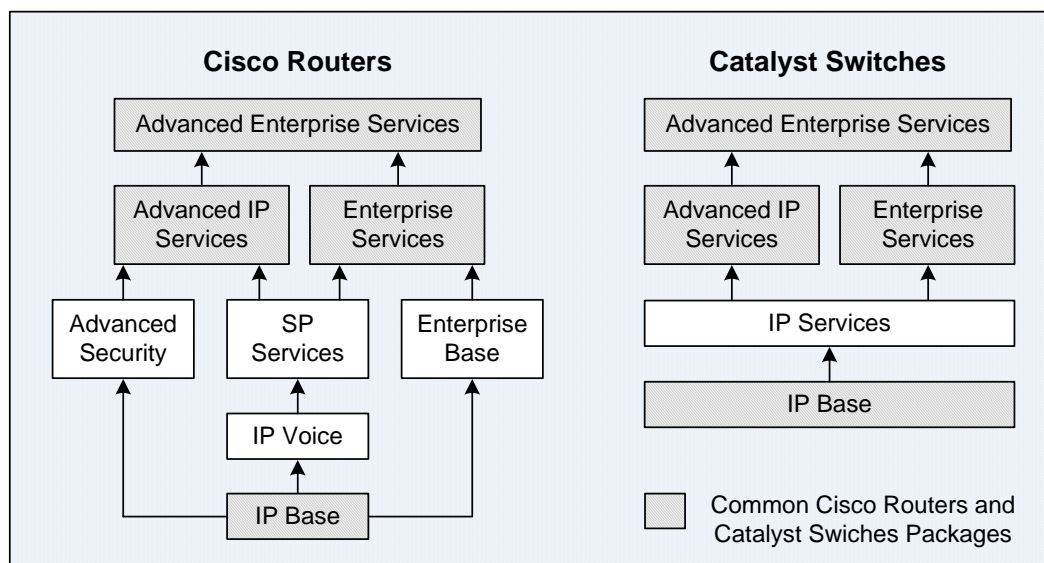


Figure A3-2: Cisco IOS Packaging

- Cisco IOS consists of 8 packages for Cisco routers and 5 packages for Catalyst switches.
- 4 packages are designed to satisfy the requirements of the following 4 service categories:
  - IP data
  - Converged voice and data
  - Security and VPN
  - Enterprise protocols
- 3 additional premium packages offer new IOS features that address more complex network requirements. All features merge in the Advanced Enterprise Services that integrates support for all routing protocols with voice, security, and VPN capabilities.
- **Feature Inheritance** → After a feature is introduced, it will be included in the more comprehensive packages. This clarifies the feature contents of different packages and the relationship between them, which facilitates IOS image and feature selection during migration.
- New Cisco IOS packaging naming convention:

<b>Base</b>	Entry-level image – IP Base, Enterprise Base.
<b>Services</b>	Addition of IP Telephony Service, MPLS, VoIP, VoFR, and ATM – SP Services, Enterprise Services.
<b>Advanced</b>	Addition of VPN, Cisco IOS Firewall, 3DES Encryption, SSH, IPSec and IDS – Advanced Security, Advanced IP Services.
<b>Enterprise</b>	Addition of multi-protocols, eg: IBM, IPX, AppleTalk – Enterprise Base, Enterprise Services.

## Cisco IOS Deployment

- **Early Deployment (ED)** releases provide new features and new platform and interface support in addition to bug fixes. There are 4 variations of the ED releases:
  - i) **Consolidated Technology Early Deployment (CTED)**. These are also known as the T train and are easily identifiable by their name, which always ends with a T (Technology), eg: 12.2T and 12.3T. T trains are very rich in features, protocols, and platforms support.
  - ii) **Specific Technology Early Deployment (STED)**. STEDs are usually platform-specific. Ex: 11.1CA, 11.1CC, 11.1CT, 11.3NA, 11.3MA, 11.3WA, and 12.0DA.
  - iii) **Specific Market Early Deployment (SMED)**. SMEDs are similar to STEDs, but they target specific market segments, eg: ISPs. Ex: 12.0S, and 12.1E.
  - iv) **Short-lived Early Deployment**, also known as **X Releases (XED)**. XEDs do not provide regular software maintenance revisions. If a bug is found in the XED before its convergence with the CTED, a software rebuild is initiated and a number is appended to the IOS name. Ex: 12.0(2)XB1 and 12.0(2)XB2 are the rebuilds of 12.0(2)XB.
  
- A **General Deployment (GD)** release is major Cisco IOS software that reaches its GD milestone in its lifecycle and Cisco feels that it is suitable for deployment anywhere in customer networks where the features and functionalities of the release are required. Criteria for reaching the GD milestone are based on, but not limited to: customer feedback surveys from production and test networks using the release, customer engineer bug reports, and reported field experience. Only major releases are candidates to reach the GD milestone.
  
- A **Limited Deployment (LD)** release is major Cisco IOS software that reaches the LD phase of its lifecycle during the period between its FCS (First Customer Shipment) and the GD milestone.
  
- **Note:** GD and LD releases are not applicable to any future Cisco IOS maintenance releases or rebuilds starting from Cisco IOS Software Release 12.4
  
- **Deferred (DF)** releases are not available for download due to known critical defects as announced by a Deferral Advisory. These releases should not be running on production routers and customers are strongly urged to migrate from the affected image to the replacement image as advised in the Deferral Advisory.
  
- **Maintenance Deployment (MD)** releases provide bug fixes and ongoing software maintenance.

# ICMPv4 and ICMPv6 Type Numbers and Code Fields

## ICMPv4 Type Numbers and Code Fields

Type	Name	
<b>0</b>	Code 0	<b>Echo Reply</b> (RFC 792 – Internet Control Message Protocol)
<b>1</b>	<b>Unassigned</b>	
<b>2</b>	<b>Unassigned</b>	
<b>3</b>	<b>Destination Unreachable</b> (RFC 792 – Internet Control Message Protocol)	
	Code 0	Network Unreachable
	Code 1	Host Unreachable
	Code 2	Protocol Unreachable
	Code 3	Port Unreachable
	Code 4	Fragmentation Needed and DF bit set (DF = Don't Fragment) Can't Fragment
	Code 5	Source Route Failed
	Code 6	Destination Network Unknown
	Code 7	Destination Host Unknown
	Code 8	Source Host Isolated
	Code 9	Communication with Destination Network is Administratively Prohibited
	Code 10	Communication with Destination Host is Administratively Prohibited
	Code 11	Destination Network Unreachable for Type of Service
	Code 12	Destination Host Unreachable for Type of Service
	Code 13	Communication Administratively Prohibited (RFC 1812 – Requirements for IPv4 Routers)
	Code 14	Host Precedence Violation (RFC 1812 – Requirements for IPv4 Routers)
	Code 15	Precedence cutoff in effect (RFC 1812 – Requirements for IPv4 Routers)
<b>4</b>	Code 0	<b>Source Quench</b> (RFC 792 – Internet Control Message Protocol)
<b>5</b>	<b>Redirect</b> (RFC 792 – Internet Control Message Protocol)	
	Code 0	Redirect Datagram for the Network
	Code 1	Redirect Datagram for the Host
	Code 2	Redirect Datagram for the Type of Service and Network
	Code 3	Redirect Datagram for the Type of Service and Host
<b>6</b>	Code 0	Alternate Address for Host
<b>7</b>	<b>Unassigned</b>	
<b>8</b>	Code 0	<b>Echo Request</b> (RFC 792 – Internet Control Message Protocol)
<b>9</b>	<b>Router Advertisement</b> (RFC 1256 – ICMP Router Discovery Messages)	
	Code 0	Normal Router Advertisement
	Code 16	Does not Route Common Traffic
<b>10</b>	<b>Router Solicitation</b> (RFC 1256 – ICMP Router Discovery Messages)	
	Code 0	No Code
<b>11</b>	<b>Time Exceeded</b> (RFC 792 – Internet Control Message Protocol)	
	Code 0	Time to Live Exceeded in Transit
	Code 1	Fragment Reassembly Time Exceeded
<b>12</b>	<b>Parameter Problem</b> (RFC792 – Internet Control Message Protocol)	
	Code 0	Pointer Indicates the Error
	Code 1	Missing a Required Option
	Code 2	Bad Length
<b>13</b>	Code 0	<b>Timestamp Request</b> (RFC792 – Internet Control Message Protocol)

<b>14</b>	Code 0	<b>Timestamp Reply</b> (RFC792 – Internet Control Message Protocol)
<b>15</b>	Code 0	<b>Information Request</b> (RFC792 – Internet Control Message Protocol)
<b>16</b>	Code 0	<b>Information Reply</b> (RFC792 – Internet Control Message Protocol)
<b>17</b>	Code 0	<b>Address Mask Request</b> (RFC 950 – Internet Standard Subnetting Procedure)
<b>18</b>	Code 0	<b>Address Mask Reply</b> (RFC 950 – Internet Standard Subnetting Procedure)
<b>19</b>	<b>Reserved for Security</b>	
<b>20 – 29</b>	<b>Reserved for Robustness Experiment</b>	
<b>30</b>	<b>Traceroute</b> (RFC 1393 – Traceroute Using an IP Option)	
<b>31</b>	<b>Datagram Conversion Error</b> (RFC 1475 – TP/IX: The Next Internet)	
<b>32</b>	<b>Mobile Host Redirect</b>	
<b>33</b>	<b>IPv6 Where-Are-You</b>	
<b>34</b>	<b>IPv6 I-Am-Here</b>	
<b>35</b>	<b>Mobile Registration Request</b>	
<b>36</b>	<b>Mobile Registration Reply</b>	
<b>39</b>	<b>SKIP</b>	
<b>40</b>	<b>Photuris</b> (RFC 2521 – ICMP Security Failures Messages)	
	Code 0	Bad SPI
	Code 1	Authentication Failed
	Code 2	Decompression Failed
	Code 3	Decryption Failed
	Code 4	Need Authentication
	Code 5	Need Authorization
<b>41</b>	ICMP messages utilized by experimental mobility protocols (RFC 4065 – Instructions for Seamoby and Experimental Mobility Protocol)	
<b>42 – 255</b>	<b>Reserved</b>	

## ICMPv6 Type Numbers and Code Fields

Type	Name	
<b>1</b>	<b>Destination Unreachable</b> (RFC 2463 – ICMPv6 for the IPv6 Specification)	
	Code 0	No route to destination
	Code 1	Communication with Destination is Administratively Prohibited
	Code 2	Beyond Scope of Source Address (RFC 4443 – ICMPv6 for the IPv6 Specification)
	Code 3	Address Unreachable
	Code 4	Port Unreachable
	Code 5	Source Address Failed Ingress / Egress Policy (RFC 4443 – ICMPv6 for the IPv6 Specification)
	Code 6	Reject Route to Destination (RFC 4443 – ICMPv6 for the IPv6 Spec.)
<b>2</b>	Code 0	<b>Packet Too Big</b> (RFC 2463 – ICMPv6 for the IPv6 Specification)
<b>3</b>	<b>Time Exceeded</b> (RFC 2463 – ICMPv6 for the IPv6 Specification)	
	Code 0	Hop Limit Exceeded in Transit
	Code 1	Fragment Reassembly Time Exceeded
<b>4</b>	<b>Parameter Problem</b> (RFC 2463 – ICMPv6 for the IPv6 Specification)	
	Code 0	Erroneous Header Field Encountered
	Code 1	Unrecognized Next Header Type Encountered
	Code 2	Unrecognized IPv6 Option Encountered
<b>128</b>	Code 0	<b>Echo Request</b> (RFC 2463 – ICMPv6 for the IPv6 Specification)

<b>129</b>	Code 0	<b>Echo Reply</b> (RFC 2463 – ICMPv6 for the IPv6 Specification)
<b>130</b>	Code 0	<b>Multicast Listener Query</b> (RFC 2710 – Multicast Listener Discovery (MLD) for IPv6)
<b>131</b>	Code 0	<b>Multicast Listener Report</b> (RFC 2710 – Multicast Listener Discovery (MLD) for IPv6)
<b>132</b>	Code 0	<b>Multicast Listener Done</b> (RFC 2710 – Multicast Listener Discovery (MLD) for IPv6)
<b>133</b>	Code 0	<b>Router Solicitation</b> (RFC 2461 – Neighbor Discovery for IPv6)
<b>134</b>	Code 0	<b>Router Advertisement</b> (RFC 2461 – Neighbor Discovery for IPv6)
<b>135</b>	Code 0	<b>Neighbor Solicitation</b> (RFC 2461 – Neighbor Discovery for IPv6)
<b>136</b>	Code 0	<b>Neighbor Advertisement</b> (RFC 2461 – Neighbor Discovery for IPv6)
<b>137</b>	Code 0	<b>Redirect Message</b> (RFC 2461 – Neighbor Discovery for IPv6)
<b>138</b>	<b>Router Renumbering</b>	
	Code 0	Router Renumbering Command
	Code 1	Router Renumbering Result
	Code 255	Sequence Number Reset
<b>139</b>	<b>ICMP Node Information Query</b>	
	Code 0	The Data field contains an IPv6 address which is the Subject of this Query.
	Code 1	The Data field contains a name which is the Subject of this Query, or is empty, as in the case of a NOOP.
	Code 2	The Data field contains an IPv4 address which is the Subject of this Query.
<b>140</b>	<b>ICMP Node Information Response</b>	
	Code 0	A successful reply. The Reply Data field may or may not be empty.
	Code 1	The Responder refuses to supply the answer. The Reply Data field will be empty.
	Code 2	The Qtype of the Query is unknown to the Responder. The Reply Data field will be empty.
<b>141</b>	Code 0	<b>Inverse Neighbor Discovery Solicitation Message</b> (RFC 3122 – Extensions to IPv6 Neighbor Discovery for Inverse Discovery)
<b>142</b>	Code 0	<b>Inverse Neighbor Discovery Advertisement Message</b> (RFC 3122 – Extensions to IPv6 Neighbor Discovery for Inverse Discovery)
<b>144</b>	Code 0	<b>Home Agent Address Discovery Request Message</b> (RFC 3775 – Mobility Support in IPv6)
<b>145</b>	Code 0	<b>Home Agent Address Discovery Reply Message</b> (RFC 3775 – Mobility Support in IPv6)
<b>146</b>	Code 0	<b>Mobile Prefix Solicitation</b> (RFC 3775 – Mobility Support in IPv6)
<b>147</b>	Code 0	<b>Mobile Prefix Advertisement</b> (RFC 3775 – Mobility Support in IPv6)

## Appendix 5 Netmask Table

Prefix Length	Dotted Decimal	Hexadecimal	Binary
/0	0.0.0.0	0x00000000	00000000 00000000 00000000 00000000
/1	128.0.0.0	0x80000000	10000000 00000000 00000000 00000000
/2	192.0.0.0	0xc0000000	11000000 00000000 00000000 00000000
/3	224.0.0.0	0xe0000000	11100000 00000000 00000000 00000000
/4	240.0.0.0	0xf0000000	11110000 00000000 00000000 00000000
/5	248.0.0.0	0xf8000000	11111000 00000000 00000000 00000000
/6	252.0.0.0	0xfc000000	11111100 00000000 00000000 00000000
/7	254.0.0.0	0xfe000000	11111110 00000000 00000000 00000000
/8	255.0.0.0	0xff000000	11111111 00000000 00000000 00000000
/9	255.128.0.0	0xff800000	11111111 10000000 00000000 00000000
/10	255.192.0.0	0xffc00000	11111111 11000000 00000000 00000000
/11	255.224.0.0	0xffe00000	11111111 11100000 00000000 00000000
/12	255.240.0.0	0xffff0000	11111111 11110000 00000000 00000000
/13	255.248.0.0	0xffff8000	11111111 11111000 00000000 00000000
/14	255.252.0.0	0xffffc000	11111111 11111100 00000000 00000000
/15	255.254.0.0	0xffffe000	11111111 11111110 00000000 00000000
/16	255.255.0.0	0xfffff000	11111111 11111111 00000000 00000000
/17	255.255.128.0	0xfffff800	11111111 11111111 10000000 00000000
/18	255.255.192.0	0xfffffc00	11111111 11111111 11000000 00000000
/19	255.255.224.0	0xfffffe00	11111111 11111111 11100000 00000000
/20	255.255.240.0	0xffffff00	11111111 11111111 11110000 00000000
/21	255.255.248.0	0xffffff80	11111111 11111111 11111000 00000000
/22	255.255.252.0	0xffffffc0	11111111 11111111 11111100 00000000
/23	255.255.254.0	0xffffffe0	11111111 11111111 11111110 00000000
/24	255.255.255.0	0xfffffff0	11111111 11111111 11111111 00000000
/25	255.255.255.128	0xfffffff8	11111111 11111111 11111111 10000000
/26	255.255.255.192	0xffffffc0	11111111 11111111 11111111 11000000
/27	255.255.255.224	0xffffffe0	11111111 11111111 11111111 11100000
/28	255.255.255.240	0xfffffff0	11111111 11111111 11111111 11110000
/29	255.255.255.248	0xfffffff8	11111111 11111111 11111111 11111000
/30	255.255.255.252	0xffffffc0	11111111 11111111 11111111 11111100
/31	255.255.255.254	0xffffffe0	11111111 11111111 11111111 11111110
/32	255.255.255.255	0xfffffff	11111111 11111111 11111111 11111111

## CCNP ROUTE Extra Knowledge

Topic	Page
Address Space Usage Calculation	364
Head-of-Line (HOL) Blocking	364
NAT Timeout	365
Traceroute Output Characters	366
Network Address Translation – Protocol Translation (NAT-PT)	368
IPv6 Ping Output Characters	378
The <b>ip default-network</b> Global Configuration Command	378
IP Unnumbered Serial Interfaces	384
DHCP – Dynamic Host Configuration Protocol	389
Easy IP	393
IP Helper Addresses	394
Directed Broadcast Forwarding	396
Bridging Cisco Router Interfaces for High Availability LAN	400
The Pitfall of Bridge Group	402
EIGRP Metric Calculation	405
The Impact of Minimum Bandwidth upon EIGRP Metric Calculation	407
EIGRP Feasible Successor	408
EIGRP DUAL Query-Reply Process	413
The EIGRP DUAL Finite State Machine	416
EIGRP Offset List	417
EIGRP Debug Commands	422
Duplicate EIGRP Router IDs Preventing Installation of EIGRP External Route	426
The OSPF Options Field	429
The OSPF Neighbor State Machine	431
OSPF TOS-based Routing	433
Updating the OSPF Link-State Database	434
The <b>show ip ospf interface EXEC</b> Command	435
OSPF Neighbor Relationship over Non-compatible OSPF Network Types	437
OSPF Point-to-Multipoint Network Type and /32 Routes	437
Discontiguous Non-Zero OSPF Areas	438
OSPF Type-3 Network-Summary-LSAs Generations and Deletions	442
OSPF Link-State Database Overload Protection	445
OSPF External Route Default Metrics	446
OSPF Forward Metric in External Type-2 (E2) Route Selection	447
OSPF Route Filtering within an Area	448
The <b>area area-id nssa no-redistribution</b> OSPF Router Subcommand	453
When OSPF Inter-Area Routing Bypasses the Backbone Area	455
OSPF Stub Areas Attack Session	456
Suboptimal Routing when Redistributing between OSPF Processes	457
DECnet Inter-Phase IV-Area Routing through DECnet Phase V Routing (IS-IS)	459
Pure CLNS Routing Network	461
Why IS-IS?	464
The Impact of the <b>clear ip route</b> Privileged Command	465
Using Extended Access Lists with Route Maps and Distribute Lists	470
Administrative Distance Load Balancing	472
The <b>ip route-cache policy</b> Interface Subcommand	474



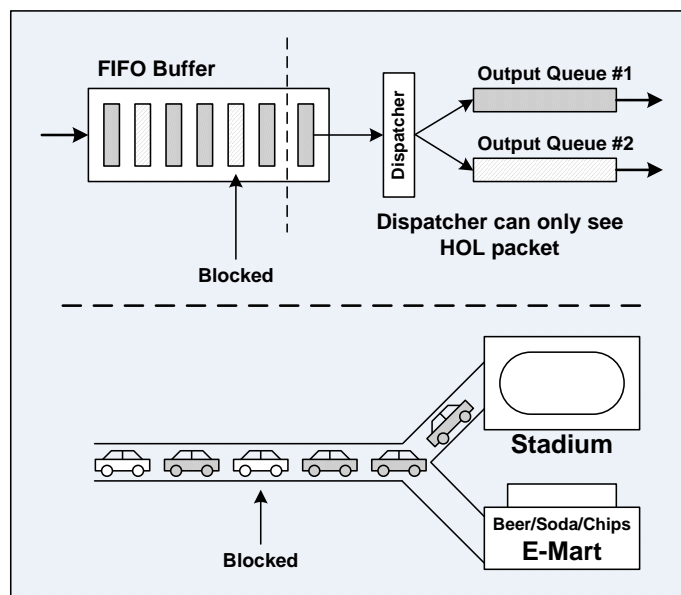




## Address Space Usage Calculation

- **Example 1:**  
0000 – 4 bits are able to provide **16** addresses.  
Type I address uses the prefix 01. Addresses with the prefix 01 are **0100, 0101, 0110, 0111**.  
There are **4** ( $2^2$ ) addresses with the prefix 01. Type I address uses  $4/16 = 1/4$  ( $2^2$ ) of the total address space.
- **Example 2:**  
00000 – 5 bits are able to provide **32** addresses.  
Type II address uses the prefix 01. There are **8** ( $2^3$ ) addresses with the prefix 01.  
Type II address uses  $8/32 = 1/4$  ( $2^2$ ) of the total address space.
- **Example 3:**  
00000 – 5 bits are able to provide **32** addresses.  
Type III address uses the prefix 001. There are **4** ( $2^4$ ) addresses with the prefix 001.  
Type III address uses  $4/32 = 1/8$  ( $2^3$ ) of the total address space.
- The conclusion is, regardless of the number of bits for the address space, the denominator for the  $1/x$  formula is always **2 to the power of the number of prefix bits**.  
The IPv6 Unique-Local unicast address uses a prefix of FD00::/8, hence the IPv6 unique-local unicast address range uses  $1/256$  ( $2^8$ ) of the total IPv6 address space.  
**Note:** Numerator and denominator are the numbers on top and bottom of a fraction respectively.  
Ex: In  $\frac{3}{4}$ , 3 is the numerator and 4 is the denominator.

## Head-of-Line (HOL) Blocking



**Figure A6-1:** Head-of-Line (HOL) Blocking

- The figure above shows a scenario of HOL blocking. Output Queue #1 is busy while Output Queue #2 is idle. The 2nd packet in the FIFO buffer could be sent to the Output Queue #2. However, only the 1st packet can be accessed in a FIFO buffer. Hence, the HOL packet can block packets behind it that are destined to other idle output queues.
- Another scenario is autos going to E-Mart are forced to wait for congested stadium traffic to clear.

## NAT Timeout

- The **ip nat translation {dns-timeout | finrst-timeout | icmp-timeout | port-timeout | pptp-timeout | syn-timeout | tcp-timeout | timeout | udp-timeout} {sec | never}** global configuration command changes the amount of time which Network Address Translation (NAT) translations time out.

```
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#ip nat translation ?
  dns-timeout      Specify timeout for NAT DNS flows
  finrst-timeout   Specify timeout for NAT TCP flows after a FIN or RST
  icmp-timeout     Specify timeout for NAT ICMP flows
  max-entries      Specify maximum number of NAT entries
  port-timeout     Specify timeout for NAT TCP/UDP port specific flows
  pptp-timeout     Specify timeout for NAT PPTP flows
  syn-timeout      Specify timeout for NAT TCP flows after a SYN and no
                  further data
  tcp-timeout      Specify timeout for NAT TCP flows
  timeout          Specify timeout for dynamic NAT translations
  udp-timeout      Specify timeout for NAT UDP flows

Router(config)#
```

- Below lists the defaults value for the corresponding NAT timeouts:

<b>dns-timeout</b>	60 seconds (1 minute)
<b>finrst-timeout</b>	60 seconds (1 minute)
<b>icmp-timeout</b>	60 seconds (1 minute)
<b>pptp-timeout</b>	86400 seconds (24 hours)
<b>syn-timeout</b>	60 seconds (1 minute)
<b>tcp-timeout</b>	86400 seconds (24 hours)
<b>timeout</b>	86400 seconds (24 hours)
<b>udp-timeout</b>	300 seconds (5 minutes)

- Below shows that the NAT timeout for an ICMP session is 1 minute:

```
Router#sh ip nat translations
Pro Inside global      Inside local      Outside local      Outside global
--- 100.100.100.21     192.168.2.21     ---                ---
--- 100.100.100.22     192.168.2.22     ---                ---
Router#
Router#sh ip nat translations
Pro Inside global      Inside local      Outside local      Outside global
icmp 100.100.100.21:6 192.168.2.21:6   172.16.2.21:6    172.16.2.21:6
--- 100.100.100.21     192.168.2.21     ---                ---
--- 100.100.100.22     192.168.2.22     ---                ---
Router#
00:51:44: NAT*: s=192.168.2.21->100.100.100.21, d=172.16.2.21 [30]
00:51:44: NAT*: s=172.16.2.21, d=100.100.100.21->192.168.2.21 [30]
00:52:44: NAT: expiring 100.100.100.21 (192.168.2.21) icmp 6 (6)
Router#
```

## Traceroute Output Characters

- The following characters can be displayed as output for the **traceroute** command:

Character	Description
nn msec	For each node, the round-trip time in milliseconds for the specified number of probes
*	The probe timed out
A	Administratively prohibited (eg: access lists)
Q	Source quench (destination too busy)
I	User interrupted test
U	Port unreachable
H	Host unreachable
N	Network unreachable
P	Protocol unreachable
T	Timeout
?	Unknown packet type

- Below shows the output of the **traceroute** command in which 11.11.11.11 is an invalid IP address in the sample network and another router (3.3.3.2) has a black hole Null0 route to 0.0.0.0/0.

```

Router#trace 11.11.11.11

Type escape sequence to abort.
Tracing the route to 11.11.11.11

 1 2.2.2.2 40 msec 68 msec 40 msec
 2 3.3.3.2 92 msec 24 msec 164 msec
 3 3.3.3.2 !H * !H
Router#
Router#debug ip icmp
ICMP packet debugging is on
Router#
Router#trace 11.11.11.11

Type escape sequence to abort.
Tracing the route to 11.11.11.11

 1 2.2.2.2 40 msec 16 msec 40 msec
 2 3.3.3.2 108 msec 72 msec 56 msec
 3 3.3.3.2 !H
00:27:17: ICMP: time exceeded rcvd from 2.2.2.2
00:27:17: ICMP: time exceeded rcvd from 2.2.2.2
00:27:17: ICMP: time exceeded rcvd from 2.2.2.2
00:27:17: ICMP: time exceeded rcvd from 3.3.3.2
00:27:17: ICMP: time exceeded rcvd from 3.3.3.2
00:27:17: ICMP: time exceeded rcvd from 3.3.3.2
00:27:18: ICMP: dst (2.2.2.1) host unreachable rcv from 3.3.3.2 * !H
00:27:21: ICMP: dst (2.2.2.1) host unreachable rcv from 3.3.3.2
Router#

```

- Below shows the output of the **tracert** command in which RT2 is configured with an access control list that blocked the UDP traceroute packets initiated by RT1:

```

RT1#ping 10.10.10.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.10.2, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 24/43/68 ms
RT1#
=====
RT2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT2(config)#access-list 1 permit 1.1.1.1
RT2(config)#int s0/0
RT2(config-if)#ip access-group 1 in
RT2(config-if)#
=====
RT1#debug ip icmp
ICMP packet debugging is on
RT1#
RT1#ping 10.10.10.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.10.2, timeout is 2 seconds:
U
00:03:25.383: ICMP: dst (10.10.10.1) administratively prohibited unreachable
rcv from 10.10.10.2.U
00:03:27.431: ICMP: dst (10.10.10.1) administratively prohibited unreachable
rcv from 10.10.10.2.U
Success rate is 0 percent (0/5)
RT1#
00:03:29.467: ICMP: dst (10.10.10.1) administratively prohibited unreachable
rcv from 10.10.10.2
RT1#
RT1#sh run | in 0.0.0.0
ip route 0.0.0.0 0.0.0.0 10.10.10.2
RT1#
RT1#trace 172.16.1.1

Type escape sequence to abort.
Tracing the route to 172.16.1.1

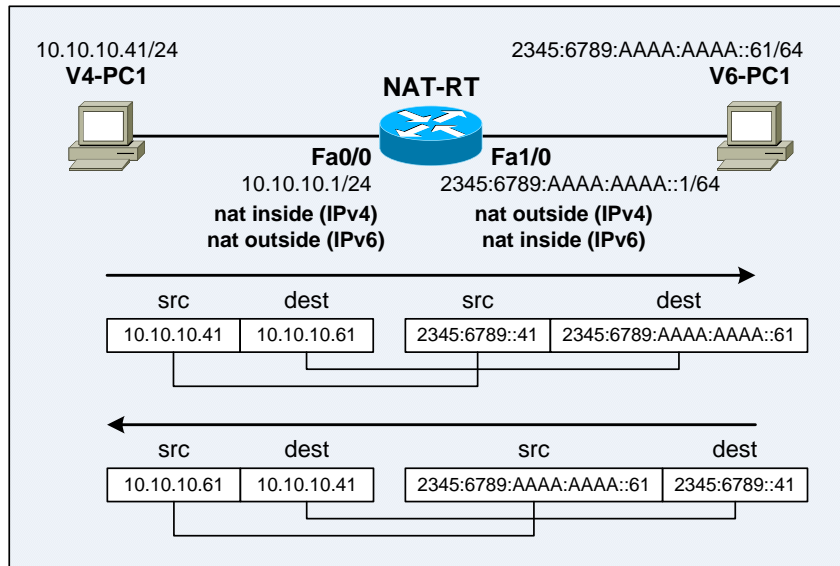
  1 10.10.10.2  !A
00:05:32.071: ICMP: dst (10.10.10.1) administratively prohibited unreachable
rcv from 10.10.10.2 *  !A
RT1#
00:05:35.119: ICMP: dst (10.10.10.1) administratively prohibited unreachable
rcv from 10.10.10.2
RT1#

```

## Network Address Translation – Protocol Translation (NAT-PT)

- **Network Address Translation – Protocol Translation (NAT-PT)** is often being used in IPv4 to IPv6 migration scenarios to provide bi-directional communications between IPv4 and IPv6 hosts. A dual-stack router is required to perform NAT-PT. Additionally, NAT-PT must be enabled on both the incoming and outgoing interfaces for NAT-PT to be operational.
- The concept and configuration of NAT-PT is much more difficult than traditional IPv4 NAT. The main difference between NAT-PT and traditional IPv4 NAT is that address translations must be performed both ways (**Bidirectional NAT**). IPv6 packets sent towards IPv4 hosts must have their source and destination IP addresses replaced with some IPv4 addresses; and vice versa. **Note:** Traditional Outbound NAT is designed to handle outbound connections, in which clients of the inside local network initiate requests to outside global Internet hosts. Bidirectional NAT, 2-Way NAT, or Inbound NAT is an enhancement upon NAT to handle connections initiated from the outside network.
- Cisco supports the following types of NAT-PT:

<b>Static NAT-PT</b>	Uses static translation rules to map 1 IPv6 address to 1 IPv4 address. Multiple static NAT-PT mappings are required when there are many IPv6-only or IPv4-only hosts that need to communicate.
<b>Dynamic NAT-PT</b>	Allows multiple NAT-PT mappings by allocating addresses from a pool. NAT-PT is configured with a pool of IPv6 and/or IPv4 addresses. The number of addresses available in the address pool determines the maximum number of concurrent sessions. The NAT-PT router would record all mapping of addresses in the dynamic translation table.
<b>Port Address Translation (Overload)</b>	Allows a single IPv4 address to be associated to multiple IPv6 hosts by multiplexing the transport layer port numbers. PAT can be performed through an interface or an address pool.
<b>IPv4-Mapped</b>	Allows traffic from an IPv6 network be sent to an IPv4 network without configuring IPv6 destination address mapping. A packet arriving at an interface is being checked upon a NAT-PT prefix configured with the <b>ipv6 nat prefix ipv6-prefix v4-mapped {acl-name   ipv6-prefix}</b> global configuration command or interface subcommand. If the prefix matches, then an access list check is performed to determine whether the source address of the packet matches the access list or prefix list. If the prefix does not match, the packet is being dropped; if the prefix matches, source address translation is performed using the configured static IPv4 address or IPv4 address pool; while destination address translation is performed using the last 32 bits of the destination IPv6 address as the destination IPv4 address and a NAT entry will be created.



**Figure A6-2:** Network Setup for NAT-PT

- NAT-RT is configured to perform static bi-directional mappings to translate the destination addresses in IPv6 packets sent to 2345:6789::41 to 10.10.10.41; and translate the destination addresses in IPv4 packets sent to 10.10.10.61 to 2345:6789:AAAA:AAAA::61. As a result, IPv4 packets with source / destination pair (10.10.10.41 – 10.10.10.61) will be translated to IPv6 packets with address pair (2345:6789::41 – 2345:6789:AAAA:AAAA::61); while IPv6 packets with source / destination pair (2345:6789:AAAA:AAAA::61 – 2345:6789::41) will be translated to IPv4 packets with address pair (10.10.10.61 – 10.10.10.41).
- Below shows the configuration on NAT-RT to perform Static NAT-PT:

```

!
ipv6 unicast-routing
!
interface FastEthernet0/0
 ip address 10.10.10.61 255.255.255.0 secondary
 ip address 10.10.10.1 255.255.255.0
 ipv6 nat
!
interface FastEthernet1/0
 no ip address
 ipv6 address 2345:6789:AAAA:AAAA::1/64
 ipv6 nat
!
ipv6 nat v4v6 source 10.10.10.41 2345:6789::41
ipv6 nat v6v4 source 2345:6789:AAAA:AAAA::61 10.10.10.61
ipv6 nat prefix 2345:6789::/96
!

```

- The **ipv6 nat v4v6 source** {*ipv4-inside-local ipv6-outside-global* | **list** {*acl-num* | *acl-name*} **pool** *pool-name*} global configuration command configures IPv4 to IPv6 NAT-PT.
- The **ipv6 nat v6v4 source** {*ipv6-inside-local ipv4-outside-global* | **list** {*acl-name* **pool** *pool-name* | **route-map** *map-name* **pool** *pool-name*} [**overload**] global configuration command configures IPv6 to IPv4 NAT-PT.



- Below shows the installation of a directly connected IPv6 route to the IPv6 NAT prefix on NAT-RT:

```

NAT-RT#debug ipv6 routing
IPv6 routing table events debugging is on
NAT-RT#
NAT-RT#conf t
Enter configuration commands, one per line. End with CNTL/Z.
NAT-RT(config)#ipv6 nat prefix 2345:6789::/96
NAT-RT(config)#
00:02:34: IPv6RT0: connected, Add 2345:6789::/96 to table
00:02:34: IPv6RT0: connected, Adding next-hop :: over Null0 for
2345:6789::/96, [0/0]
NAT-RT(config)#
NAT-RT(config)#^Z
NAT-RT#
NAT-RT#sh ipv6 route connected
IPv6 Routing Table - 5 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
C    2345:6789::/96 [0/0]
     via ::, Null0
C    2345:6789:AAAA:AAAA::/64 [0/0]
     via ::, FastEthernet1/0
NAT-RT#

```

- IPv6 stack classifies packets for NAT-PT via a special IPv6 NAT prefix, which represents the whole IPv4 address space ( $2^{32}$ ) embedded within the IPv6 super-space, and always has the length of 96 bits ( $128 - 32 = 96$ ). Every IPv6 packet with the destination address matching the NAT-PT prefix will be translated by NAT-PT to an IPv4 packet using the configured mapping rules. The IPv6 NAT prefix can be configured globally for all interfaces or with different NAT-PT prefixes on a per interface basis with the **ipv6 nat prefix *ipv6-prefix/96*** global configuration command or interface subcommand (configured on an IPv6 interface). Using different NAT-PT prefixes on multiple interfaces allows a NAT-PT router to support an IPv6 network with multiple exit points to IPv4 networks.

- Below shows the output of **debug ip icmp** privileged command on V4-PC1 when V4-PC1 issued ping to 10.10.10.61, the IPv4 address associated to 2345:6789:AAAA:AAAA::61:

```

V4-PC1#debug ip icmp
ICMP packet debugging is on
V4-PC1#
V4-PC1#ping 10.10.10.61

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.10.61, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 44/76/112 ms
V4-PC1#
00:03:39: ICMP: echo reply rcvd, src 10.10.10.61, dst 10.10.10.41
00:03:39: ICMP: echo reply rcvd, src 10.10.10.61, dst 10.10.10.41
00:03:39: ICMP: echo reply rcvd, src 10.10.10.61, dst 10.10.10.41
00:03:39: ICMP: echo reply rcvd, src 10.10.10.61, dst 10.10.10.41
V4-PC1#

```

- Below shows that an IPv6 default route is required on V6-PC1 to reach 2345:6789::41; the output of the **debug ipv6 icmp** privileged command on V6-PC1 when V4-PC1 issued ping to 10.10.10.61, as well as V6-PC1 access to V4-PC1 using 2345:6789::41:

```

V6-PC1#sh ipv6 route 2345:6789::41
% Route not found
V6-PC1#
V6-PC1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
V6-PC1(config)#ipv6 route 0::/0 2345:6789:AAAA:AAAA::1
V6-PC1(config)#^Z
V6-PC1#
V6-PC1#sh ipv6 route 2345:6789::41
IPv6 Routing Table - 5 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
        U - Per-user Static route
        I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
        O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
        ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
S   ::/0 [1/0]
    via 2345:6789:AAAA:AAAA::1
V6-PC1#
V6-PC1#debug ipv6 icmp
ICMP packet debugging is on
V6-PC1#
00:03:33: ICMPv6: Received ICMPv6 packet from FE80::204:4EFF:FE11:1111, type
134
00:03:42: ICMPv6: Received ICMPv6 packet from 2345:6789::41, type 128
00:03:42: ICMPv6: Received echo request from 2345:6789::41
00:03:42: ICMPv6: Sending echo reply to 2345:6789::41
00:03:42: ICMPv6: Received ICMPv6 packet from 2345:6789::41, type 128
00:03:42: ICMPv6: Received echo request from 2345:6789::41
00:03:42: ICMPv6: Sending echo reply to 2345:6789::41
00:03:43: ICMPv6: Received ICMPv6 packet from 2345:6789::41, type 128
00:03:43: ICMPv6: Received echo request from 2345:6789::41
00:03:43: ICMPv6: Sending echo reply to 2345:6789::41
00:03:43: ICMPv6: Received ICMPv6 packet from 2345:6789::41, type 128
00:03:43: ICMPv6: Received echo request from 2345:6789::41
00:03:43: ICMPv6: Sending echo reply to 2345:6789::41
V6-PC1#
V6-PC1#telnet 2345:6789::41
Trying 2345:6789::41 ... Open

User Access Verification

Password:
V4-PC1>

```

- Below shows the output of the **debug ipv6 nat** privileged command on NAT-RT when V4-PC1 issued a ping to 10.10.10.61, the IPv4 address associated to 2345:6789:AAAA:AAAA::61:

```

NAT-RT#debug ipv6 nat
IPv6 NAT-PT debugging is on
NAT-RT#
00:03:38: IPv6 NAT: icmp src (10.10.10.41) -> (2345:6789::41), dst
(10.10.10.61) -> (2345:6789:AAAA:AAAA::61)
00:03:38: IPv6 NAT: icmp src (2345:6789:AAAA:AAAA::61) -> (10.10.10.61), dst
(2345:6789::41) -> (10.10.10.41)
00:03:38: IPv6 NAT: icmp src (10.10.10.41) -> (2345:6789::41), dst
(10.10.10.61) -> (2345:6789:AAAA:AAAA::61)
00:03:38: IPv6 NAT: icmp src (2345:6789:AAAA:AAAA::61) -> (10.10.10.61), dst
(2345:6789::41) -> (10.10.10.41)
00:03:38: IPv6 NAT: icmp src (10.10.10.41) -> (2345:6789::41), dst
(10.10.10.61) -> (2345:6789:AAAA:AAAA::61)
00:03:38: IPv6 NAT: icmp src (2345:6789:AAAA:AAAA::61) -> (10.10.10.61), dst
(2345:6789::41) -> (10.10.10.41)
00:03:38: IPv6 NAT: icmp src (10.10.10.41) -> (2345:6789::41), dst
(10.10.10.61) -> (2345:6789:AAAA:AAAA::61)
00:03:38: IPv6 NAT: icmp src (2345:6789:AAAA:AAAA::61) -> (10.10.10.61), dst
(2345:6789::41) -> (10.10.10.41)
NAT-RT#
NAT-RT#sh ipv6 nat translations
Prot  IPv4 source          IPv6 source
      IPv4 destination    IPv6 destination
---   ---                ---
      10.10.10.41         2345:6789::41

---   10.10.10.61         2345:6789:AAAA:AAAA::61
      10.10.10.41         2345:6789::41

---   10.10.10.61         2345:6789:AAAA:AAAA::61
      ---                ---

NAT-RT#
NAT-RT#sh ipv6 nat statistics
Total active translations: 3 (2 static, 1 dynamic; 0 extended)
NAT-PT interfaces:
  FastEthernet0/0, FastEthernet1/0
Hits: 0 Misses: 0
Expired translations: 0
NAT-RT#

```

- Another alternative configuration for NAT-RT is perform NAT-PT static mapping between (10.10.10.41 – 20.20.20.61) and (2345:6789::41 – 2345:6789:AAAA:AAAA::61). The **ip address 10.10.10.61 255.255.255.0 secondary** interface subcommand on NAT-RT can be removed, and an IPv4 default route must be configured on V4-PC1.

```

!
ipv6 unicast-routing
!
interface FastEthernet0/0
 ip address 10.10.10.1 255.255.255.0
 ipv6 nat
!
interface FastEthernet1/0
 no ip address
 ipv6 address 2345:6789:AAAA:AAAA::1/64
 ipv6 nat
!
ipv6 nat v4v6 source 10.10.10.41 2345:6789::41
ipv6 nat v6v4 source 2345:6789:AAAA:AAAA::61 20.20.20.61
ipv6 nat prefix 2345:6789::/96
!

```

- Below shows the output of the **debug ipv6 nat** privileged command when V4-PC1 issued a ping to 20.20.20.61, the IPv4 address associated to 2345:6789:AAAA:AAAA::61:

```

NAT-RT#debug ipv6 nat
IPv6 NAT-PT debugging is on
NAT-RT#
00:02:50: IPv6 NAT: icmp src (10.10.10.41) -> (2345:6789::41), dst
(20.20.20.61) -> (2345:6789:AAAA:AAAA::61)
00:02:50: IPv6 NAT: icmp src (2345:6789:AAAA:AAAA::61) -> (20.20.20.61), dst
(2345:6789::41) -> (10.10.10.41)
00:02:50: IPv6 NAT: icmp src (10.10.10.41) -> (2345:6789::41), dst
(20.20.20.61) -> (2345:6789:AAAA:AAAA::61)
00:02:50: IPv6 NAT: icmp src (2345:6789:AAAA:AAAA::61) -> (20.20.20.61), dst
(2345:6789::41) -> (10.10.10.41)
00:02:50: IPv6 NAT: icmp src (10.10.10.41) -> (2345:6789::41), dst
(20.20.20.61) -> (2345:6789:AAAA:AAAA::61)
00:02:50: IPv6 NAT: icmp src (2345:6789:AAAA:AAAA::61) -> (20.20.20.61), dst
(2345:6789::41) -> (10.10.10.41)
00:02:50: IPv6 NAT: icmp src (10.10.10.41) -> (2345:6789::41), dst
(20.20.20.61) -> (2345:6789:AAAA:AAAA::61)
00:02:50: IPv6 NAT: icmp src (2345:6789:AAAA:AAAA::61) -> (20.20.20.61), dst
(2345:6789::41) -> (10.10.10.41)
NAT-RT#
NAT-RT#sh ipv6 nat translations
Prot  IPv4 source          IPv6 source
     IPv4 destination      IPv6 destination
---  ---                  ---
     10.10.10.41          2345:6789::41

---  20.20.20.61          2345:6789:AAAA:AAAA::61
     10.10.10.41          2345:6789::41

---  20.20.20.61          2345:6789:AAAA:AAAA::61
     ---                  ---

NAT-RT#

```

- Below shows the output of **debug ip icmp** on V4-PC1 when issue a ping to 20.20.20.61, the IPv4 address associated to 2345:6789:AAAA:AAAA::61:

```
V4-PC1#debug ip icmp
ICMP packet debugging is on
V4-PC1#
V4-PC1#ping 20.20.20.61

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 20.20.20.61, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 56/99/160 ms
V4-PC1#
00:02:53: ICMP: echo reply rcvd, src 20.20.20.61, dst 10.10.10.41
00:02:53: ICMP: echo reply rcvd, src 20.20.20.61, dst 10.10.10.41
00:02:53: ICMP: echo reply rcvd, src 20.20.20.61, dst 10.10.10.41
00:02:54: ICMP: echo reply rcvd, src 20.20.20.61, dst 10.10.10.41
V4-PC1#
```

- Below shows the output of **debug ip icmp** on V4-PC1 when issue a ping to 20.20.20.61, the IPv4 address associated to 2345:6789:AAAA:AAAA::61:

```
V6-PC1#debug ipv6 icmp
ICMP packet debugging is on
V6-PC1#
00:02:54: ICMPv6: Received ICMPv6 packet from FE80::204:4EFF:FE11:1111, type
135
00:02:54: ICMPv6: Received ICMPv6 packet from 2345:6789::41, type 128
00:02:54: ICMPv6: Received echo request from 2345:6789::41
00:02:54: ICMPv6: Sending echo reply to 2345:6789::41
00:02:54: ICMPv6: Received ICMPv6 packet from 2345:6789:AAAA:AAAA::1, type
136
00:02:54: ICMPv6: Received ICMPv6 packet from 2345:6789::41, type 128
00:02:54: ICMPv6: Received echo request from 2345:6789::41
00:02:54: ICMPv6: Sending echo reply to 2345:6789::41
00:02:54: ICMPv6: Received ICMPv6 packet from 2345:6789::41, type 128
00:02:54: ICMPv6: Received echo request from 2345:6789::41
00:02:54: ICMPv6: Sending echo reply to 2345:6789::41
00:02:54: ICMPv6: Received ICMPv6 packet from 2345:6789::41, type 128
00:02:54: ICMPv6: Received echo request from 2345:6789::41
00:02:54: ICMPv6: Sending echo reply to 2345:6789::41
V6-PC1#
```

## IPv4-Mapped NAT-PT Translation

- RFC 4038 – Application Aspects of IPv6 Transition and RFC 4291 – IP Version 6 Addressing Architecture define a standard called the **IPv4-Mapped IPv6 address** to address IPv4 packets that originated from an IPv6 network. An IPv6 host would use the IPv6 destination address `::FFFF:A.B.C.D` when it would like to send a packet to an IPv4 host whose address is A.B.C.D.
- The sample configuration below configures an IPv6-IPv4 address pool (20.20.20.61 – 20.20.20.69) that represents IPv6 hosts to the IPv4 network. Note that an IPv4 is able to initiate connection to an IPv6 host only after the NAT translation entry is created after an IPv6 host has accessed an IPv4 host. Static mapping should be implemented instead in order to provide 2-way connectivity – an IPv6 host is able to initiate connection to an IPv4 host and vice versa. However, if all connections will be originated from the IPv6 network, Port Address Translation (PAT) can be implemented to map all internal IPv6 addresses into a single external IPv4 address.
- IPv4-Mapped NAT-PT Configuration on NAT-RT:

```
!  
ipv6 unicast-routing  
!  
interface FastEthernet0/0  
 ip address 10.10.10.1 255.255.255.0  
 ipv6 nat  
!  
interface FastEthernet1/0  
 ipv6 address 2345:6789:AAAA:AAAA::1/64  
 ipv6 nat  
!  
ipv6 nat v6v4 source list NAT_TRAFFIC pool IPv6-IPv4  
ipv6 nat v6v4 pool IPv6-IPv4 20.20.20.61 20.20.20.69 prefix-length 24  
ipv6 nat prefix ::FFFF:0.0.0.0/96 v4-mapped NAT_TRAFFIC  
!  
!  
ipv6 access-list NAT_TRAFFIC  
 permit ipv6 any ::FFFF:0.0.0.0/96  
!
```

- Below shows that initially V4-PC1 was unable to initiate connections to V6-PC1 due to the NAT entry for 20.20.20.61 was not created yet until V6-PC1 initiated connections to the IPv4 network.

```
V4-PC1#debug ip icmp  
ICMP packet debugging is on  
V4-PC1#  
V4-PC1#ping 20.20.20.61  
  
Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echos to 20.20.20.61, timeout is 2 seconds:  
.U  
00:02:08: ICMP: dst (10.10.10.41) host unreachable rcv from 10.10.10.1.U  
00:02:10: ICMP: dst (10.10.10.41) host unreachable rcv from 10.10.10.1.  
Success rate is 0 percent (0/5)  
V4-PC1#
```

- Below shows the output of the **debug ipv6 icmp** and **debug ip icmp** on V6-PC1 and V4-PC1 when V6-PC1 initiated an ICMP ping to V4-PC1.

```
V6-PC1#debug ipv6 icmp
ICMP packet debugging is on
V6-PC1#
V6-PC1#ping ::FFFF:A0A:A29 | ping ::FFFF:10.10.10.41

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to ::FFFF:10.10.10.41, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 24/56/116 ms
V6-PC1#
00:03:29: ICMPv6: Sending echo request to ::FFFF:10.10.10.41
00:03:29: ICMPv6: Received ICMPv6 packet from ::FFFF:10.10.10.41, type 129
00:03:29: ICMPv6: Received echo reply from ::FFFF:10.10.10.41
00:03:29: ICMPv6: Sending echo request to ::FFFF:10.10.10.41
00:03:29: ICMPv6: Received ICMPv6 packet from ::FFFF:10.10.10.41, type 129
00:03:29: ICMPv6: Received echo reply from ::FFFF:10.10.10.41
00:03:29: ICMPv6: Sending echo request to ::FFFF:10.10.10.41
00:03:29: ICMPv6: Received ICMPv6 packet from ::FFFF:10.10.10.41, type 129
00:03:29: ICMPv6: Received echo reply from ::FFFF:10.10.10.41
00:03:30: ICMPv6: Sending echo request to ::FFFF:10.10.10.41
00:03:30: ICMPv6: Received ICMPv6 packet from ::FFFF:10.10.10.41, type 129
00:03:30: ICMPv6: Received echo reply from ::FFFF:10.10.10.41
00:03:30: ICMPv6: Sending echo request to ::FFFF:10.10.10.41
00:03:30: ICMPv6: Received ICMPv6 packet from ::FFFF:10.10.10.41, type 129
00:03:30: ICMPv6: Received echo reply from ::FFFF:10.10.10.41
V6-PC1#
=====
V4-PC1#
00:03:29: ICMP: echo reply sent, src 10.10.10.41, dst 20.20.20.61
00:03:29: ICMP: echo reply sent, src 10.10.10.41, dst 20.20.20.61
00:03:29: ICMP: echo reply sent, src 10.10.10.41, dst 20.20.20.61
00:03:29: ICMP: echo reply sent, src 10.10.10.41, dst 20.20.20.61
00:03:29: ICMP: echo reply sent, src 10.10.10.41, dst 20.20.20.61
V4-PC1#
```

- Below shows the NAT-PT processing on NAT-RT when V6-PC1 initiated an ICMP ping to V4-PC1.

```

NAT-RT#debug ipv6 nat
IPv6 NAT-PT debugging is on
NAT-RT#
00:03:28: IPv6 NAT: icmp src (2345:6789:AAAA:AAAA::61) -> (20.20.20.61), dst
 (::FFFF:10.10.10.41) -> (10.10.10.41)
00:03:28: IPv6 NAT: icmp src (10.10.10.41) -> (::FFFF:10.10.10.41), dst
 (20.20.20.61) -> (2345:6789:AAAA:AAAA::61)
00:03:28: IPv6 NAT: icmp src (2345:6789:AAAA:AAAA::61) -> (20.20.20.61), dst
 (::FFFF:10.10.10.41) -> (10.10.10.41)
00:03:28: IPv6 NAT: icmp src (10.10.10.41) -> (::FFFF:10.10.10.41), dst
 (20.20.20.61) -> (2345:6789:AAAA:AAAA::61)
00:03:28: IPv6 NAT: icmp src (2345:6789:AAAA:AAAA::61) -> (20.20.20.61), dst
 (::FFFF:10.10.10.41) -> (10.10.10.41)
00:03:28: IPv6 NAT: icmp src (10.10.10.41) -> (::FFFF:10.10.10.41), dst
 (20.20.20.61) -> (2345:6789:AAAA:AAAA::61)
00:03:28: IPv6 NAT: icmp src (2345:6789:AAAA:AAAA::61) -> (20.20.20.61), dst
 (::FFFF:10.10.10.41) -> (10.10.10.41)
00:03:28: IPv6 NAT: icmp src (10.10.10.41) -> (::FFFF:10.10.10.41), dst
 (20.20.20.61) -> (2345:6789:AAAA:AAAA::61)
00:03:28: IPv6 NAT: icmp src (2345:6789:AAAA:AAAA::61) -> (20.20.20.61), dst
 (::FFFF:10.10.10.41) -> (10.10.10.41)
00:03:28: IPv6 NAT: icmp src (10.10.10.41) -> (::FFFF:10.10.10.41), dst
 (20.20.20.61) -> (2345:6789:AAAA:AAAA::61)
00:03:28: IPv6 NAT: icmp src (2345:6789:AAAA:AAAA::61) -> (20.20.20.61), dst
 (::FFFF:10.10.10.41) -> (10.10.10.41)
00:03:28: IPv6 NAT: icmp src (10.10.10.41) -> (::FFFF:10.10.10.41), dst
 (20.20.20.61) -> (2345:6789:AAAA:AAAA::61)
NAT-RT#
NAT-RT#sh ipv6 nat translations
Prot  IPv4 source          IPv6 source
     IPv4 destination    IPv6 destination
---  ---
     10.10.10.41          2345:6789::A0A:A29

58    20.20.20.61,7155      2345:6789:AAAA:AAAA::61,7155
     10.10.10.41,7155    2345:6789::A0A:A29,7155

---   20.20.20.61          2345:6789:AAAA:AAAA::61
     10.10.10.41          2345:6789::A0A:A29

---   20.20.20.61          2345:6789:AAAA:AAAA::61
     ---                  ---

NAT-RT#
NAT-RT#debug ipv6 nat detailed
IPv6 NAT-PT detailed debugging is on
NAT-RT#
00:04:28: IPv6 NAT: deleted a NAT entry after timeout
NAT-RT#

```

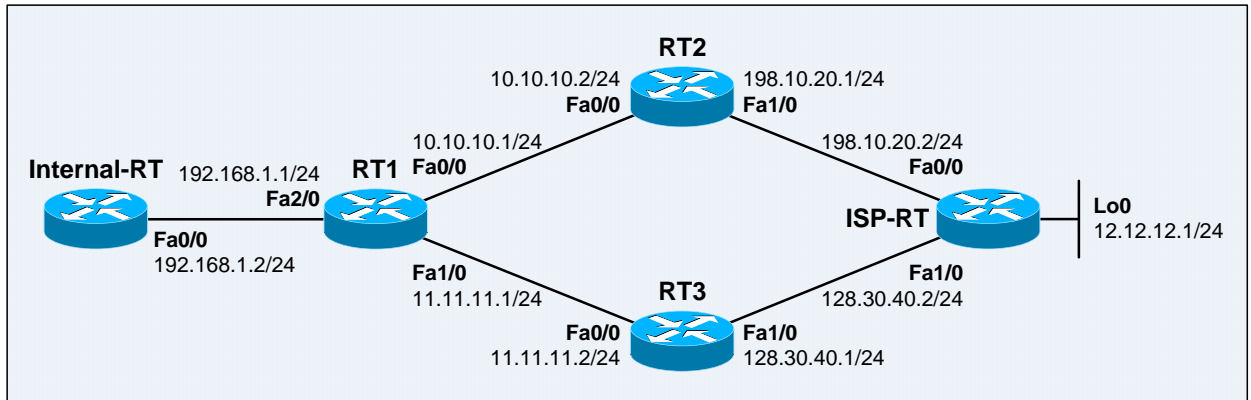


## IPv6 Ping Output Characters

- The following characters can be displayed as output for the IPv6 **ping** command:

Character	Description
!	Indicates a receipt of a reply.
.	Indicates the request has timed out while waiting for a reply.
?	Unknown error.
@	Unreachable for unknown reason.
A	Administratively unreachable. Usually means that an access control list (ACL) is blocking traffic.
B	Packet too big.
H	Host unreachable.
N	Network unreachable (beyond scope).
P	Port unreachable.
R	Parameter problem.
T	Time exceeded.
U	No route to host.

## The ip default-network Global Configuration Command



**Figure A6-3:** Network Setup for the **ip default-network** Global Configuration Command

- This configuration example shows that without any dynamic routing protocol running, a router can be configured to choose from a number of candidate default routes based on whether the routing table has routes to networks other than 0.0.0.0/0. The **ip default-network** command allows a router to choose a default route or a gateway of last resort by checking its routing table, rather than based on static routes configured to specific next-hops.
- The **ip default-network** A.B.C.D command sets network A.B.C.D as the default route. Packets destined to networks that are not in the routing table (unknown destination networks) will be directed or sent to this default network.

- Below shows the routing table of RT1 before any **ip default-network** command is configured:

```

RT1 (config) #ip route 198.10.20.0 255.255.255.0 10.10.10.2
RT1 (config) #ip route 128.30.40.0 255.255.255.0 11.11.11.2
RT1 (config) #^Z
RT1#
RT1#sh ip route

Gateway of last resort is not set

    128.30.0.0/24 is subnetted, 1 subnets
S       128.30.40.0 [1/0] via 11.11.11.2
    10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0
    11.0.0.0/24 is subnetted, 1 subnets
C       11.11.11.0 is directly connected, FastEthernet1/0
C       192.168.1.0/24 is directly connected, FastEthernet2/0
S       198.10.20.0/24 [1/0] via 10.10.10.2
RT1#

```

- If a router has a route to the network specified by the **ip default-network** command, the router will consider the route to that network as the gateway of last resort and flag the route as a **candidate default route**. Below shows the routing table of RT1 when the **ip default-network 198.10.20.0** command is configured on it:

```

RT1 (config) #ip default-network 198.10.20.0
RT1 (config) #^Z
RT1#
RT1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default
       o - ODR, P - periodic downloaded static route

Gateway of last resort is 10.10.10.2 to network 198.10.20.0

    128.30.0.0/24 is subnetted, 1 subnets
S       128.30.40.0 [1/0] via 11.11.11.2
    10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0
    11.0.0.0/24 is subnetted, 1 subnets
C       11.11.11.0 is directly connected, FastEthernet1/0
C       192.168.1.0/24 is directly connected, FastEthernet2/0
S*     198.10.20.0/24 [1/0] via 10.10.10.2
RT1#
RT1#sh ip protocols

RT1#
RT1#ping 12.12.12.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.12.12.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 68/92/124 ms
RT1#

```

- 10.10.10.2 is now the gateway of last resort of RT1; and this result is not dependent on any routing protocol.
- Below shows the configuration steps for adding another candidate default route with another **ip default-network** command:

```

RT1(config)#do sh run | in ip default-network|ip route
ip default-network 198.10.20.0
ip route 128.30.40.0 255.255.255.0 11.11.11.2
ip route 198.10.20.0 255.255.255.0 10.10.10.2
RT1(config)#
RT1(config)#ip default-network 128.30.40.0
RT1(config)#do sh run | in ip default-network|ip route
ip default-network 198.10.20.0
ip route 128.30.0.0 255.255.0.0 128.30.40.0
ip route 128.30.40.0 255.255.255.0 11.11.11.2
ip route 198.10.20.0 255.255.255.0 10.10.10.2
RT1(config)#
RT1(config)#no ip route 128.30.0.0 255.255.0.0 128.30.40.0
%No matching route to delete
RT1(config)#
RT1(config)#do sh ip route

Gateway of last resort is 10.10.10.2 to network 198.10.20.0

    128.30.0.0/16 is variably subnetted, 2 subnets, 2 masks
S    128.30.0.0/16 [1/0] via 128.30.40.0
S     128.30.40.0/24 [1/0] via 11.11.11.2
    10.0.0.0/24 is subnetted, 1 subnets
C     10.10.10.0 is directly connected, FastEthernet0/0
    11.0.0.0/24 is subnetted, 1 subnets
C     11.11.11.0 is directly connected, FastEthernet1/0
S*   198.10.20.0/24 [1/0] via 10.10.10.2

```

**Note:** The network specified by the new **ip default-network** command is not flagged as a candidate default route; additionally, the command is not shown in the running configuration. The **ip default-network** command is classful. When the network specified with this command is a subnet of a major network (eg: 10.10.0.0, 172.16.1.0), the router will configure a static route to the major network number (**ip route 128.30.0.0 255.255.0.0**) for the **ip default-network** command. The static route is configured without the notice of the user, as no message will be displayed. Deleting this route requires the **no ip default-network** command instead of the **no ip route** command!

- The **ip default-network** command must be issued again using the major network number in order to flag the network as a candidate default route. **Note:** The **ip default-network 128.30.40.0** command must be issued in order to obtain the output below.

```

RT1(config)#ip default-network 128.30.0.0
RT1(config)#do sh run | in ip default-network
ip default-network 198.10.20.0
ip default-network 128.30.0.0
RT1(config)#
RT1(config)#do sh ip route

Gateway of last resort is 128.30.40.0 to network 128.30.0.0

* 128.30.0.0/16 is variably subnetted, 2 subnets, 2 masks
S* 128.30.0.0/16 [1/0] via 128.30.40.0
S 128.30.40.0/24 [1/0] via 11.11.11.2
10.0.0.0/24 is subnetted, 1 subnets
C 10.10.10.0 is directly connected, FastEthernet0/0
11.0.0.0/24 is subnetted, 1 subnets
C 11.11.11.0 is directly connected, FastEthernet1/0
C 192.168.1.0/24 is directly connected, FastEthernet2/0
S* 198.10.20.0/24 [1/0] via 10.10.10.2
RT1#
RT1#debug ip packet
IP packet debugging is on
RT1#
RT1#ping 12.12.12.1 rep 1

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 12.12.12.1, timeout is 2 seconds:
!
Success rate is 100 percent (1/1), round-trip min/avg/max = 268/268/268 ms
RT1#
00:11:28: IP: tableid=0, s=11.11.11.1 (local), d=12.12.12.1
(FastEthernet1/0), routed via FIB
00:11:28: IP: s=11.11.11.1 (local), d=12.12.12.1 (FastEthernet1/0), len 100,
sending
00:11:28: IP: tableid=0, s=12.12.12.1 (FastEthernet1/0), d=11.11.11.1
(FastEthernet1/0), routed via RIB
00:11:28: IP: s=12.12.12.1 (FastEthernet1/0), d=11.11.11.1
(FastEthernet1/0), len 100, rcvd 3
RT1#

```

The output of the **debug ip packet** privileged command shows that the new default route to 128.30.40.0 via Fa1/0 to 11.11.11.2 is in effect.

- **Note:** If the original static route is destined to the major network, configuring the default network twice would not be necessary. To be specific, if **ip route 128.30.0.0 255.255.0.0 11.11.11.2** is configured instead of **ip route 128.30.40.0 255.255.255.0 11.11.11.2**, only **ip default-network 128.30.0.0** is required instead of both **ip default-network 128.30.40.0** and **ip default-network 128.30.0.0**.

```

RT1#sh ip route

Gateway of last resort is 11.11.11.2 to network 128.30.0.0

S*   128.30.0.0/16 [1/0] via 11.11.11.2
     10.0.0.0/24 is subnetted, 1 subnets
C     10.10.10.0 is directly connected, FastEthernet0/0
     11.0.0.0/24 is subnetted, 1 subnets
C     11.11.11.0 is directly connected, FastEthernet1/0
C     192.168.1.0/24 is directly connected, FastEthernet2/0
S*   198.10.20.0/24 [1/0] via 10.10.10.2
RT1#
RT1#sh run | in ip default-network|ip route
ip default-network 198.10.20.0
ip default-network 128.30.0.0
ip route 128.30.0.0 255.255.0.0 11.11.11.2
ip route 198.10.20.0 255.255.255.0 10.10.10.2
RT1#

```

- The major difference between configuring a static default route with the **ip route 0.0.0.0 0.0.0.0** command and using the **ip default-network** command is that a static default route only defines a default route for the router which the command is configured on; while the default route selected by the **ip default-network** command can be propagated via a routing protocol. Below shows that RT1 advertises the default route as 0.0.0.0 to Internal-RT via RIPv1. The route is designated as a candidate default route on Internal-RT.

```

RT1:
!
router rip
 network 192.168.1.0
 default-information originate
!
=====
Internal-RT:
!
router rip
 network 192.168.1.0
!
Internal-RT#sh ip route

Gateway of last resort is 192.168.1.1 to network 0.0.0.0

C     192.168.1.0/24 is directly connected, FastEthernet0/0
R*   0.0.0.0/0 [120/1] via 192.168.1.1, 00:00:07, FastEthernet0/0
Internal-RT#
Internal-RT#ping 12.12.12.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.12.12.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 64/101/132 ms
Internal-RT#

```

- **Note:** The **default-information originate** RIP router subcommand is required on RT1. Some routing protocols (eg: EIGRP) do not require this command to redistribute the default route.
- Below shows that when RT1 loses the route to the current default network, it will select other candidate default routes as the new default network:

```

RT1#debug ip routing
IP routing debugging is on
RT1#
RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#no ip route 128.30.40.0 255.255.255.0 11.11.11.2
RT1(config)#^Z
RT1#
00:03:09: RT: del 128.30.40.0/24 via 11.11.11.2, static metric [1/0]
00:03:09: RT: delete subnet route to 128.30.40.0/24
00:03:09: RT: NET-RED 128.30.40.0/24
RT1#
RT1#sh ip route

Gateway of last resort is 128.30.40.0 to network 128.30.0.0

S* 128.30.0.0/16 [1/0] via 128.30.40.0
  10.0.0.0/24 is subnetted, 1 subnets
C    10.10.10.0 is directly connected, FastEthernet0/0
  11.0.0.0/24 is subnetted, 1 subnets
C    11.11.11.0 is directly connected, FastEthernet1/0
C    192.168.1.0/24 is directly connected, FastEthernet2/0
S* 198.10.20.0/24 [1/0] via 10.10.10.2
RT1#
00:04:02: RT: recursion error routing 128.30.40.0 - probable routing loop
00:04:02: RT: del 128.30.0.0 via 128.30.40.0, static metric [1/0]
00:04:02: RT: delete network route to 128.30.0.0
00:04:02: RT: NET-RED 128.30.0.0/16
00:04:02: RT: NET-RED 128.30.0.0/16
00:04:02: RT: default path is now 198.10.20.0 via 10.10.10.2
00:04:02: RT: new default network 198.10.20.0
00:04:02: RT: NET-RED 198.10.20.0/24
00:04:02: RT: NET-RED 198.10.20.0/24
RT1#
RT1#sh ip route

Gateway of last resort is 10.10.10.2 to network 198.10.20.0

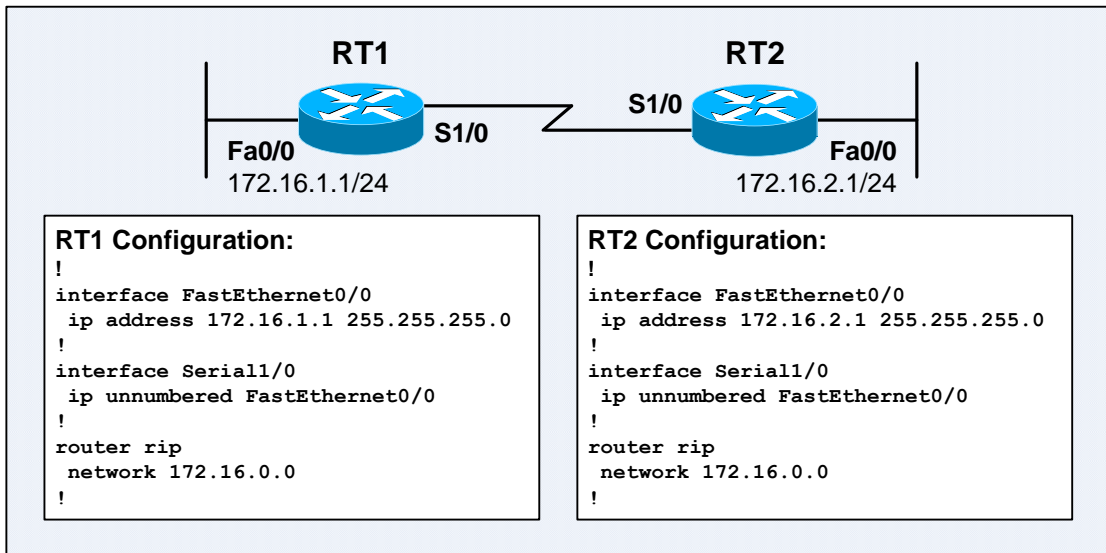
  10.0.0.0/24 is subnetted, 1 subnets
C    10.10.10.0 is directly connected, FastEthernet0/0
  11.0.0.0/24 is subnetted, 1 subnets
C    11.11.11.0 is directly connected, FastEthernet1/0
C    192.168.1.0/24 is directly connected, FastEthernet2/0
S* 198.10.20.0/24 [1/0] via 10.10.10.2
RT1#
RT1#sh run | in ip default-network|ip route
ip default-network 198.10.20.0
ip default-network 128.30.0.0
ip route 128.30.0.0 255.255.0.0 128.30.40.0
ip route 198.10.20.0 255.255.255.0 10.10.10.2
RT1#

```

## IP Unnumbered Serial Interfaces

- The **ip unnumbered** *{intf-type intf-num}* interface subcommand enables IP processing on an interface **without assigning an IP address** to it hence can conserve network address space. The *intf-type intf-num* parameters specify another active interface (line protocol is up) that has an assigned IP address. It **cannot** specify another unnumbered interface. It is recommended to point an unnumbered interface to a loopback interface since loopback interfaces never fail.
- An unnumbered interface will use (or borrow) the address of the specified interface as the source address whenever it generates a packet. The address of the specified interface is also used to determine the routing processes that are sending routing updates over the unnumbered interface.  
Ex: An RIP router that is configured when the **network 10.0.0.0** router subcommand indicates that all physical interfaces with an address in network 10.0.0.0 and all unnumbered interfaces that specify an interface that has an address in network 10.0.0.0 will participate in the RIP.
- Below list the **restrictions** of unnumbered interfaces:
  - i) Serial interfaces with HDLC, PPP, LAPB, Frame Relay, SLIP, and tunnel interfaces **ca** be unnumbered. X.25 and SMDS interfaces **cannot** be unnumbered.
  - ii) The **ping** command cannot be used to determine whether an unnumbered interface is up, as the interface has no address. SNMP can be used to monitor the status of the interface.
  - iii) IP security features are not supported on unnumbered interfaces.
  - iv) IP unnumbered is supported on point-to-point interfaces only. The "Point-to-point (non-multi-access) interfaces only" error message will be received when configuring this feature on a multi-access (eg: Ethernet) or loopback interface.
- **Note:** The following information is inaccurate and has been proven using real network setups! Routers normally insert the source address of a routing update as the next hop in the routing table, and the next hop is often an address on a directly connected network. However, this is no longer the case when the IP unnumbered interfaces of a serial link are associated with interfaces reside in different subnets or major networks. Routes learned through an IP unnumbered interface have the interface as the next hop instead of the source address of the routing update. The purpose is to avoid the invalid next hop address problem in which a routing update sourced from a next hop of a non-directly connected network is received.  
Ex: RT1 and RT2 are connected with an unnumbered serial link. RT1 has all its interfaces and unnumbered interface in network 172.16.0.0, while RT2 has all its interfaces and unnumbered interface in network 172.17.0.0. If OSPF is configured to run on the unnumbered serial link, it must be configured to summarize the subnet information as OSPF won't send it across the link.
- Below shows the steps for configuring an unnumbered serial interface:

```
Router(config)#int lo0
Router(config-if)#ip add 10.10.10.1 255.255.255.0
Router(config-if)#int s1/0
Router(config-if)#ip unnumbered lo0
Router(config-if)#^Z
Router#sh ip int brief
Interface          IP-Address      OK? Method Status      Protocol
Loopback0          10.10.10.1     YES manual  up          up
Serial1/0          10.10.10.1     YES NVRAM   up          up
Router#
```



**Figure A6-4: Same Major Network, Different Subnets IP Unnumbered Network**

- Below shows the routing tables of RT1 and RT2 for the configuration above:

```

RT1#sh ip route

    172.16.0.0/24 is subnetted, 3 subnets
R       172.16.0.0 [120/1] via 172.16.2.1, 00:00:10, Serial1/0
C       172.16.1.0 is directly connected, FastEthernet0/0
R       172.16.2.0 [120/1] via 172.16.2.1, 00:00:10, Serial1/0
RT1#
RT1#ping 172.16.2.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.2.1, timeout is 2 seconds:
!!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/47/92 ms
RT1#ping 172.16.2.0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.2.0, timeout is 2 seconds:
!!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/36/64 ms
-----
RT2#sh ip route

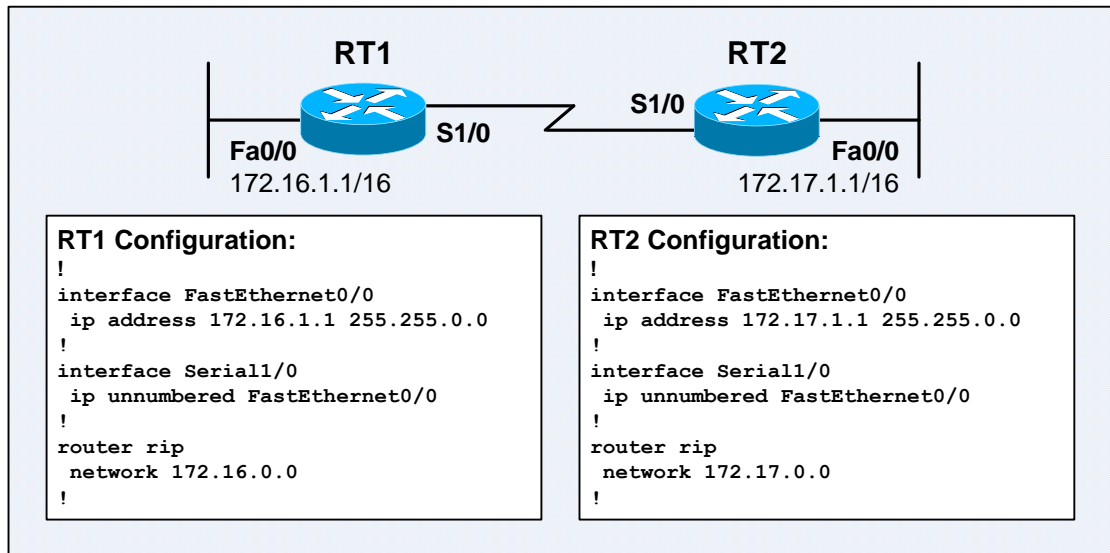
    172.16.0.0/24 is subnetted, 3 subnets
R       172.16.0.0 [120/1] via 172.16.1.1, 00:00:15, Serial1/0
R       172.16.1.0 [120/1] via 172.16.1.1, 00:00:15, Serial1/0
C       172.16.2.0 is directly connected, FastEthernet0/0
RT2#
RT2#ping 172.16.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
!!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/33/88 ms
RT2#ping 172.16.1.0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.0, timeout is 2 seconds:
!!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/26/44 ms

```





**Figure A6-5: Different Major Networks, No Subnet IP Unnumbered Network**

- Below shows the routing tables of RT1 and RT2 for the configuration above:

```

RT1#sh ip route

R    172.17.0.0/16 [120/1] via 172.17.1.1, 00:00:11, Serial1/0
C    172.16.0.0/16 is directly connected, FastEthernet0/0
RT1#
RT1#ping 172.17.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.17.1.1, timeout is 2 seconds:
!!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/28/48 ms
RT1#ping 172.17.1.0

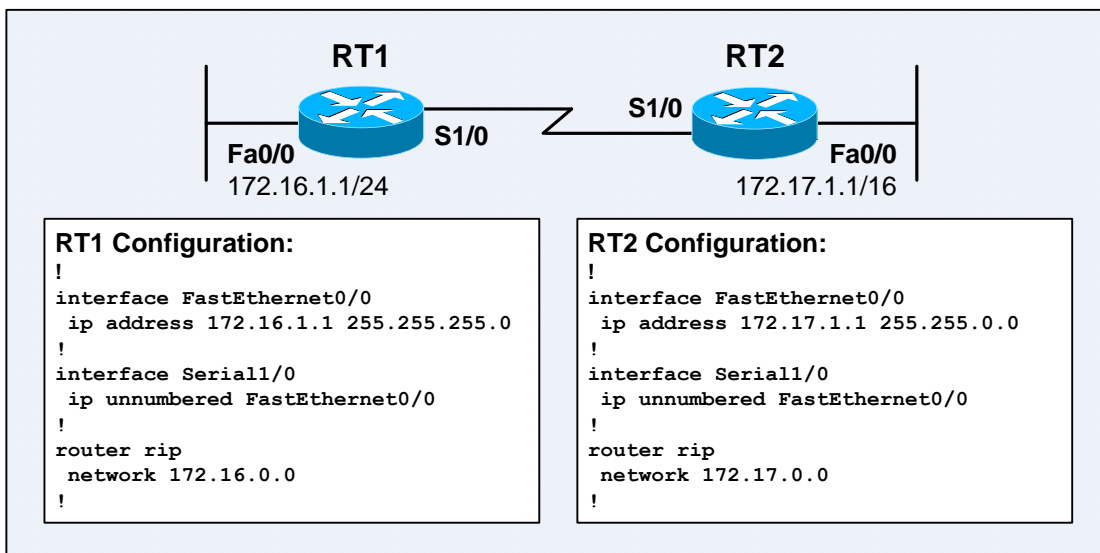
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.17.1.0, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
RT1#
-----
RT2#sh ip route

C    172.17.0.0/16 is directly connected, FastEthernet0/0
R    172.16.0.0/16 [120/1] via 172.16.1.1, 00:00:11, Serial1/0
RT2#
RT2#ping 172.16.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
!!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/41/88 ms
RT2#ping 172.16.1.0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.0, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
RT2#

```



**Figure A6-6:** Different Major Networks, With and Without Subnets IP Unnumbered Network

- Below shows the routing tables of RT1 and RT2 for the configuration above:

```

RT1#sh ip route

R    172.17.0.0/16 [120/1] via 172.17.1.1, 00:00:01, Serial1/0
     172.16.0.0/24 is subnetted, 1 subnets
C     172.16.1.0 is directly connected, FastEthernet0/0
RT1#
RT1#ping 172.17.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.17.1.1, timeout is 2 seconds:
!!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/41/68 ms
RT1#ping 172.17.1.0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.17.1.0, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
-----

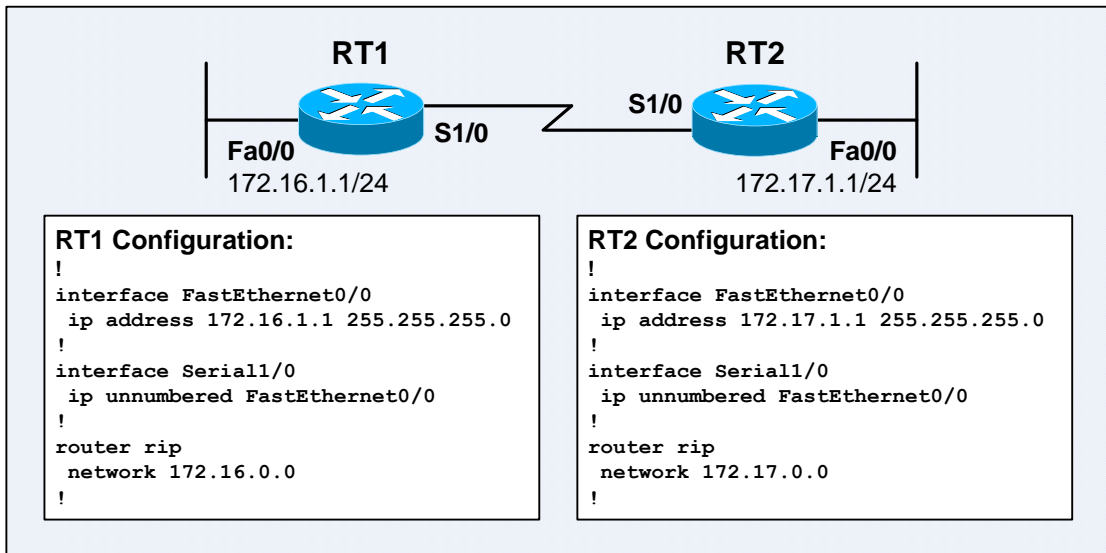
RT2#sh ip route

C    172.17.0.0/16 is directly connected, FastEthernet0/0
     172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
R    172.16.0.0/16 [120/1] via 172.16.1.1, 00:00:17, Serial1/0
R    172.16.1.0/32 [120/1] via 172.16.1.1, 00:00:17, Serial1/0
RT2#
RT2#ping 172.16.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
!!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/41/64 ms
RT2#ping 172.16.1.0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.0, timeout is 2 seconds:
!!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/30/36 ms

```



**Figure A6-7: Different Major Networks, Different Subnets IP Unnumbered Network**

- Below shows the routing tables of RT1 and RT2 for the configuration above:

RT1#**sh ip route**

```

172.17.0.0/16 is variably subnetted, 2 subnets, 2 masks
R    172.17.1.0/32 [120/1] via 172.17.1.1, 00:00:06, Serial1/0
R    172.17.0.0/16 [120/1] via 172.17.1.1, 00:00:06, Serial1/0
172.16.0.0/24 is subnetted, 1 subnets
C    172.16.1.0 is directly connected, FastEthernet0/0

```

RT1#**ping 172.17.1.1**

```

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.17.1.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/45/104 ms

```

RT1#**ping 172.17.1.0**

```

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.17.1.0, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/36/60 ms
-----

```

RT2#**sh ip route**

```

172.17.0.0/24 is subnetted, 1 subnets
C    172.17.1.0 is directly connected, FastEthernet0/0
172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
R    172.16.0.0/16 [120/1] via 172.16.1.1, 00:00:03, Serial1/0
R    172.16.1.0/32 [120/1] via 172.16.1.1, 00:00:03, Serial1/0

```

RT2#**ping 172.16.1.1**

```

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/35/48 ms

```

RT2#**ping 172.16.1.0**

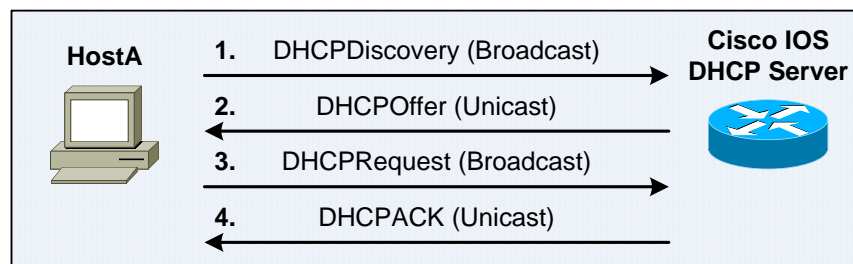
```

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.0, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/42/64 ms

```

## DHCP – Dynamic Host Configuration Protocol

- DHCP is often used to automatically assign IP configurations to desktop clients in an enterprise.
- DHCP clients lease the IP configuration information from the DHCP server for an administratively defined period. When the lease duration is expired, the client must request for another address, and it is normally reassigned with the same IP address.
- DHCP servers are normally setup to assign IP addresses from predefined pool. DHCP servers can also provide other information, eg: default gateway address, DNS server addresses, WINS (NetBIOS) server addresses, and domain names.  
**Note:** WINS is typically used to resolve a NetBIOS name to an IP address.
- DHCP is a superset of BOOTP; it uses the same protocol structure as BOOTP with many enhancements. The 2 major enhancements are support for address pools and lease time.



**Figure A6-8:** The DHCP Operation

- Below shows the typical operation of DHCP:
  - 1) HostA (a DHCP client) requests an IP address upon system startup or a network cable is connected. It tries to locate a DHCP server by sending a **DHCPDiscovery** broadcast. DHCPDiscovery messages are sent as FFFF.FFFF.FFFF L2 broadcast and 255.255.255.255 L3 broadcast.
  - 2) Upon received the DHCPDiscovery broadcast, the DHCP server determines whether it can service the request with its own database. If yes, the DHCP server offers the client IP configuration in the form of a **DHCPOffer** unicast; if no, the server may relay and forward the request to other configured DHCP server(s). The DHCPOffer is a proposed configuration that may include IP address, DNS server addresses, and lease time.
  - 3) The DHCP client sends a **DHCPRequest** broadcast to request the IP parameters specified in the DHCPOffer. Broadcast instead of unicast is used, as the DHCPDiscovery broadcast might have reach multiple DHCP servers and multiple offers have been made. The DHCPRequest broadcast informs all DHCP servers which offer was accepted (usually the first offer received) and the client declining other offers.
  - 4) Upon received the DHCPRequest, the DHCP server makes the configuration official by sending a **DHCPACK** unicast. The DHCP client begins to use the assigned address upon it received the DHCPACK. It is possible but very unlikely that the DHCP server does not send the DHCPACK due to the information might have been leased to another client. The server can also respond with a **DHCPNACK** message, which informs the client that the offer is no longer valid and the client should request addressing information again.

**Note:** DHCPOffer and DHCPACK messages are **broadcasts** instead of unicasts when the Broadcast bit in the DHCPDiscovery message is set.

- It is unable to predict which DHCP server will respond to the DHCPDiscovery broadcast message of a DHCP client when multiple DHCP servers are active on the same network. A DHCP Offer for an IP address from the DHCP server is not guarantee that it will be allocated to the client; but the DHCP server usually reserves the IP address until the DHCP client has formally accept the offer and use the IP address.
- By default, a Cisco IOS DHCP server always perform **server-based IP conflict detection** by issuing 2 ICMP Echo Requests (ping) to verify that an address is not in use before it is being offered to a DHCP client. The purpose is to avoid potential IP address conflict due to statically configured address. The timeout of those pings is 500 milliseconds. If the ping times out, the address will be assigned; if an ICMP Echo Reply returns, the address should not be assigned! The **ip dhcp ping packets {0 – 10}** and the **ip dhcp ping timeout {timeout-value}** global configuration commands modify the default values. This conflict detection mechanism using ICMP Echo Request / Reply can be disabled by specifying 0 as the value for ping packets.
- **Gratuitous ARP** is an ARP request that is sent out from a host to resolve its own IP address. A host always sends out multiple Gratuitous ARP requests upon boot up or link up to detect duplicate address before using an IP address in order to avoid IP conflict. Normally there should be no Gratuitous ARP reply, as there should not be another host that is also configured with the same IP address as the Gratuitous ARP sender. Gratuitous ARP is normally being used in **client-based IP conflict detection**. A client will send DHCPDecline when it detects there will be an IP address conflict when using an offered IP address.
- A Gratuitous ARP Request (and Reply) comprises of source MAC address of the sender, destination MAC address FFFF.FFFF.FFFF, and source and destination IP addresses of the sender. **Note:** The destination IP address of a normal ARP request is the IP address to be resolved.
- An existing host in a network would detect an IP address conflict when it receives a Gratuitous ARP request that comprises of a source IP address that matches its IP address.
- Another usage of Gratuitous ARP is **ARP Table Update**. When a clustering or high-availability solution moves an IP address from an NIC to another NIC (resides on the same or different host), it will broadcast Gratuitous ARP replies destined to the network layer broadcast address (eg: 192.168.0.255) to inform other hosts to update their ARP tables with the new MAC address. **Note:** Gratuitous ARP reply is also being sent upon issuing the **clear arp** privileged command.
- Most DHCP deployments use the **dynamic** method to assign IP addresses to hosts for a limited period of time (lease period) or until the hosts voluntarily and explicitly release the IP addresses. This mechanism supports automatic address reuse when the host to which the IP address has been assigned no longer requires the IP address.  
DHCP supports 2 other address allocation mechanisms as below:
  - **Automatic** – DHCP chooses an IP address from an address pool, and the IP address assignment for a host is **permanent**. This method requires the setup of database agents.
  - **Manual** – The network admin assigns and maps an IP address to a specific MAC address. The primary responsibility of DHCP is simply delivering the appropriate IP addresses to the corresponding hosts. Manual bindings cannot be configured within the same pool that is configured for automatic bindings.
- An **address binding** is the mapping between the IP address and MAC address of a client. The IP address of the client can be configured by the network admin (manual binding) or automatic assigned via a DHCP pool. Cisco IOS devices store the address pools in NVRAM, and therefore retained upon router reboots. There is no limit on the number of address pools.

- **Automatic bindings** are IP addresses that have been automatically mapped to the MAC addresses of hosts that are found in the DHCP database. Automatic bindings are stored on a remote host called the **database agent**, which is any host that stores the DHCP binding database. The bindings are saved as static mapping text entries for easy maintenance. The typical database agents are TFTP server, FTP server, and RCP server. Multiple DHCP database agents and the interval between database updates and transfers for each agent can be configured.
  - Cisco strongly recommended using database agents; however, Chris Bryant (CCIE#12933) has encountered unpredictable results when setting up database agents in lab environments! The **ip dhcp database {url} [timeout seconds | write-delay seconds]** DHCP subcommand configures a Cisco IOS DHCP server to save automatic bindings on a database agent. The **write-delay** keyword specifies the period of time to wait before writing database changes.
  - When we choose not to use database agents, it is recommended to disable DHCP conflict logging which records the address conflicts on the DHCP server using the **no ip dhcp conflict logging** global configuration command. Because if conflict logging is enabled but no database agent is configured, automatic bindings are lost upon router reboots. Possible false conflicts can occur and cause the address to be removed from the address pool until the network admin intervenes.
  - **Manual bindings** which are simply address pools are also stored in NVRAM and retained upon router reboots; and there is no limit on the number of manual bindings.
- Sample Cisco IOS DHCP manual binding configuration:

```

Router (config) #ip dhcp pool manual-binding01
Router (dhcp-config) #host ?
    A.B.C.D  IP address in dotted-decimal notation

Router (dhcp-config) #host 192.168.1.101 ?
    /nn or A.B.C.D  Network mask or prefix length
    <cr>

Router (dhcp-config) #host 192.168.1.101
Router (dhcp-config) #hardware-address ?
    WORD  Dotted-hexadecimal string (aabb.ccdd.eeff ...)

Router (dhcp-config) #hardware-address 1122.3344.5566
Router (dhcp-config) #hardware-address 1122.3344.5566 ?
    <0-255>  ARP hardware type from "Assigned Numbers" RFC
    ethernet  10Mb Ethernet
    ieee802   IEEE 802 networks
    <cr>

Router (dhcp-config) #hardware-address 1122.3344.5566 ethernet
Router (dhcp-config) #client-name IT-Manager
Router (dhcp-config) #end
Router#
Router#sh run
--- output omitted ---
!
ip dhcp pool manual-binding01
    host 192.168.1.1 255.255.255.0
    hardware-address 1122.3344.5566
    client-name IT-Manager
!
--- output omitted ---

```

- **Attribute Inheritance.** The DHCP database is organized as a tree. The root of the tree is the network address pools, branches are subnetwork address pools, and leaves are manual bindings to clients. Subnetworks inherit network parameters and clients inherit subnetwork parameters. Therefore, common parameters, eg: domain name, should be configured at the highest level (network or subnetwork) of the tree. Inherited parameters can be overridden – if a parameter is defined in both the network and subnetwork, the definition of the subnetwork is used. Address leases are not inherited. If a lease is not specified for an IP address, the DHCP server assigns a one-day lease for the IP address by default.
- Configuration parameters and other control information are carried in tagged data items that are stored in the Options fields of the DHCP messages. Options provide a method for appending additional information and therefore are often being utilized by DHCP vendors to provide additional information that is not designed into the DHCP protocol to their DHCP clients.
- The Cisco IOS DHCP server accepts address assignment requests and renewals and assigns the IP addresses from predefined address groups contained within the DHCP address pools. These address pools can also be configured to provide additional info to the requesting clients, eg: the IP address of the default gateway router, the DNS server, and other useful parameters. The Cisco IOS DHCP server is able to accept broadcasts from locally-attached LAN segments and DHCP requests that have been forwarded by other DHCP relay agents on the network.
- The Cisco IOS DHCP server is able to allocate dynamic IP addresses based on the **DHCP Option 82 – DHCP Relay Agent Information Option** sent by the DHCP relay agents. Automatic DHCP address allocation is typically based on an IP address, whether it be the gateway address (in the giaddr field of the DHCP packet) or the incoming interface IP address. Sometimes it is necessary to use additional information to determine the IP address allocation. By using DHCP Option 82, the Cisco IOS relay agent is able to include additional information about itself when forwarding the client-originated DHCP packets to a DHCP server. The Cisco IOS DHCP server can then utilize the additional information in DHCP Option 82 to allocate IP addresses to DHCP clients appropriately.
- Sample Cisco IOS DHCP configuration:

```

Router (config) #ip dhcp pool GndFloor_DHCPPool
Router (dhcp-config) #network 172.16.1.0 255.255.255.0
Router (dhcp-config) #domain-name cisco.com
Router (dhcp-config) #default-router 172.16.1.1
Router (dhcp-config) #dns-server 172.16.1.11 172.16.1.12
Router (dhcp-config) #netbios-name-server 172.16.1.11 172.16.1.12
Router (dhcp-config) #lease 8
Router (dhcp-config) #exit
Router (config) #ip dhcp excluded-address 172.16.1.1 172.16.1.50
Router (config) #ip dhcp excluded-address 172.16.1.201 172.16.1.254
Router (config) #end
Router#

```

**Note:** The DHCP pool configuration mode is identified by the Router (dhcp-config) # prompt; the prompt is not Router (config-dhcp) # as we might expect.

- The **[no] service dhcp** global configuration command enables or disables the Cisco IOS DHCP server and relay agent processes respectively. Both services are enabled by default.

- The **ip dhcp pool** {*pool-name*} global configuration command defines a pool of addresses to be leased to hosts.
- The **network** {*network-number*} {*network-mask* | *prefix-length*} DHCP subcommand is used to define the range of addresses to be leased to hosts. The **ip dhcp excluded-address** {*start-ip-addr*} [*end-ip-addr*] global configuration command reserves IP addresses that are statically assigned to devices and hence exclude them from the DHCP pool to reduce conflicts.
- Cisco IOS DHCP server configuration supports up to **8** default gateways, **8** DNS servers, and **8** Microsoft NetBIOS WINS (**Windows Internet Name Service**) servers.
- The **lease** {*days* [*hours*] [*mins*] | **infinite**} DHCP subcommand defines the duration of the DHCP lease time. The default lease time is 1 day. The DHCP lease can also be made indefinite. The lease time specifies a time period that the client is allowed to use the IP address.
- The **show ip dhcp pool** [*pool-name*] EXEC command displays information of the DHCP address pools.
- The **show ip dhcp binding** [*ip-addr*] EXEC command displays address bindings (MAC to IP address), the lease expiration, and the lease type (how the IP address was assigned). Lease bindings are **automatically** created and released.
- The **show ip dhcp conflict** EXEC command displays address conflicts with the corresponding detection method and detection time.
- In the past, each Cisco IOS DHCP server must be configured with all the parameters and options. The Cisco IOS has been revised and enhanced to allow remote Cisco IOS DHCP servers to import option parameters from a centralized server. Network administrators can now configure one or more centralized DHCP servers to update specific DHCP options within the DHCP pools. The remote DHCP servers can then request or **import** these option parameters from the centralized DHCP servers.

## Easy IP

- Easy IP is a Cisco IOS feature that allows a router to request an IP address (as a DHCP client) via PPP, and performs NAT for LAN hosts with the dynamically assigned WAN IP address.
- Easy IP is commonly used in SOHO routers. It requires little or no administrative intervention.



## IP Helper Addresses

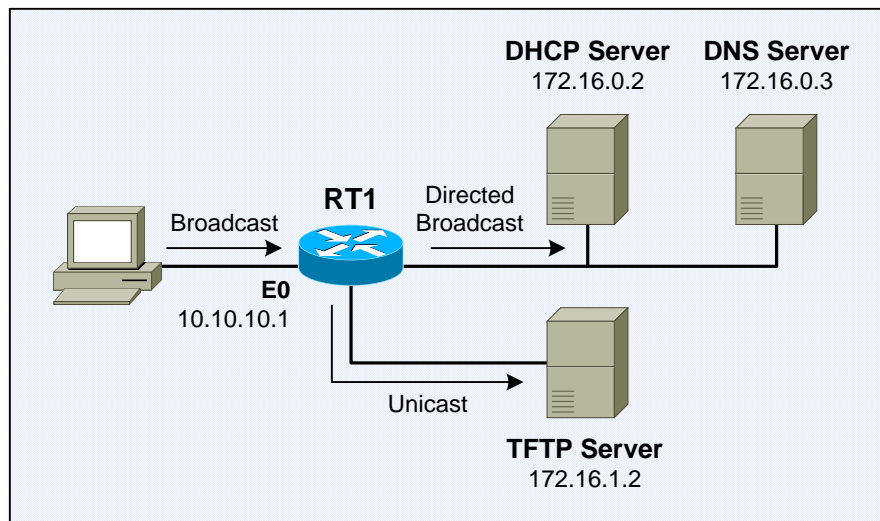
- DHCP and DNS rely mostly on broadcasts for their operations. Other usages of broadcasts are:
  - i) Routers may use broadcasts to locate TFTP servers.
  - ii) Clients may use broadcasts to locate a TACACS security server.
- Clients in a hierarchical network normally do not reside on the same network as the DHCP and DNS servers. They rely mostly on broadcasts to locate those servers. However, routers do not forward broadcasts beyond a broadcast domain by default. It is possible but impractical to setup DHCP and DNS servers on all subnets. IP Helper addresses are normally used for this situation. **Note:** By default, the destination IP address – 255.255.255.255, which is sent as a link-layer broadcast – FFFF.FFFF.FFFF, are not forwarded beyond a broadcast domain.
- The **ip helper-address** {*ip-addr*} interface subcommand relays (accepts and forwards) UDP broadcasts as unicasts to a specified IP address – a server on another network or as directed broadcasts to a specified subnet. UDP broadcasts are forwarded to **all helper addresses** as soon as they are being received by the router interface. A Cisco router configured with the **ip helper-address** command is known as a DHCP Relay Agent.
- Configure the **ip helper-address** command on the router interface that will **receive** the UDP broadcast requests. Be default, the command relays the following 9 UDP services:

Service	UDP Port Number
Time	37
TACACS	49
DNS	53
BOOTP/DHCP Server	67
BOOTP/DHCP Client	68
TFTP	69
NetBIOS Name Service	137
NetBIOS Datagram Service	138
IEN 116 name service (obsolete)	42

**IEN** – Internet Engineering Notes (<http://www.potaroo.net/ietf/html/ienindex.html>)

- Use the [**no**] **ip forward-protocol udp** {*port-num*} global configuration command to add or remove an additional / unwanted UDP service to / from the default group respectively.
- **Note:** All broadcasts are sent to all configured helper addresses, regardless of whether a server is able to process a particular request. This could consume unnecessary bandwidth on the network. Ex: ServerA is a DHCP server, ServerB is a DNS server. IP addresses for both servers have been configured as helper addresses. DHCP and DNS broadcast requests are sent to both servers. An ICMP Port Unreachable error message will be generated when a server is unable to serve a particular type of request.
- Additional helper addresses are not required on intermediate routers along a path, as the first router has modified the destination address from broadcast to unicast or directed broadcast.
- Configuring the **ip helper-address** command is often sufficient to obtain the desired results. In some large networks, it may be necessary for the DHCP relay agent (eg: Cisco router) to insert additional information in the DHCP requests being forwarded to the central DHCP server. Such information is useful for statistical analysis, as well as allows DHCP to make decisions based on where the DHCP request is coming from and even which user is making the request. This is achieved by enabling the **DHCP Option 82 – DHCP Relay Agent Information Option**.

- The DHCP Relay Agent capability is enabled by default. The **ip helper-address** command does not enable the DHCP Relay Agent; it is being used to configure the DHCP Relay Agent where to relay the broadcast DHCP requests.



**Figure A6-9: IP Helper Address with Multiple Servers**

- IP Helper Address configuration on RT1:

```

RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#int e0
RT1(config-if)#ip helper-address 172.16.0.255
RT1(config-if)#ip helper-address 172.16.1.2
RT1(config-if)#exit
RT1(config)#int e1
RT1(config-if)#ip directed-broadcast
RT1(config-if)#^Z
RT1#
RT1#sh ip int e0
Ethernet0 is up, line protocol is up
  Internet address is 10.10.10.1/24
  Helper addresses are 172.16.0.255
                     172.16.1.2

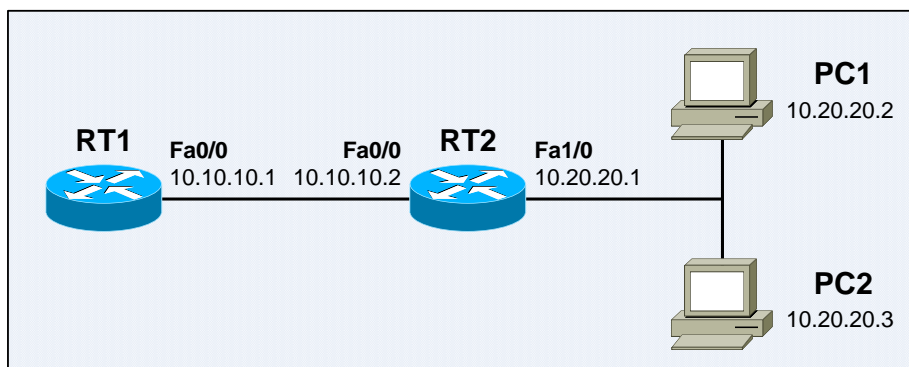
  Directed broadcast forwarding is disabled
--- output omitted ---
RT1#sh ip int e1
Ethernet1 is up, line protocol is up
  Internet address is 172.16.0.1/24
  Helper addresses is not set
  Directed broadcast forwarding is enabled
--- output omitted ---

```

- Configuring a helper address destined to the directed broadcast to the server farm subnet (172.16.0.0/24) is more efficient than configuring multiple unicast helper addresses when there are multiple servers reside in the same subnet.
- The **ip directed-broadcast** interface subcommand enables directed broadcast forwarding on an interface. This feature is disabled (directed broadcasts are dropped) by default in Cisco IOS Release 12.0 and later due to the default **no ip directed-broadcast** command.

## Directed Broadcast Forwarding

- A **directed broadcast** is a broadcast intended for all nodes on a **non-local** network.  
**Note:** The **ip directed-broadcast** interface subcommand which enables Directed Broadcast Forwarding for an interface only forwards directed broadcast packets from remote networks.  
Ex: By enabling DBF on RT2 Fa1/0, RT2 would only forwards directed broadcast traffic sourced from networks other than 10.20.20.0.



**Figure A6-10:** Network Setup for Directed Broadcast Forwarding

- A packet destined to the directed broadcast address 10.20.20.255 is intended for all nodes whose network address is 10.20.20.0 (PC1 and PC2). A router which does not directly connect to 10.20.20.0 (RT1) simply forwards the directed broadcast address packet to its next hop (RT2, 10.10.10.2). A router on network 10.20.20.0 (RT2) which has directed broadcast forwarding enabled on Fa1/0 would accept and forward the packet to all nodes in 10.20.20.0.
- Below shows the output of the **debug ip packet** privileged command on RT1 and RT2 before DBF is enabled on RT2 Fa1/0. PC1 and PC2 do not response when RT1 ping 10.20.20.255.

```
RT2#sh run int fa1/0
Building configuration...
!
interface FastEthernet1/0
 ip address 10.20.20.1 255.255.255.0
end
RT2#sh ip int fa1/0
FastEthernet1/0 is up, line protocol is up
 Internet address is 10.20.20.1/24
 Directed broadcast forwarding is disabled
--- output omitted ---
-----
RT1#ping 10.20.20.255 repeat 1
Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 10.20.20.255, timeout is 2 seconds:
!
Success rate is 100 percent (1/1), round-trip min/avg/max = 44/44/44 ms
RT1#
00:07:21: IP: tableid=0, s=10.10.10.1 (local), d=10.20.20.255
(FastEthernet0/0), routed via RIB
00:07:21: IP: s=10.10.10.1 (local), d=10.20.20.255 (FastEthernet0/0), len
100, sending
00:07:21: IP: tableid=0, s=10.10.10.2 (FastEthernet0/0), d=10.10.10.1
(FastEthernet0/0), routed via RIB
00:07:21: IP: s=10.10.10.2 (FastEthernet0/0), d=10.10.10.1
(FastEthernet0/0), len 100, rcvd 3
```

```

RT2#
00:07:21: IP: tableid=0, s=10.10.10.1 (FastEthernet0/0), d=10.20.20.255
(FastEthernet1/0), routed via RIB
00:07:21: IP: s=10.10.10.1 (FastEthernet0/0), d=10.20.20.255
(FastEthernet1/0), len100, rcvd 5
00:07:21: IP: tableid=0, s=10.10.10.2 (local), d=10.10.10.1
(FastEthernet0/0), routed via RIB
00:07:21: IP: s=10.10.10.2 (local), d=10.10.10.1 (FastEthernet0/0), len 100,
sending
RT2#

```

- Below shows the output of the **debug ip packet** privileged command on RT1 and RT2 after DBF is enabled on RT2 Fa1/0. PC1 and PC2 response to RT1 when RT1 ping 10.20.20.255.

```

RT2#sh run int fa1/0
Building configuration...
!
interface FastEthernet1/0
 ip address 10.20.20.1 255.255.255.0
 ip directed-broadcast
end
RT2#sh ip int fa1/0
FastEthernet1/0 is up, line protocol is up
 Internet address is 10.20.20.1/24
 Directed broadcast forwarding is enabled
--- output omitted ---
-----
RT1#ping 10.20.20.255 repeat 1
Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 10.20.20.255, timeout is 2 seconds:
!
Success rate is 100 percent (1/1), round-trip min/avg/max = 40/40/40 ms
RT1#
00:10:34: IP: tableid=0, s=10.10.10.1 (local), d=10.20.20.255
(FastEthernet0/0), routed via RIB
00:10:34: IP: s=10.10.10.1 (local), d=10.20.20.255 (FastEthernet0/0), len
100, sending
00:10:34: IP: tableid=0, s=10.10.10.2 (FastEthernet0/0), d=10.10.10.1
(FastEthernet0/0), routed via RIB
00:10:34: IP: s=10.10.10.2 (FastEthernet0/0), d=10.10.10.1
(FastEthernet0/0), len 100, rcvd 3
00:10:34: IP: tableid=0, s=10.20.20.2 (FastEthernet0/0), d=10.10.10.1
(FastEthernet0/0), routed via RIB
00:10:34: IP: s=10.20.20.2 (FastEthernet0/0), d=10.10.10.1
(FastEthernet0/0), len 100, rcvd 3
00:10:34: IP: tableid=0, s=10.20.20.3 (FastEthernet0/0), d=10.10.10.1
(FastEthernet0/0), routed via RIB
00:10:34: IP: s=10.20.20.3 (FastEthernet0/0), d=10.10.10.1
(FastEthernet0/0), len 100, rcvd 3
RT1#

```

```

RT2#
00:10:34: IP: tableid=0, s=10.10.10.1 (FastEthernet0/0), d=10.20.20.255
(FastEthernet1/0), routed via RIB
00:10:34: IP: s=10.10.10.1 (FastEthernet0/0), d=10.20.20.255
(FastEthernet1/0), g=255.255.255.255, len 100, forward directed broadcast
00:10:34: IP: s=10.10.10.1 (FastEthernet0/0), d=10.20.20.255
(FastEthernet1/0), len100, rcvd 5
00:10:34: IP: tableid=0, s=10.10.10.2 (local), d=10.10.10.1
(FastEthernet0/0), routed via RIB
00:10:34: IP: s=10.10.10.2 (local), d=10.10.10.1 (FastEthernet0/0), len 100,
sending
00:10:34: IP: tableid=0, s=10.20.20.2 (FastEthernet1/0), d=10.10.10.1
(FastEthernet0/0), routed via RIB
00:10:34: IP: s=10.20.20.2 (FastEthernet1/0), d=10.10.10.1
(FastEthernet0/0), g=10.10.10.1, len 100, forward
00:10:34: IP: tableid=0, s=10.20.20.3 (FastEthernet1/0), d=10.10.10.1
(FastEthernet0/0), routed via RIB
00:10:34: IP: s=10.20.20.3 (FastEthernet1/0), d=10.10.10.1
(FastEthernet0/0), g=10.10.10.1, len 100, forward
RT2#
-----
PC1#
xxxx: IP: s=10.10.10.1 (FastEthernet0/0), d=255.255.255.255, len 100, rcvd 2
xxxx: IP: s=10.20.20.2 (local), d=10.10.10.1 (FastEthernet0/0), len 100,
sending
PC1#
-----
PC2#
xxxx: IP: s=10.10.10.1 (FastEthernet0/0), d=255.255.255.255, len 100, rcvd 2
xxxx: IP: s=10.20.20.3 (local), d=10.10.10.1 (FastEthernet0/0), len 100,
sending
PC2#

```

- Below shows the output of the **debug ip packet** privileged command on RT2 when RT2 ping 10.20.20.255. The result is same regardless of the DBF configuration on RT2 Fa1/0.

```

RT2#ping 10.20.20.255 repeat 1
Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 10.20.20.255, timeout is 2 seconds:

00:14:11: IP: s=10.20.20.1 (local), d=255.255.255.255 (FastEthernet1/0), len
100, sending broad/multicast
00:14:11: IP: tableid=0, s=10.20.20.2 (FastEthernet1/0), d=10.20.20.1
(FastEthernet1/0), routed via RIB
00:14:11: IP: s=10.20.20.2 (FastEthernet1/0), d=10.20.20.1
(FastEthernet1/0), len 100, rcvd 3
00:14:11: IP: tableid=0, s=10.20.20.3 (FastEthernet1/0), d=10.20.20.1
(FastEthernet1/0), routed via RIB
00:14:11: IP: s=10.20.20.3 (FastEthernet1/0), d=10.20.20.1
(FastEthernet1/0), len 100, rcvd 3
Reply to request 0 from 10.20.20.2, 24 ms
Reply to request 0 from 10.20.20.3, 40 ms
RT2#
-----
PC1#
xxxx: IP: s=10.20.20.1 (FastEthernet0/0), d=255.255.255.255, len 100, rcvd 2
xxxx: IP: s=10.20.20.2 (local), d=10.20.20.1 (FastEthernet0/0), len 100,
sending
PC1#

```

- Below shows the output of the **debug ip packet** privileged command on RT2 when RT1 ping 255.255.255.255. PC1 and PC2 do not reply while RT2 reply when RT1 ping 255.255.255.255.

```
RT1#ping 255.255.255.255 rep 1
Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 255.255.255.255, timeout is 2 seconds:

00:14:30: IP: s=10.10.10.1 (local), d=255.255.255.255 (FastEthernet0/0), len
100, sending broad/multicast
00:14:30: IP: tableid=0, s=10.10.10.2 (FastEthernet0/0), d=10.10.10.1
(FastEthernet0/0), routed via RIB
00:14:30: IP: s=10.10.10.2 (FastEthernet0/0), d=10.10.10.1
(FastEthernet0/0), len 100, rcvd 3
Reply to request 0 from 10.10.10.2, 84 ms
RT1#
-----
RT2#
00:14:30: IP: s=10.10.10.1 (FastEthernet0/0), d=255.255.255.255, len 100,
rcvd 2
00:14:30: IP: tableid=0, s=10.10.10.2 (local), d=10.10.10.1
(FastEthernet0/0), routed via RIB
00:14:30: IP: s=10.10.10.2 (local), d=10.10.10.1 (FastEthernet0/0), len 100,
sending
RT2#
```

- DoS (Denial-of-Service) attacks, eg: Smurf and Fraggle take advantage of directed broadcasts. Smurf attacks send a large number of ICMP Echo Request packets with a spoofed source address (victim) to a directed broadcast and cause all hosts to respond to the Echo Request packets, results in wasting network bandwidth resources and unnecessary processing on the victim host. As a result, it is recommended to disable Directed Broadcast Forwarding on any interface where directed broadcasts are not needed.

## Bridging Cisco Router Interfaces for High Availability LAN

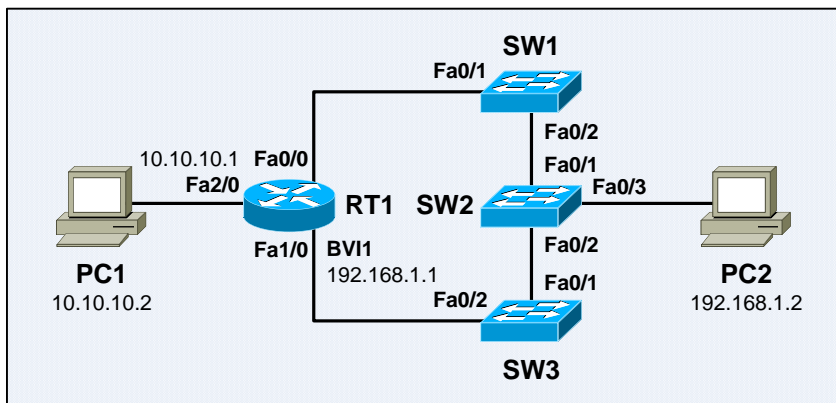


Figure A6-11: Bridging Cisco Router Interfaces for High Availability LAN

- The figure above shows a sample High Availability LAN. It eliminates the single point of failure in cascading switches topology in which a switch failure would result in total outage of the LAN.
- By using **Integrated Routing and Bridging (IRB)** technique, a Cisco router can be turned into a L3 switch. IP addresses can be assigned on **Bridged-Group Virtual Interfaces (BVIs)**, similar to VLAN interfaces as in L3 switches. A BVI is a virtual routed interface that has all network layer attributes, eg: a network address can be assigned to it, able to perform filtering, and does not support bridging.
- **Bridge groups** are defined by a unique number and are used for router bridging configuration. Network traffic is bridged between all interfaces that belong to the same bridge group.
- Integrated Routing and Bridging configuration on RT1:

```

RT1 (config) #int fa0/0
RT1 (config-if) #bridge-group 1
RT1 (config-if) #no shut
RT1 (config-if) #exit
RT1 (config) #
RT1 (config-if) #int fa1/0
RT1 (config-if) #bridge-group 1
RT1 (config-if) #no shut
RT1 (config-if) #exit
RT1 (config) #
RT1 (config) #int bvi1
Integrated Routing and Bridging is not configured!
RT1 (config) #bridge irb
RT1 (config) #int bvi1
RT1 (config-if) #ip add 192.168.1.1 255.255.255.0
RT1 (config-if) #exit
RT1 (config) #bridge 1 protocol ieee
RT1 (config) #bridge 1 route ip
RT1 (config) #^Z
    
```

**Note:** The **bridge 1 protocol ieee** global configuration command removes **bridge-group 1 spanning-disabled** interface subcommand on Fa0/0 and Fa1/0 and hence enables STP.

RT1#**sh spanning-tree brief**

```
Bridge group 1
Spanning tree enabled protocol ieee
Root ID    Priority    32768
           Address    cc01.01e0.0000
           This bridge is the root
           Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec

Bridge ID  Priority    32768
           Address    cc01.01e0.0000
           Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec
           Aging Time 300
```

Interface Name	Port ID	Prio	Cost	Sts	Designated Cost	Bridge ID	Port
FastEthernet0/0	128.2	128	19	FWD	0 32768	cc01.01e0.0000	128.2
FastEthernet1/0	128.3	128	19	FWD	0 32768	cc01.01e0.0000	128.3

RT1#  
RT1#**sh bridge group**

```
Bridge Group 1 is running the IEEE compatible Spanning Tree protocol

Port 2 (FastEthernet0/0) of bridge group 1 is forwarding
Port 3 (FastEthernet1/0) of bridge group 1 is forwarding
```

RT1#  
RT1#**sh int irb**

FastEthernet0/0

**Routed protocols on FastEthernet0/0:**

ip

```
Bridged protocols on FastEthernet0/0:
appletalk  clns      ip
```

--- output omitted ---

FastEthernet1/0

**Routed protocols on FastEthernet1/0:**

ip

```
Bridged protocols on FastEthernet1/0:
appletalk  clns      ip
```

--- output omitted ---

FastEthernet2/0

**Routed protocols on FastEthernet2/0:**

ip

BVI1

**Routed protocols on BVI1:**

ip

RT1#



## The Pitfall of Bridge Group

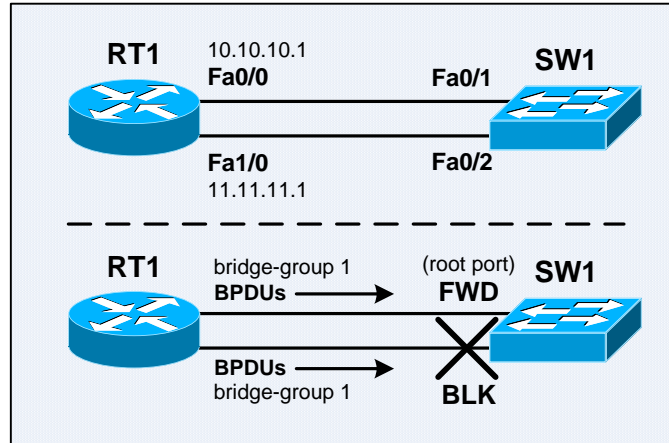


Figure A6-12: The Pitfall of Bridge Group

- This problem scenario is encountered during the migration of dummy switches to Cisco switches. The networks are running fine with dummy switches in which spanning tree is not running. When the LAN switches are replaced with new Cisco switches, the 11.11.11.0/24 network begins to become unavailable. RT1 forwards packets destined to end systems reside in the 11.11.11.0/24 network out its Fa1/0, but the frames are ignored and discarded when they arrived as SW1 Fa0/2 due to spanning tree blocking state on SW1 Fa0/2.
- Below shows the network setup of RT1 and SW1 before the **bridge-group** configuration:

```

RT1#sh cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID          Local Intrfce    Holdtme    Capability    Platform    Port ID
SW1                 Fas 0/0         143        S             3640        Fas 0/1
SW1                 Fas 1/0         143        S             3640        Fas 0/2
RT1#
=====

SW1#sh spanning-tree brief

VLAN1
  Spanning tree enabled protocol ieee
  Root ID          Priority    32768
                 Address    cc01.0870.0000
                 This bridge is the root
                 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

  Bridge ID       Priority    32768
                 Address    cc01.0870.0000
                 Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec
                 Aging Time 0

Interface
Name              Port ID Prio Cost    Sts Cost  Bridge ID          Port ID
-----
FastEthernet0/1  128.2  128   19 FWD    0 32768 cc01.0870.0000 128.2
FastEthernet0/2  128.3  128   19 FWD    0 32768 cc01.0870.0000 128.3

SW1#debug spanning-tree events
Spanning Tree event debugging is on
SW1#

```

- Below shows the spanning tree convergence for the **bridge-group** configuration on RT1 Fa0/0:

```

RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#bridge 1 protocol ieee
RT1(config)#int fa0/0
RT1(config-if)#bridge-group 1
=====
SW1#
00:17:07: STP: VLAN1 heard root 32768-cc00.0870.0000 on Fa0/1
00:17:07: current Root has 32768-cc01.0870.0000
00:17:07:      supersedes 32768-cc01.0870.0000
00:17:07: STP: VLAN1 new root is 32768, cc00.0870.0000 on port Fa0/1, cost
19
00:17:28: STP: VLAN1 we are the spanning tree root
00:17:29: STP: VLAN1 heard root 32768-cc00.0870.0000 on Fa0/1
00:17:29: current Root has 32768-cc01.0870.0000
00:17:29:      supersedes 32768-cc01.0870.0000
00:17:29: STP: VLAN1 new root is 32768, cc00.0870.0000 on port Fa0/1, cost
19
00:17:29: STP: VLAN1 sent Topology Change Notice on Fa0/1
SW1#
SW1#sh spanning-tree brief

VLAN1
  Spanning tree enabled protocol ieee
  Root ID      Priority      32768
              Address      cc00.0870.0000
              Cost        19
              Port        2 (FastEthernet0/1)
  Hello Time   2 sec    Max Age 20 sec    Forward Delay 15 sec

  Bridge ID    Priority      32768
              Address      cc01.0870.0000
              Hello Time 2 sec    Max Age 20 sec    Forward Delay 15 sec
              Aging Time 0

Interface                               Designated
Name                                     Port ID Prio Cost  Sts Cost  Bridge ID                               Port ID
-----
FastEthernet0/1                         128.2   128   19 FWD   0 32768 cc00.0870.0000 128.2
FastEthernet0/2                         128.3   128   19 FWD   19 32768 cc01.0870.0000 128.3

SW1#

```

- Below shows the spanning tree convergence for the **bridge-group** configuration on RT1 Fa1/0:

```

RT1(config-if)#int fa1/0
RT1(config-if)#bridge-group 1
=====
SW1#
00:20:08: STP: VLAN1 sent Topology Change Notice on Fa0/1
00:20:08: STP: VLAN1 Fa0/2 -> blocking
SW1#sh spanning-tree brief

VLAN1
  Spanning tree enabled protocol ieee
  Root ID    Priority    32768
            Address    cc00.0870.0000
            Cost      19
            Port      2 (FastEthernet0/1)
            Hello Time 2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32768
            Address    cc01.0870.0000
            Hello Time 2 sec  Max Age 20 sec  Forward Delay 15 sec
            Aging Time 0

Interface
Name          Port ID Prio Cost  Sts Cost  Bridge ID          Port ID
-----
FastEthernet0/1  128.2  128   19  FWD   0 32768 cc00.0870.0000 128.2
FastEthernet0/2  128.3  128   19  BLK   0 32768 cc00.0870.0000 128.3

SW1#

```

- Below shows that after removed the **bridge-group** configuration on RT1 Fa1/0, SW1 Fa0/2 would wait for the MaxAge timer (20 seconds) to expire before starting the STP convergence:

```

RT1#sh clock
00:22:17 UTC Mon Mar 1 1993
RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#int fa1/0
RT1(config-if)#no bridge-group 1
RT1(config-if)#
=====
SW1#
00:22:36: STP: VLAN1 Fa0/2 -> listening
00:22:51: STP: VLAN1 Fa0/2 -> learning
00:23:06: STP: VLAN1 sent Topology Change Notice on Fa0/1
00:23:06: STP: VLAN1 Fa0/2 -> forwarding
SW1#

```

## EIGRP Metric Calculation

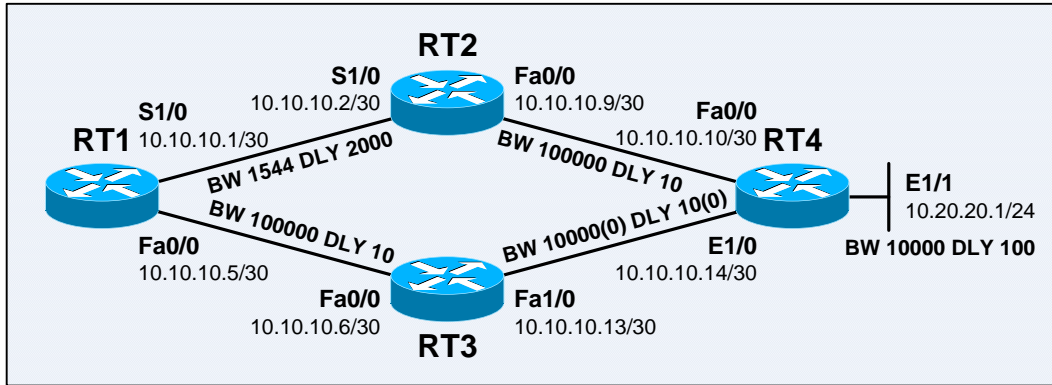


Figure A6-13: Network Setup for EIGRP Metric Calculation

- The figure above shows the sample network used for the discussion of EIGRP metric calculation. **Note:** RT3 treats the RT3 – RT4 link (10.10.10.12/30) as Fast Ethernet (BW 100000 DLY 10, metric 28160) while RT4 treats the link as Ethernet (BW 10000 DLY 100, metric 28160).
- The destination network in question is 10.20.20.0/24, the network attached on RT4 E1/1.
- The simplified EIGRP metric calculation formula is  $\left[ \left( \frac{10000000}{\text{minimum bandwidth}} \right) + \text{total delay} \right] \times 256$
- Below shows the EIGRP topology table on RT4:

```
RT4#sh int | in Ethernet|Internet address|MTU
FastEthernet0/0 is up, line protocol is up
  Internet address is 10.10.10.10/30
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
Ethernet1/0 is up, line protocol is up
  Internet address is 10.10.10.14/30
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,
Ethernet1/1 is up, line protocol is up
  Internet address is 10.20.20.1/24
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,
--- output omitted ---
RT4#
RT4#sh ip eigrp topology
IP-EIGRP Topology Table for AS(100)/ID(10.2.2.1)

P 10.10.10.0/30, 1 successors, FD is 2172416
   via 10.10.10.9 (2172416/2169856), FastEthernet0/0
P 10.10.10.4/30, 1 successors, FD is 284160
   via 10.10.10.13 (284160/28160), Ethernet1/0
P 10.10.10.8/30, 1 successors, FD is 28160
   via Connected, FastEthernet0/0
P 10.10.10.12/30, 1 successors, FD is 281600
   via Connected, Ethernet1/0
P 10.20.20.0/24, 1 successors, FD is 281600
   via Connected, Ethernet1/1
RT4#
```

- The EIGRP metric for RT4 to reach 10.10.10.8/30 is  $\left[ \left( \frac{10000000}{100000} \right) + 10 \right] \times 256 = 110 \times 256 = \mathbf{28160}$
- The EIGRP metric for RT4 to reach 10.20.20.0/24 is  $\left[ \left( \frac{10000000}{10000} \right) + 100 \right] \times 256 = 1100 \times 256 = \mathbf{281600}$

- Below shows the EIGRP topology table on RT2:

```

RT2#sh int s1/0
Serial1/0 is up, line protocol is up
  Internet address is 10.10.10.2/30
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
  --- output omitted ---
RT2#sh ip eigrp topology
IP-EIGRP Topology Table for AS(100)/ID(10.1.1.9)

P 10.10.10.0/30, 1 successors, FD is 2169856
    via Connected, Serial1/0
P 10.10.10.4/30, 1 successors, FD is 286720
    via 10.10.10.10 (286720/284160), FastEthernet0/0
    via 10.10.10.1 (2172416/28160), Serial1/0
P 10.10.10.8/30, 1 successors, FD is 28160
    via Connected, FastEthernet0/0
P 10.10.10.12/30, 1 successors, FD is 284160
    via 10.10.10.10 (284160/281600), FastEthernet0/0
    via 10.10.10.1 (2174976/30720), Serial1/0
P 10.20.20.0/24, 1 successors, FD is 284160
    via 10.10.10.10 (284160/281600), FastEthernet0/0
RT2#

```

- The EIGRP metric from RT2 to reach 10.10.10.0/30 is  $\left[ \left( \frac{10000000}{1544} \right) + 2000 \right] \times 256 = 8476 \times 256 = \mathbf{2169856}$
- The EIGRP metric from RT2 to reach 10.20.20.0/24 is  $\left[ \left( \frac{10000000}{1000} \right) + 110 \right] \times 256 = 1110 \times 256 = \mathbf{284160}$
- Cisco IOS uses the floor() function as in C programming language when rounding floating-point numbers into decimal numbers, in which a floating-point number is rounded **downward** to a decimal number. Ex:  $10000000 / 1544 = \mathbf{6476.68} \rightarrow \mathbf{6476}$ .
- When RT2 calculates the EIGRP metric to 10.10.10.4/30, the bandwidth and delay between RT3 and RT4 is 10000 and 100 respectively (Ethernet), as advertised by RT4 to RT2.

The EIGRP metric from RT2 to reach 10.10.10.4/30 is  $\left[ \left( \frac{10000000}{10000} \right) + 120 \right] \times 256 = 1120 \times 256 = \mathbf{286720}$

- Below shows the EIGRP topology table on RT1.

```

RT1#sh ip eigrp topology
IP-EIGRP Topology Table for AS(100)/ID(10.10.10.5)

--- output omitted ---
P 10.20.20.0/24, 1 successors, FD is 286720
    via 10.10.10.6 (286720/284160), FastEthernet0/0
    via 10.10.10.2 (2198016/284160), Serial1/0
RT1#

```

- The EIGRP metric from RT1 to reach 10.20.20.0/24 via RT3 is  $\left[ \left( \frac{10000000}{10000} \right) + 120 \right] \times 256 = 1120 \times 256 = \mathbf{286720}$
- 10.10.10.6 is the successor while 10.10.10.2 is the feasible successor to 10.20.20.0/24, as the AD of the FS is 284160, which is less than the FD – 286720.

## The Impact of Minimum Bandwidth upon EIGRP Metric Calculation

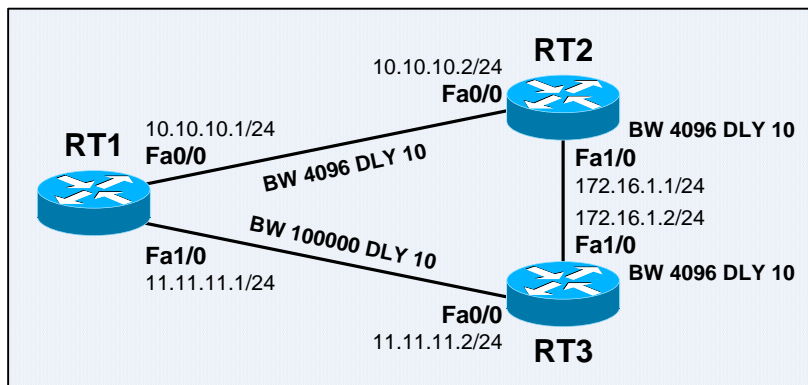


Figure A6-14: The Impact of Minimum Bandwidth upon EIGRP Metric Calculation

- Below shows that EIGRP running on RT1 considers the parallel paths to 172.16.1.0/24 via RT2 and RT3 are equal; while OSPF considers the path via RT3 is better (lower metric or cost).

```

RT1#sh cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID          Local Intrfce    Holdtme    Capability    Platform    Port ID
RT2                 Fas 0/0         148        R             3620        Fas 0/0
RT3                 Fas 1/0         160        R             3620        Fas 0/0
RT1#
RT1#sh ip route eigrp
 172.16.0.0/24 is subnetted, 1 subnets
D       172.16.1.0 [90/630016] via 10.10.10.2, 00:00:23, FastEthernet0/0
        [90/630016] via 11.11.11.2, 00:00:23, FastEthernet1/0
RT1#
RT1#sh ip eigrp topology 172.16.1.0/24
IP-EIGRP (AS 10): Topology entry for 172.16.1.0/24
State is Passive, Query origin flag is 1, 2 Successor(s), FD is 630016
Routing Descriptor Blocks:
 11.11.11.2 (FastEthernet1/0), from 11.11.11.2, Send flag is 0x0
  Composite metric is (630016/627456), Route is Internal
  Vector metric:
    Minimum bandwidth is 4096 Kbit
    Total delay is 200 microseconds
    Reliability is 255/255
    Load is 1/255
    Minimum MTU is 1500
    Hop count is 1
 10.10.10.2 (FastEthernet0/0), from 10.10.10.2, Send flag is 0x0
  Composite metric is (630016/627456), Route is Internal
  Vector metric:
    Minimum bandwidth is 4096 Kbit
    Total delay is 200 microseconds
    Reliability is 255/255
    Load is 1/255
    Minimum MTU is 1500
    Hop count is 1
RT1#
RT1#sh int | in line protocol|DLY
FastEthernet0/0 is up, line protocol is up
  MTU 1500 bytes, BW 4096 Kbit, DLY 100 usec,
FastEthernet1/0 is up, line protocol is up
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
    
```

# EIGRP Feasible Successor

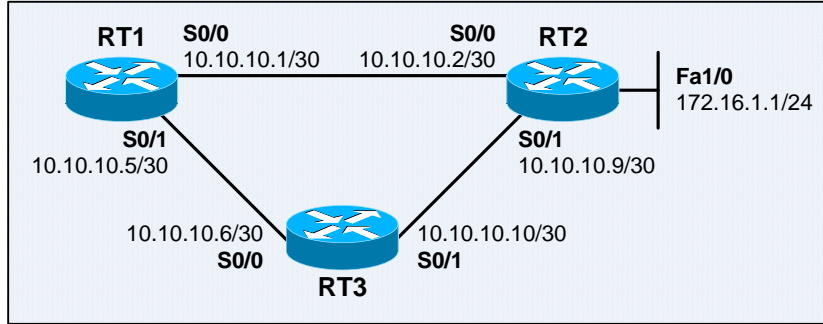


Figure A6-15: EIGRP Feasible Successor

- Below shows the routing table and EIGRP topology table on RT1:

```

RT1#sh ip route

Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 1 subnets
D       172.16.1.0 [90/2172416] via 10.10.10.2, 00:00:03, Serial0/0
    10.0.0.0/30 is subnetted, 3 subnets
D       10.10.10.8 [90/2681856] via 10.10.10.2, 00:00:05, Serial0/0
        [90/2681856] via 10.10.10.6, 00:00:05, Serial0/1
C       10.10.10.0 is directly connected, Serial0/0
C       10.10.10.4 is directly connected, Serial0/1
RT1#
RT1#sh ip eigrp topology
IP-EIGRP Topology Table for AS(100)/ID(10.10.10.5)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.10.10.8/30, 2 successors, FD is 2681856
   via 10.10.10.6 (2681856/2169856), Serial0/1
   via 10.10.10.2 (2681856/2169856), Serial0/0
P 10.10.10.0/30, 1 successors, FD is 2169856
   via Connected, Serial0/0
P 10.10.10.4/30, 1 successors, FD is 2169856
   via Connected, Serial0/1
P 172.16.1.0/24, 1 successors, FD is 2172416
   via 10.10.10.2 (2172416/28160), Serial0/0
RT1#
RT1#sh ip eigrp topology all-links
IP-EIGRP Topology Table for AS(100)/ID(10.10.10.5)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.10.10.8/30, 2 successors, FD is 2681856, serno 6
   via 10.10.10.6 (2681856/2169856), Serial0/1
   via 10.10.10.2 (2681856/2169856), Serial0/0
P 10.10.10.0/30, 1 successors, FD is 2169856, serno 3
   via Connected, Serial0/0
P 10.10.10.4/30, 1 successors, FD is 2169856, serno 2
   via Connected, Serial0/1
P 172.16.1.0/24, 1 successors, FD is 2172416, serno 4
   via 10.10.10.2 (2172416/28160), Serial0/0
   via 10.10.10.6 (2684416/2172416), Serial0/1
RT1#
    
```

```

RT1#sh ip eigrp topo 172.16.1.0/24
IP-EIGRP (AS 100): Topology entry for 172.16.1.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 2172416
  Routing Descriptor Blocks:
    10.10.10.2 (Serial0/0), from 10.10.10.2, Send flag is 0x0
      Composite metric is (2172416/28160), Route is Internal
      Vector metric:
        Minimum bandwidth is 1544 Kbit
        Total delay is 20100 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
    10.10.10.6 (Serial0/1), from 10.10.10.6, Send flag is 0x0
      Composite metric is (2684416/2172416), Route is Internal
      Vector metric:
        Minimum bandwidth is 1544 Kbit
        Total delay is 40100 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2

RT1#debug ip eigrp
IP-EIGRP Route Events debugging is on
RT1#debug ip routing
IP routing debugging is on
RT1#debug eigrp fsm
EIGRP FSM Events/Actions debugging is on
RT1#

```

**Note:** The advertised distance from RT3 (10.10.10.6) to reach 172.16.1.0/24 is **2172416**, which is same and not less than RT1's feasible distance. Therefore it is not considered as a FS.

- Below shows the EIGRP topology table on RT3 and modifying the delay value of S0/1.

```

RT3#sh ip eigrp topo
IP-EIGRP Topology Table for AS(100)/ID(10.10.10.10)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.10.10.8/30, 1 successors, FD is 2169856
   via Connected, Serial0/1
P 10.10.10.0/30, 2 successors, FD is 2681856
   via 10.10.10.9 (2681856/2169856), Serial0/1
   via 10.10.10.5 (2681856/2169856), Serial0/0
P 10.10.10.4/30, 1 successors, FD is 2169856
   via Connected, Serial0/0
P 172.16.1.0/24, 1 successors, FD is 2172416
   via 10.10.10.9 (2172416/28160), Serial0/1

RT3#
RT3#sh int s0/1 | in DLY
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
RT3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT3(config)#int s0/1
RT3(config-if)#delay 1999
RT3(config-if)#do sh int s0/1 | in DLY
  MTU 1500 bytes, BW 1544 Kbit, DLY 19990 usec,
RT3(config-if)#

```



- Below shows the EIGRP convergence on RT1 upon changing the delay on RT3 S0/1.

```
RT1#
00:03:25: IP-EIGRP(Default-IP-Routing-Table:100): Processing incoming UPDATE
packet
00:03:25: IP-EIGRP(Default-IP-Routing-Table:100): Int 10.10.10.8/30 M
2681600 - 1657856 1023744 SM 2169600 - 1657856 511744
00:03:25: DUAL: rcvupdate: 10.10.10.8/30 via 10.10.10.6 metric
2681600/2169600
00:03:25: DUAL: Find FS for dest 10.10.10.8/30. FD is 2681856, RD is 2681856
00:03:25: DUAL:          10.10.10.6 metric 2681600/2169600
00:03:25: DUAL:          10.10.10.2 metric 2681856/2169856 found Dmin is
2681600
00:03:25: RT: metric change to 10.10.10.8 via 10.10.10.6, eigrp metric
[90/2681856]
      new metric [90/2681600]
00:03:25: RT: del 10.10.10.8/30 via 10.10.10.2, eigrp metric [90/2681856]
00:03:25: RT: NET-RED 10.10.10.8/30
00:03:25: RT: NET-RED 10.10.10.8/30
00:03:25: DUAL: RT installed 10.10.10.8/30 via 10.10.10.6
00:03:25: IP-EIGRP(Default-IP-Routing-Table:100): 10.10.10.8/30 routing
table not updated thru 10.10.10.2
00:03:25: DUAL: Send update about 10.10.10.8/30. Reason: metric chg
00:03:25: DUAL: Send update about 10.10.10.8/30. Reason: lost if
00:03:25: IP-EIGRP(Default-IP-Routing-Table:100): 10.10.10.8/30 - do
advertise out Serial0/0
00:03:25: IP-EIGRP(Default-IP-Routing-Table:100): Int 10.10.10.8/30 metric
2681600 - 1657856 1023744
00:03:25: IP-EIGRP(Default-IP-Routing-Table:100): Int 10.10.10.8/30 metric
2681600 - 1657856 1023744
00:03:35: IP-EIGRP(Default-IP-Routing-Table:100): Processing incoming UPDATE
packet
00:03:35: IP-EIGRP(Default-IP-Routing-Table:100): Int 10.10.10.0/30 M
3193600 - 1657856 1535744 SM 2681600 - 1657856 1023744
00:03:35: DUAL: dest(10.10.10.0/30) not active
00:03:35: DUAL: rcvupdate: 10.10.10.0/30 via 10.10.10.6 metric
3193600/2681600
00:03:35: DUAL: Find FS for dest 10.10.10.0/30. FD is 2169856, RD is 2169856
00:03:35: DUAL:          0.0.0.0 metric 2169856/0
00:03:35: DUAL:          10.10.10.6 metric 3193600/2681600 found Dmin is
2169856
00:03:35: IP-EIGRP(Default-IP-Routing-Table:100): 10.10.10.0/30 routing
table not updated thru 10.10.10.6
00:03:35: IP-EIGRP(Default-IP-Routing-Table:100): Int 172.16.1.0/24 M
2684160 - 1657856 1026304 SM 2172160 - 1657856 514304
00:03:35: DUAL: rcvupdate: 172.16.1.0/24 via 10.10.10.6 metric
2684160/2172160
00:03:35: DUAL: Find FS for dest 172.16.1.0/24. FD is 2172416, RD is 2172416
00:03:35: DUAL:          10.10.10.2 metric 2172416/28160
00:03:35: DUAL:          10.10.10.6 metric 2684160/2172160 found Dmin is
2172416
00:03:35: DUAL: RT installed 172.16.1.0/24 via 10.10.10.2
00:03:35: IP-EIGRP(Default-IP-Routing-Table:100): 172.16.1.0/24 routing
table not updated thru 10.10.10.6
RT1#
```

```

RT1#sh ip eigrp topo
IP-EIGRP Topology Table for AS(100)/ID(10.10.10.5)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.10.10.8/30, 1 successors, FD is 2681600
   via 10.10.10.6 (2681600/2169600), Serial0/1
   via 10.10.10.2 (2681856/2169856), Serial0/0
P 10.10.10.0/30, 1 successors, FD is 2169856
   via Connected, Serial0/0
P 10.10.10.4/30, 1 successors, FD is 2169856
   via Connected, Serial0/1
P 172.16.1.0/24, 1 successors, FD is 2172416
   via 10.10.10.2 (2172416/28160), Serial0/0
   via 10.10.10.6 (2684160/2172160), Serial0/1
RT1#

```

- The path to 172.16.1.0/24 from RT1 to RT2 via RT3 has become a feasible successor (backup) as the advertise distance from RT3, **2172160** is less than RT1's feasible distance, **2172416**.
- Below simulates a network connectivity problem by applying an access list to prevent RT2 EIGRP Hello packets from reaching to RT1 S0/0.

```

RT1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
RT1(config)#access-list 101 deny eigrp any host 224.0.0.10
RT1(config)#int s0/0
RT1(config-if)#ip access-group 101 in
RT1(config-if)#^Z
RT1#
RT1#sh ip eigrp neighbors
IP-EIGRP neighbors for process 100
H   Address                Interface      Hold Uptime    SRTT   RTO   Q   Seq
                               (sec)          (ms)          Cnt  Num
1   10.10.10.6              Se0/1         10 00:02:17    347   2082  0   10
0   10.10.10.2              Se0/0         5  00:02:32    486   2916  0   6
RT1#

```

- Below shows the output messages of the **debug ip eigrp**, **debug ip routing**, and **debug eigrp fsm** privileged command on RT1 for EIGRP convergence upon the connectivity problem. The EIGRP Query-Reply process did not occur as there is a FS to the network.

```

RT1#
00:03:09: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.2 (Serial0/0) is down: holding time expired
00:03:09: DUAL: linkdown: start - 10.10.10.2 via Serial0/0
00:03:09: DUAL: Destination 10.10.10.8/30
00:03:09: DUAL: Removing dest 10.10.10.8/30, nexthop 10.10.10.2
00:03:09: DUAL: Destination 10.10.10.0/30
00:03:09: DUAL: Destination 10.10.10.4/30
00:03:09: DUAL: Destination 172.16.1.0/24
00:03:09: DUAL: Find FS for dest 172.16.1.0/24. FD is 2172416, RD is 2172416
00:03:09: DUAL:      10.10.10.2 metric 4294967295/4294967295
00:03:09: DUAL:      10.10.10.6 metric 2684160/2172160 found Dmin is 2684160
00:03:09: RT: delete route to 172.16.1.0 via 10.10.10.2, eigrp metric [90/2172416]
00:03:09: RT: no routes to 172.16.1.0
00:03:09: RT: NET-RED 172.16.1.0/24
00:03:09: RT: delete subnet route to 172.16.1.0/24
00:03:09: RT: NET-RED 172.16.1.0/24
00:03:09: RT: delete network route to 172.16.0.0
00:03:09: RT: NET-RED 172.16.0.0/16
00:03:09: DUAL: Removing dest 172.16.1.0/24, nexthop 10.10.10.2
00:03:09: RT: add 172.16.1.0/24 via 10.10.10.6, eigrp metric [90/2684160]
00:03:09: RT: NET-RED 172.16.1.0/24
00:03:09: DUAL: RT installed 172.16.1.0/24 via 10.10.10.6
00:03:09: DUAL: Send update about 172.16.1.0/24. Reason: metric chg
00:03:09: DUAL: Send update about 172.16.1.0/24. Reason: new if
00:03:09: DUAL: linkdown: finish
00:03:09: IP-EIGRP(Default-IP-Routing-Table:100): Int 172.16.1.0/24 metric 2684160 - 1657856 1026304
RT1#
RT1#sh ip eigrp topo
IP-EIGRP Topology Table for AS(100)/ID(10.10.10.5)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

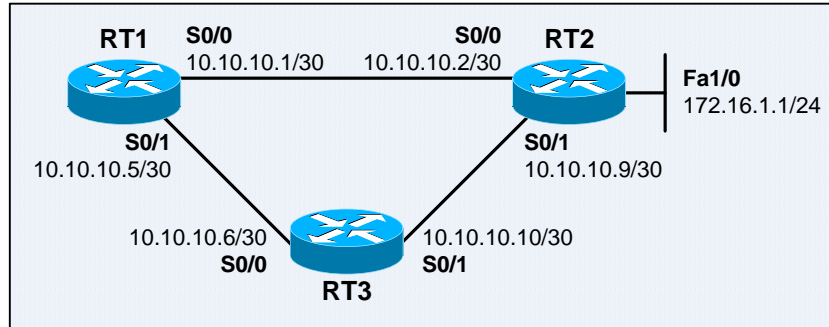
P 10.10.10.8/30, 1 successors, FD is 2681600
   via 10.10.10.6 (2681600/2169600), Serial0/1
P 10.10.10.0/30, 1 successors, FD is 2169856
   via Connected, Serial0/0
P 10.10.10.4/30, 1 successors, FD is 2169856
   via Connected, Serial0/1
P 172.16.1.0/24, 1 successors, FD is 2172416
   via 10.10.10.6 (2684160/2172160), Serial0/1
RT1#
RT1#sh ip route

Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 1 subnets
D    172.16.1.0 [90/2684160] via 10.10.10.6, 00:00:20, Serial0/1
    10.0.0.0/30 is subnetted, 3 subnets
D     10.10.10.8 [90/2681600] via 10.10.10.6, 00:01:30, Serial0/1
C     10.10.10.0 is directly connected, Serial0/0
C     10.10.10.4 is directly connected, Serial0/1
RT1#

```

## EIGRP DUAL Query-Reply Process



**Figure A6-16:** Network Setup for EIGRP Feasible Successor and DUAL Query-Reply Process

- Below shows the routing table and EIGRP topology table on RT1:

```
RT1#sh ip route
Gateway of last resort is not set

  172.16.0.0/24 is subnetted, 1 subnets
D       172.16.1.0 [90/2172416] via 10.10.10.2, 00:00:14, Serial0/0
  10.0.0.0/30 is subnetted, 3 subnets
D       10.10.10.8 [90/2681856] via 10.10.10.2, 00:00:15, Serial0/0
        [90/2681856] via 10.10.10.6, 00:00:15, Serial0/1
C       10.10.10.0 is directly connected, Serial0/0
C       10.10.10.4 is directly connected, Serial0/1
RT1#
RT1#sh ip eigrp topo
IP-EIGRP Topology Table for AS(100)/ID(10.10.10.5)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.10.10.8/30, 2 successors, FD is 2681856
   via 10.10.10.6 (2681856/2169856), Serial0/1
   via 10.10.10.2 (2681856/2169856), Serial0/0
P 10.10.10.0/30, 1 successors, FD is 2169856
   via Connected, Serial0/0
P 10.10.10.4/30, 1 successors, FD is 2169856
   via Connected, Serial0/1
P 172.16.1.0/24, 1 successors, FD is 2172416
   via 10.10.10.2 (2172416/28160), Serial0/0
RT1#
```

- RT1 has no feasible successor to 172.16.1.0/24. Hence it will query RT3 for a new successor when the link between RT1 and RT2 has failed.

- Below simulates a network connectivity problem by applying an access list to prevent RT2 EIGRP Hello packets from reaching to RT1 S0/0.

```

RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#access-list 101 deny eigrp any host 224.0.0.10
RT1(config)#int s0/0
RT1(config-if)#ip access-group 101 in
RT1(config-if)#^Z
RT1#
RT1#sh ip eigrp neighbors
IP-EIGRP neighbors for process 100
H   Address                Interface           Hold Uptime    SRTT   RTO   Q   Seq
                               (sec)              (ms)          Cnt  Num
1   10.10.10.6              Se0/1              13 00:02:06 1508  5000  0   4
0   10.10.10.2              Se0/0              5  00:02:20  400  2400  0   5
RT1#

```

**Note:** The EIGRP default Hello and hold time intervals are 5 seconds and 15 seconds respectively.

- Below shows the output messages of the **debug ip eigrp**, **debug ip routing** and **debug eigrp fsm** privileged command on RT1 for EIGRP convergence upon the connectivity problem. The EIGRP Query-Reply process occurred as there is no FS to the network.

```

RT1#
00:03:08: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.2
(Serial0/0) is down: holding time expired
00:03:08: DUAL: linkdown: start - 10.10.10.2 via Serial0/0
00:03:08: DUAL: Destination 10.10.10.8/30
00:03:08: DUAL: Find FS for dest 10.10.10.8/30. FD is 2681856, RD is 2681856
00:03:08: DUAL:          10.10.10.6 metric 2681856/2169856
00:03:08: DUAL:          10.10.10.2 metric 4294967295/4294967295 found Dmin
is 2681856
00:03:08: RT: delete route to 10.10.10.8 via 10.10.10.2, eigrp metric
[90/2681856]
00:03:08: RT: NET-RED 10.10.10.8/30
00:03:08: DUAL: Removing dest 10.10.10.8/30, nexthop 10.10.10.2
00:03:08: DUAL: RT installed 10.10.10.8/30 via 10.10.10.6
00:03:08: DUAL: Send update about 10.10.10.8/30. Reason: lost if
00:03:08: DUAL: Destination 10.10.10.0/30
00:03:08: DUAL: Destination 10.10.10.4/30
00:03:08: DUAL: Destination 172.16.1.0/24
00:03:08: DUAL: Find FS for dest 172.16.1.0/24. FD is 2172416, RD is 2172416
00:03:08: DUAL:          10.10.10.2 metric 4294967295/4294967295
00:03:08: DUAL:          10.10.10.6 metric 2684416/2172416 not found Dmin is
2684416
00:03:08: DUAL: Peer total/stub 1/0 template/full-stub 1/0
00:03:08: DUAL: Dest 172.16.1.0/24 entering active state.
00:03:08: DUAL: Set reply-status table. Count is 1.
00:03:08: DUAL: Not doing split horizon
00:03:08: DUAL: linkdown: finish
00:03:08: IP-EIGRP(Default-IP-Routing-Table:100): 172.16.1.0/24 - do
advertise out Serial0/1
00:03:08: IP-EIGRP(Default-IP-Routing-Table:100): Int 172.16.1.0/24 metric
4294967295 - 1657856 4294967295
00:03:08: IP-EIGRP(Default-IP-Routing-Table:100): Processing incoming REPLY
packet
00:03:08: IP-EIGRP(Default-IP-Routing-Table:100): Int 172.16.1.0/24 M
2684416 - 1657856 1026560 SM 2172416 - 1657856 514560

```

```

00:03:08: DUAL: rcvreply: 172.16.1.0/24 via 10.10.10.6 metric
2684416/2172416
00:03:08: DUAL: reply count is 1
00:03:08: DUAL: Clearing handle 1, count now 0
00:03:08: DUAL: Freeing reply status table
00:03:08: DUAL: Find FS for dest 172.16.1.0/24. FD is 4294967295, RD is
4294967295 found
00:03:08: RT: delete route to 172.16.1.0 via 10.10.10.2, eigrp metric
[90/2172416]
00:03:08: RT: no routes to 172.16.1.0
00:03:08: RT: NET-RED 172.16.1.0/24
00:03:08: RT: delete subnet route to 172.16.1.0/24
00:03:08: RT: NET-RED 172.16.1.0/24
00:03:08: RT: delete network route to 172.16.0.0
00:03:08: RT: NET-RED 172.16.0.0/16
00:03:08: DUAL: Removing dest 172.16.1.0/24, nexthop 10.10.10.2
00:03:08: RT: add 172.16.1.0/24 via 10.10.10.6, eigrp metric [90/2684416]
00:03:08: RT: NET-RED 172.16.1.0/24
00:03:08: DUAL: RT installed 172.16.1.0/24 via 10.10.10.6
00:03:08: DUAL: Send update about 172.16.1.0/24. Reason: metric chg
00:03:08: DUAL: Send update about 172.16.1.0/24. Reason: new if
00:03:08: IP-EIGRP(Default-IP-Routing-Table:100): Int 10.10.10.8/30 metric
2681856 - 1657856 1024000
00:03:08: IP-EIGRP(Default-IP-Routing-Table:100): Int 172.16.1.0/24 metric
2684416 - 1657856 1026560
RT1#
RT1#sh ip eigrp topo
IP-EIGRP Topology Table for AS(100)/ID(10.10.10.5)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.10.10.8/30, 1 successors, FD is 2681856
   via 10.10.10.6 (2681856/2169856), Serial0/1
P 10.10.10.0/30, 1 successors, FD is 2169856
   via Connected, Serial0/0
P 10.10.10.4/30, 1 successors, FD is 2169856
   via Connected, Serial0/1
P 172.16.1.0/24, 1 successors, FD is 2684416
   via 10.10.10.6 (2684416/2172416), Serial0/1
RT1#
RT1#sh ip route

Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 1 subnets
D    172.16.1.0 [90/2684416] via 10.10.10.6, 00:00:15, Serial0/1
    10.0.0.0/30 is subnetted, 3 subnets
D      10.10.10.8 [90/2681856] via 10.10.10.6, 00:00:15, Serial0/1
C      10.10.10.0 is directly connected, Serial0/0
C      10.10.10.4 is directly connected, Serial0/1
RT1#

```

## The EIGRP DUAL Finite State Machine

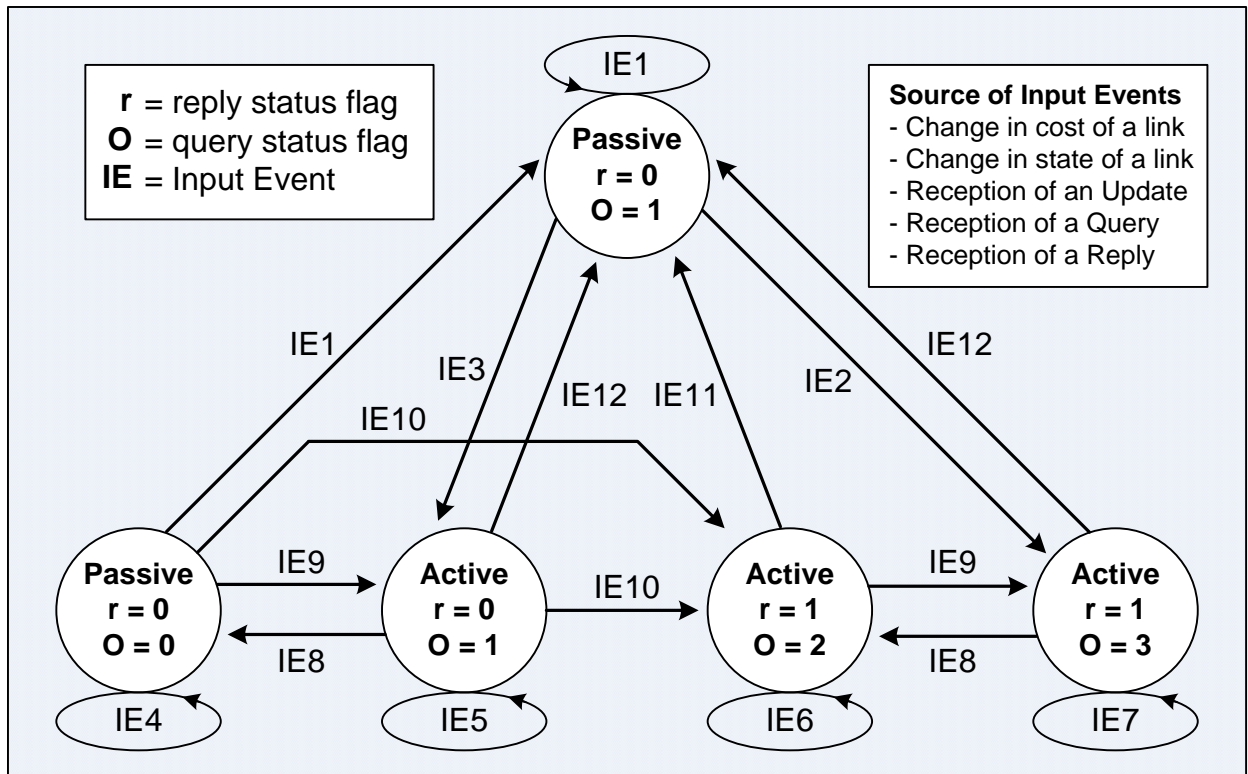


Figure A6-17: EIGRP DUAL Finite State Machine

- Below describes the input events for the DUAL finite state machine. A **finite state machine** (FSM) is a model of behavior composed of a finite number of states, transitions between the states, and actions. **Note:** FC is referred to as feasible condition, which states that a feasible successor is considered valid when the AD from a neighbor is less than the FD to the destination.

Input Event	Description
IE1	Any input event for which FC is satisfied or the destination is unreachable.
IE2	Query received from the successor; feasibility condition is not satisfied.
IE3	Input event other than a query from the successor; FC is not satisfied.
IE4	Input event other than last reply or a query from the successor.
IE5	Input event other than last reply, a query from the successor, or an increase in distance to destination.
IE6	Input event other than last reply.
IE7	Input event other than last reply or increase in distance to destination.
IE8	Increase of distance to destination.
IE9	Last reply received; FC is not satisfied with current feasible distance.
IE10	Query received from the successor.
IE11	Last reply received; FC is satisfied with current feasible distance.
IE12	Last reply received; set feasible distance to infinity.

## EIGRP Offset List

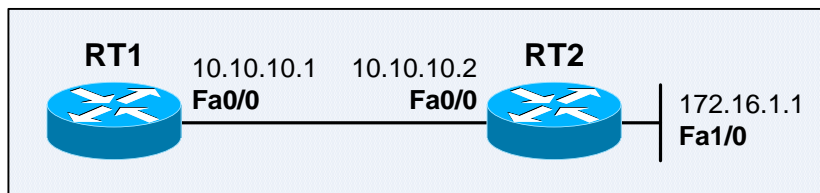


Figure A6-18: Network Setup for EIGRP Offset List

- A positive offset value can be added to the metrics of incoming or outgoing EIGRP routes.
- The **offset-list** *{acl-num | acl-name} {in | out} {offset} [intf-type intf-num]* EIGRP router subcommand configures an EIGRP offset list. Access list number 0 indicates all access lists. No action will be taken if the *offset* value is 0. An offset list with an interface type and number is considered extended and takes precedence over a non-extended offset list which is without an interface type and number. When a route entry passes the extended and normal offset lists, the offset of the extended offset list will be in effect and added to the metric for the route entry.
- Below shows the route entry learnt from RT2 via EIGRP before implementing any offset list:

```
RT1#sh ip route 172.16.1.0
Routing entry for 172.16.1.0/24
  Known via "eigrp 100", distance 90, metric 30720, type internal
  Redistributing via eigrp 100
  Last update from 10.10.10.2 on FastEthernet0/0, 00:13:36 ago
  Routing Descriptor Blocks:
  * 10.10.10.2, from 10.10.10.2, 00:13:36 ago, via FastEthernet0/0
    Route metric is 30720, traffic share count is 1
    Total delay is 200 microseconds, minimum bandwidth is 100000 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 1

RT1#
```

- The metric and delay has increased by 100 and 3 respectively after an offset list is implemented. It seems that EIGRP offset list alters the delay variable in order to change the EIGRP metric.

```
RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#router eigrp 100
RT1(config-router)#offset-list 0 in 100
RT1(config-router)#
00:14:29: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.2
(FastEthernet0/0) is down: route configuration changed
00:14:30: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.2
(FastEthernet0/0) is up: new adjacency
RT1(config-router)#do sh ip route 172.16.1.0
Routing entry for 172.16.1.0/24
  Known via "eigrp 100", distance 90, metric 30820, type internal
  Redistributing via eigrp 100
  Last update from 10.10.10.2 on FastEthernet0/0, 00:00:03 ago
  Routing Descriptor Blocks:
  * 10.10.10.2, from 10.10.10.2, 00:00:03 ago, via FastEthernet0/0
    Route metric is 30820, traffic share count is 1
    Total delay is 203 microseconds, minimum bandwidth is 100000 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 1
```



- Modifying the weights of EIGRP variables on RT1 and RT2 in order to remove the delay variable used for EIGRP metric computation and remove the offset list to restore the original state:

```

RT1(config-router)#do sh ip prot | in weight
EIGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
RT1(config-router)#
RT1(config-router)#metric weights 0 1 0 0 0
RT1(config-router)#no offset-list 0 in 100
RT1(config-router)#
RT1(config-router)#do sh ip prot | in weight
EIGRP metric weight K1=1, K2=0, K3=0, K4=0, K5=0
RT1(config-router)#
=====

RT2#conf t
Enter configuration commands, one per line. End with CNTL/Z.
RT2(config)#router eigrp 100
RT2(config-router)#metric weights 0 1 0 0 0
RT2(config-router)#
RT2(config-router)#do sh ip prot | in weight
EIGRP metric weight K1=1, K2=0, K3=0, K4=0, K5=0
RT2(config-router)#
=====

RT1(config-router)#do sh ip route 172.16.1.0
Routing entry for 172.16.1.0/24
  Known via "eigrp 100", distance 90, metric 25600, type internal
  Redistributing via eigrp 100
  Last update from 10.10.10.2 on FastEthernet0/0, 00:00:36 ago
  Routing Descriptor Blocks:
  * 10.10.10.2, from 10.10.10.2, 00:00:36 ago, via FastEthernet0/0
    Route metric is 25600, traffic share count is 1
    Total delay is 200 microseconds, minimum bandwidth is 100000 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 1

RT1(config-router)#

```

- The metric remains unchanged while the delay has increased by 3. This shows that setting the weight for the delay variable to 0 breaks the functionality and operation of EIGRP offset list.

```

RT1(config-router)#offset-list 0 in 100
RT1(config-router)#
00:17:19: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.2
(FastEthernet0/0) is down: route configuration changed
00:17:20: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.2
(FastEthernet0/0) is up: new adjacency
RT1(config-router)#do sh ip route 172.16.1.0
Routing entry for 172.16.1.0/24
  Known via "eigrp 100", distance 90, metric 25600, type internal
  Redistributing via eigrp 100
  Last update from 10.10.10.2 on FastEthernet0/0, 00:00:00 ago
  Routing Descriptor Blocks:
  * 10.10.10.2, from 10.10.10.2, 00:00:00 ago, via FastEthernet0/0
    Route metric is 25600, traffic share count is 1
    Total delay is 203 microseconds, minimum bandwidth is 100000 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 1

RT1(config-router)#

```

- Modifying the weight for the delay variable to 4 on RT1 and RT2 in order to observe how offset list handle a delay variable of more than 1 and remove the offset list to restore the original state:

```

RT1(config-router)#metric weights 0 1 0 4 0 0
RT1(config-router)#no offset-list 0 in 100
=====
RT2(config-router)#metric weights 0 1 0 4 0 0
=====
RT1(config-router)#do sh ip route 172.16.1.0
Routing entry for 172.16.1.0/24
  Known via "eigrp 100", distance 90, metric 46080, type internal
  Redistributing via eigrp 100
  Last update from 10.10.10.2 on FastEthernet0/0, 00:01:01 ago
  Routing Descriptor Blocks:
  * 10.10.10.2, from 10.10.10.2, 00:01:01 ago, via FastEthernet0/0
    Route metric is 46080, traffic share count is 1
    Total delay is 200 microseconds, minimum bandwidth is 100000 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 1

RT1(config-router)#

```

- The metric has increased by 400 instead of the 100 as specified in the offset list due to the weight for the delay variable is set to 4 times the default.

```

RT1(config-router)#offset-list 0 in 100
RT1(config-router)#
00:21:46: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.2 (FastE
thernet0/0) is down: route configuration changed
00:21:49: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.2 (FastE
thernet0/0) is up: new adjacency
RT1(config-router)#do sh ip route 172.16.1.0
Routing entry for 172.16.1.0/24
  Known via "eigrp 100", distance 90, metric 46480, type internal
  Redistributing via eigrp 100
  Last update from 10.10.10.2 on FastEthernet0/0, 00:00:01 ago
  Routing Descriptor Blocks:
  * 10.10.10.2, from 10.10.10.2, 00:00:01 ago, via FastEthernet0/0
    Route metric is 46480, traffic share count is 1
    Total delay is 203 microseconds, minimum bandwidth is 100000 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 1

RT1(config-router)#

```

- Modifying the weight for the delay variable to 1 and make it the only variable used for EIGRP metric calculation on RT1 and RT2 and remove the offset list to restore the original state:

```

RT1(config-router)#metric weights 0 0 0 1 0 0
RT1(config-router)#no offset-list 0 in 100
=====
RT2(config-router)#metric weights 0 0 0 1 0 0
=====
RT1(config-router)#do sh ip route 172.16.1.0
Routing entry for 172.16.1.0/24
  Known via "eigrp 100", distance 90, metric 5120, type internal
  Redistributing via eigrp 100
  Last update from 10.10.10.2 on FastEthernet0/0, 00:00:02 ago
  Routing Descriptor Blocks:
  * 10.10.10.2, from 10.10.10.2, 00:00:02 ago, via FastEthernet0/0
    Route metric is 5120, traffic share count is 1
    Total delay is 200 microseconds, minimum bandwidth is 100000 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 1

```

- The metric has increased by 100 as expected. Additionally, the weight for the bandwidth variable is not required for the functionality and operation of EIGRP offset list.

```

RT1(config-router)#offset-list 0 in 100
RT1(config-router)#
00:24:25: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.2 (FastE
thernet0/0) is down: route configuration changed
RT1(config-router)#offset-list 0 in 100
00:24:27: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.2 (FastE
thernet0/0) is up: new adjacency
RT1(config-router)#do sh ip route 172.16.1.0
Routing entry for 172.16.1.0/24
  Known via "eigrp 100", distance 90, metric 5220, type internal
  Redistributing via eigrp 100
  Last update from 10.10.10.2 on FastEthernet0/0, 00:00:01 ago
  Routing Descriptor Blocks:
  * 10.10.10.2, from 10.10.10.2, 00:00:01 ago, via FastEthernet0/0
    Route metric is 5220, traffic share count is 1
    Total delay is 203 microseconds, minimum bandwidth is 100000 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 1

RT1(config-router)#

```

- Lastly, modify the weight for the delay variable to 4 and make it the only variable used for EIGRP metric calculation on RT1 and RT2 and remove the offset list to restore the original state:

```

RT1(config-router)#metric weights 0 0 0 4 0 0
RT1(config-router)#no offset-list 0 in 100
=====
RT2(config-router)#metric weights 0 0 0 4 0 0
=====
RT1(config-router)#do sh ip route 172.16.1.0
Routing entry for 172.16.1.0/24
  Known via "eigrp 100", distance 90, metric 20480, type internal
  Redistributing via eigrp 100
  Last update from 10.10.10.2 on FastEthernet0/0, 00:00:08 ago
  Routing Descriptor Blocks:
  * 10.10.10.2, from 10.10.10.2, 00:00:08 ago, via FastEthernet0/0
    Route metric is 20480, traffic share count is 1
    Total delay is 200 microseconds, minimum bandwidth is 100000 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 1

RT1(config-router)#

```

- The metric has increased by 400 instead of the 100 as specified in the offset list due to the weight for the delay variable is set to 4 times the default.

```

RT1(config-router)#offset-list 0 in 100
00:27:13: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.2 (FastE
thernet0/0) is down: route configuration changed
00:27:15: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.2 (FastE
thernet0/0) is up: new adjacency
RT1(config-router)#do sh ip route 172.16.1.0
Routing entry for 172.16.1.0/24
  Known via "eigrp 100", distance 90, metric 20880, type internal
  Redistributing via eigrp 100
  Last update from 10.10.10.2 on FastEthernet0/0, 00:00:05 ago
  Routing Descriptor Blocks:
  * 10.10.10.2, from 10.10.10.2, 00:00:05 ago, via FastEthernet0/0
    Route metric is 20880, traffic share count is 1
    Total delay is 203 microseconds, minimum bandwidth is 100000 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 1

RT1(config-router)#

```

- Throughout the various combinations of EIGRP offset list configurations above, it is proven that the EIGRP offset list only alters the delay variable and is only accurate when the weight for the delay variable is set to 1 (default).

## EIGRP Debug Commands

- Below shows the output of the **debug eigrp packets** privileged command captured on a router:

```
RT1#debug eigrp packets
EIGRP Packets debugging is on
  (UPDATE, REQUEST, QUERY, REPLY, HELLO, IPXSAP, PROBE, ACK, STUB,
  SIAQUERY, SIAREPLY)
RT1#
00:01:00: EIGRP: Sending HELLO on FastEthernet0/0
00:01:00:   AS 100, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
00:01:05: EIGRP: Sending HELLO on FastEthernet0/0
00:01:05:   AS 100, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
00:01:07: EIGRP: Received HELLO on FastEthernet0/0 nbr 10.10.10.2
00:01:07:   AS 100, Flags 0x0, Seq 0/0 idbQ 0/0
00:01:07: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.2
(FastEthernet0/0) is up: new adjacency
00:01:07: EIGRP: Enqueueing UPDATE on FastEthernet0/0 nbr 10.10.10.2 iidbQ
un/rely 0/1 peerQ un/rely 0/0 serno 1-1
00:01:07: EIGRP: Requeued unicast on FastEthernet0/0
00:01:07: EIGRP: Sending UPDATE on FastEthernet0/0 nbr 10.10.10.2
00:01:07:   AS 100, Flags 0x9, Seq 1/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ
un/rely 0/1 serno 1-1
00:01:09: EIGRP: Sending UPDATE on FastEthernet0/0 nbr 10.10.10.2, retry 1,
RTO 3000
00:01:09:   AS 100, Flags 0x9, Seq 1/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ
un/rely 0/1 serno 1-1
00:01:09: EIGRP: Sending HELLO on FastEthernet0/0
00:01:09:   AS 100, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
00:01:10: EIGRP: Received UPDATE on FastEthernet0/0 nbr 10.10.10.2
00:01:10:   AS 100, Flags 0x9, Seq 1/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ
un/rely 0/1
00:01:10: EIGRP: Enqueueing UPDATE on FastEthernet0/0 iidbQ un/rely 0/1
serno 2-2
00:01:10: EIGRP: Building Sequence TLV
00:01:10: EIGRP: Enqueueing UPDATE on FastEthernet0/0 nbr 10.10.10.2 iidbQ
un/rely 0/0 peerQ un/rely 0/1 serno 2-2
00:01:10: EIGRP: Sending HELLO with Sequence TLV on FastEthernet0/0, seq 2
00:01:10: EIGRP: Sending HELLO on FastEthernet0/0
00:01:10:   AS 100, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
00:01:10: EIGRP: Sending UPDATE on FastEthernet0/0
00:01:10:   AS 100, Flags 0x2, Seq 2/0 idbQ 0/0 iidbQ un/rely 0/0 serno 2-2
00:01:11: EIGRP: Received HELLO on FastEthernet0/0 nbr 10.10.10.2
00:01:11:   AS 100, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ
un/rely 0/2
00:01:12: EIGRP: Received UPDATE on FastEthernet0/0 nbr 10.10.10.2
00:01:12:   AS 100, Flags 0x9, Seq 1/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ
un/rely 0/2, last received seq 1, out of sequence, this seq 1
00:01:12: EIGRP: Sending UPDATE on FastEthernet0/0 nbr 10.10.10.2, retry 2,
RTO 4500
00:01:12:   AS 100, Flags 0x9, Seq 1/1 idbQ 0/0 iidbQ un/rely 0/0 peerQ
un/rely 0/2 serno 1-1
00:01:12: EIGRP: Received ACK on FastEthernet0/0 nbr 10.10.10.2
00:01:12:   AS 100, Flags 0x0, Seq 0/1 idbQ 0/0 iidbQ un/rely 0/0 peerQ
un/rely 0/2
00:01:12: EIGRP: Sending UPDATE on FastEthernet0/0 nbr 10.10.10.2
00:01:12:   AS 100, Flags 0x0, Seq 2/1 idbQ 0/0 iidbQ un/rely 0/0 peerQ
un/rely 0/1 serno 2-2
00:01:13: EIGRP: Received ACK on FastEthernet0/0 nbr 10.10.10.2
00:01:13:   AS 100, Flags 0x0, Seq 0/2 idbQ 0/0 iidbQ un/rely 0/0 peerQ
un/rely 0/1
00:01:13: EIGRP: FastEthernet0/0 multicast flow blocking cleared
```

```

00:01:14: EIGRP: Sending HELLO on FastEthernet0/0
00:01:14:   AS 100, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
00:01:16: EIGRP: Received HELLO on FastEthernet0/0 nbr 10.10.10.2
00:01:16:   AS 100, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ
un/rely 0/0
00:01:18: EIGRP: Sending HELLO on FastEthernet0/0
00:01:18:   AS 100, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
00:01:21: EIGRP: Received HELLO on FastEthernet0/0 nbr 10.10.10.2
00:01:21:   AS 100, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ
un/rely 0/0

```

RT1#

**RT1#sh ip route**

Gateway of last resort is not set

```

          172.16.0.0/24 is subnetted, 1 subnets
D       172.16.1.0 [90/30720] via 10.10.10.2, 00:00:28, FastEthernet0/0
          10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0

```

RT1#

**RT1#sh ip eigrp topology all-links**

IP-EIGRP Topology Table for AS(100)/ID(10.10.10.1)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,  
r - reply Status, s - sia Status

```

P 10.10.10.0/24, 1 successors, FD is 28160, serno 1
   via Connected, FastEthernet0/0
P 172.16.1.0/24, 1 successors, FD is 30720, serno 2
   via 10.10.10.2 (30720/28160), FastEthernet0/0

```

RT1#

**RT1#sh ip eigrp neighbors**

IP-EIGRP neighbors for process 100

H	Address	Interface	Hold (sec)	Uptime	SRTT (ms)	RTO	Q Cnt	Seq Num
0	10.10.10.2	Fa0/0	597	00:00:20	396	2376	0	<b>1</b>

RT1#

**RT2#sh ip eigrp neighbors**

IP-EIGRP neighbors for process 100

H	Address	Interface	Hold (sec)	Uptime	SRTT (ms)	RTO	Q Cnt	Seq Num
0	10.10.10.1	Fa0/0	597	00:00:25	1680	5000	0	<b>2</b>

RT2#

**RT2#sh ip eigrp topology all-links**

IP-EIGRP Topology Table for AS(100)/ID(10.10.10.2)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,  
r - reply Status, s - sia Status

```

P 10.10.10.0/24, 1 successors, FD is 28160, serno 1
   via Connected, FastEthernet0/0
P 172.16.1.0/24, 1 successors, FD is 28160, serno 2
   via Connected, FastEthernet1/0

```

RT2#

- The **debug eigrp packets** privileged command traces the transmission and reception of EIGRP packets. Hello packets are sent unreliably, hence the sequence number (seq) does not increment.
- When the router sends the final update packet after the EIGRP convergence to its neighboring router, values appear in the sequence number field – Seq 2/1 indicates that the router sends an update packet with a sequence number of 2 and acknowledges the receipt of a packet with a sequence number of 1 from its neighboring router. The Seq Num field in the outputs of the **show ip eigrp neighbors EXEC** command on both routers reflect this information.
- Eventually, RT1 and RT2 are expecting to receive the next reliable packet from the corresponding neighbor with a sequence number of 2 and 3 respectively.
- The serial number (serno 2-2) indicates the number of changes that the 2 neighboring routers register in their EIGRP topology tables.
- The sequence number increments each time an update, query, or reply packet is sent; while the serial number increments each time the topology table changes. Therefore, if the topology table has more than 100 changes, the serial number increases substantially, but the sequence number may only increase by 1.
- The **debug eigrp packets** privileged command must be used with caution and only use it when diving deeper into a problem. Never start troubleshooting EIGRP with this command.
- Below shows the output of the **debug ip eigrp** privileged command captured on a router:

```

RT1#debug ip eigrp
IP-EIGRP Route Events debugging is on
RT1#
00:00:44: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.2
(FastEthernet0/0) is up: new adjacency
00:00:44: IP-EIGRP(Default-IP-Routing-Table:100): 10.10.10.0/24 - do
advertise out FastEthernet0/0
00:00:46: IP-EIGRP(Default-IP-Routing-Table:100): Processing incoming UPDATE
packet
00:00:46: IP-EIGRP(Default-IP-Routing-Table:100): Int 172.16.1.0/24 M 30720
- 25600 5120 SM 28160 - 25600 2560
00:00:46: IP-EIGRP(Default-IP-Routing-Table:100): Int 172.16.1.0/24 metric
30720 - 25600 5120
00:00:46: IP-EIGRP(Default-IP-Routing-Table:100): 10.10.10.0/24 - do
advertise out FastEthernet0/0
00:00:47: IP-EIGRP(Default-IP-Routing-Table:100): Int 172.16.1.0/24 metric
30720 - 25600 5120
RT1#
RT1#sh ip eigrp topology
IP-EIGRP Topology Table for AS(100)/ID(10.10.10.1)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.10.10.0/24, 1 successors, FD is 28160
   via Connected, FastEthernet0/0
P 172.16.1.0/24, 1 successors, FD is 30720
   via 10.10.10.2 (30720/28160), FastEthernet0/0
RT1#

```

- The **debug ip eigrp [neighbor as-num neighbor-id]** privileged command verifies the operation of EIGRP.
- The EIGRP metric is equal to the bandwidth plus the delay by default. The EIGRP process uses the source metric (SM) information in the update packet to calculate the advertised distance (AD) from a neighboring router and place it into the EIGRP topology table. In this example, the SM for 172.16.1.0/24 via 10.10.10.2 is SM 28160 - 25600 2560, which means the source metric (AD) is 25600 (bandwidth) + 2560 (delay) = 28160.
- The EIGRP metric calculation for the total delay uses the metric (M) information in the update. In this example, the M information is M 30720 - 25600 5120, which means the metric (FD) is 25600 (bandwidth) + 5120 (delay) = 30720.

```

RT1#sh ip eigrp topology 172.16.1.0/24
IP-EIGRP (AS 100): Topology entry for 172.16.1.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 30720
  Routing Descriptor Blocks:
    10.10.10.2 (FastEthernet0/0), from 10.10.10.2, Send flag is 0x0
      Composite metric is (30720/28160), Route is Internal
      Vector metric:
        Minimum bandwidth is 100000 Kbit
        Total delay is 200 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
RT1#

```

$$\frac{10000000}{100000} \times 256 = 25600 \text{ (bandwidth)}, 20 \times 256 = 5120 \text{ (delay)}.$$



## Duplicate EIGRP Router IDs Preventing Installation of EIGRP External Route

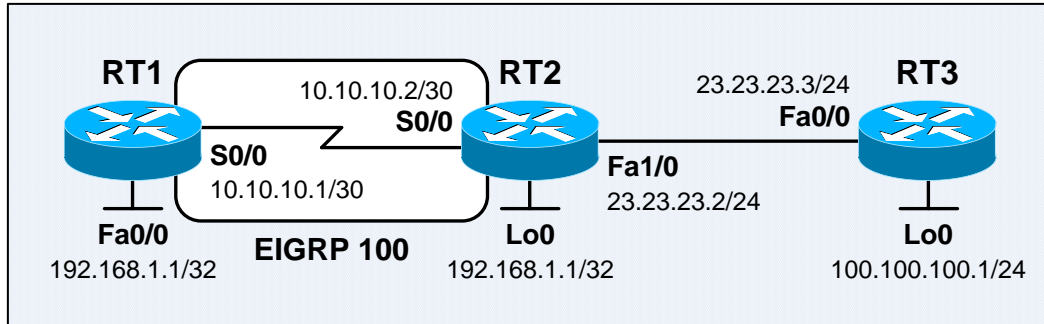


Figure A6-19: Network Setup for Duplicate EIGRP Router IDs

- EIGRP automatically selects an IP address as the Router ID when an EIGRP process is started. EIGRP does not use Router ID as extensively as OSPF. The only time EIGRP uses the Router ID is when redistributing external routes into an EIGRP routing domain. The Originating Router field of an EIGRP IP External Route packet indicates the Router ID of the redistributing router.
- The EIGRP Router ID is selected in the same manner as OSPF Router ID – the highest local IP address is selected and loopback interfaces are preferred. The Router ID is not changed unless the EIGRP process is totally restarted with the **no router eigrp** command or when the Router ID is manually configured and changed with the **eigrp router-id {router-id}** router subcommand.
- Below shows the routing tables on RT1 and RT2 before redistributing the static route to 100.100.100.0/24 into EIGRP on RT2:

```
RT1#sh ip route
```

```
Gateway of last resort is not set
```

```

    10.0.0.0/30 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, Serial0/0
C       192.168.1.0/24 is directly connected, FastEthernet1/0
RT1#
```

```
RT1#clear ip eigrp events
```

```
RT1#sh ip eigrp events
```

```
Event information for AS 100:
  Event log is empty.
RT1#
```

```
-----
RT2#sh ip route
```

```
Gateway of last resort is not set
```

```

    100.0.0.0/24 is subnetted, 1 subnets
S       100.100.100.0 [1/0] via 23.23.23.3
    23.0.0.0/24 is subnetted, 1 subnets
C       23.23.23.0 is directly connected, FastEthernet1/0
    10.0.0.0/30 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, Serial0/0
    192.168.1.0/32 is subnetted, 1 subnets
C       192.168.1.1 is directly connected, Loopback0
RT2#
```

- Below shows configuration steps on RT2 to redistribute the static route into EIGRP:

```

RT2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT2(config)#router eigrp 100
RT2(config-router)#redistribute static ?
  metric      Metric for redistributed routes
  route-map   Route map reference
  <cr>

RT2(config-router)#redistribute static
RT2(config-router)#^Z
RT2#
RT2#sh ip eigrp topology
IP-EIGRP Topology Table for AS(100)/ID(192.168.1.1)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.10.10.0/30, 1 successors, FD is 2169856
   via Connected, Serial0/0
P 100.100.100.0/24, 1 successors, FD is 28160
   via Rstatic (28160/0)

RT2#
RT2#sh ip eigrp topology 100.100.100.0
% IP-EIGRP (AS 100): Route not in topology table
RT2#sh ip eigrp topology 100.100.100.0 255.255.255.0
IP-EIGRP (AS 100): Topology entry for 100.100.100.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 28160
  Routing Descriptor Blocks:
  23.23.23.3, from Rstatic, Send flag is 0x0
    Composite metric is (28160/0), Route is External
  Vector metric:
    Minimum bandwidth is 100000 Kbit
    Total delay is 100 microseconds
    Reliability is 255/255
    Load is 1/255
    Minimum MTU is 1500
    Hop count is 0
  External data:
    Originating router is 192.168.1.1 (this system)
    AS number of route is 0
    External protocol is Static, external metric is 0
    Administrator tag is 0 (0x00000000)

RT2#

```

- Below shows that RT1 discards the external route originated from RT2 as it found out that the Originating Router field of the external route is same as its Router ID. RT1 does not insert the route into its EIGRP topology table (and routing table) to prevent a routing loop – it thinks that it is the originator of the route; by receiving the route back from other neighbors, it must be a loop.

```

RT1#sh ip route

Gateway of last resort is not set

    10.0.0.0/30 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, Serial0/0
C       192.168.1.0/24 is directly connected, FastEthernet1/0
RT1#

```

```

RT1#sh ip eigrp events
Event information for AS 100:
1    00:02:29.967 Ignored route, metric: 100.100.100.0 2172416
2    00:02:29.967 Ignored route, neighbor info: 10.10.10.2 Serial0/1
3    00:02:29.967 Ignored route, dup router: 192.168.1.1
RT1#
RT1#sh ip eigrp topology
IP-EIGRP Topology Table for AS(100)/ID(192.168.1.1)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.10.10.0/30, 1 successors, FD is 2169856
   via Connected, Serial0/0
RT1#

```

- Below shows a possible solution of the problem – modify the EIGRP Router ID of RT1. Changing the subnet between RT1 and RT2 from 10.10.10.0/30 to 200.200.200.0/30 is another feasible solution the Router ID of RT1 will be changed from 192.168.1.1 to 200.200.200.1.

```

RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#router eigrp 100
RT1(config-router)#eigrp router-id 1.1.1.1
RT1(config-router)#^Z
RT1#
00:04:25: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.2
(Serial0/0) is down: route configuration changed
00:04:28: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.2
(Serial0/0) is up: new adjacency
RT1#sh ip route

Gateway of last resort is not set

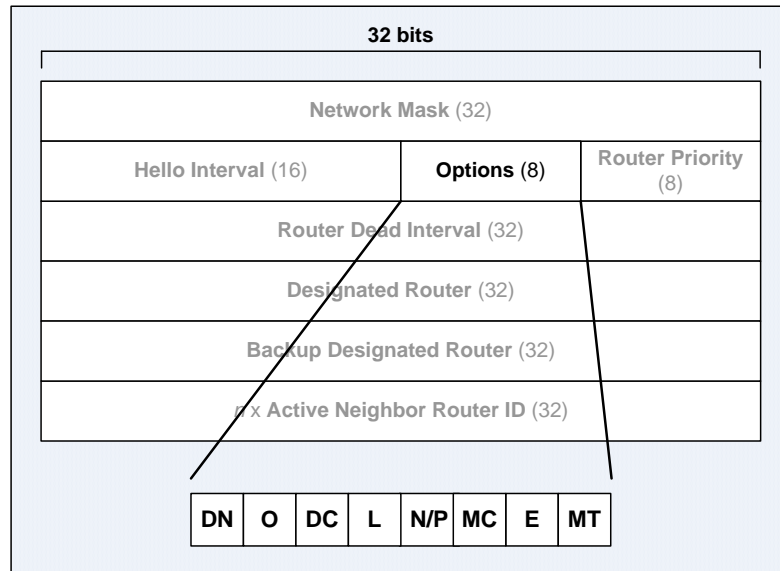
    100.0.0.0/24 is subnetted, 1 subnets
D EX    100.100.100.0 [170/2172416] via 10.10.10.2, 00:00:08, Serial0/0
    10.0.0.0/30 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, Serial0/0
C       192.168.1.0/24 is directly connected, FastEthernet1/0
RT1#
RT1#sh ip eigrp topology
IP-EIGRP Topology Table for AS(100)/ID(1.1.1.1)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.10.10.0/30, 1 successors, FD is 2169856
   via Connected, Serial0/0
P 100.100.100.0/24, 1 successors, FD is 2172416
   via 10.10.10.2 (2172416/28160), Serial0/0
RT1#
-----
RT2#
00:04:22: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.1
(Serial0/0) is down: Interface Goodbye received
00:04:27: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 10.10.10.1
(Serial0/0) is up: new adjacency
RT2#

```

## The OSPF Options Field



**Figure A6-20: OSPF Options Field**

- The 8-bit Options field is present in every OSPF Hello and DBD packets, as well as every LSA. It allows OSPF routers to communicate their optional capabilities. OSPF routers may not form adjacency due to compatibility issue of capabilities.
- Below describes the bits in the Options field:

<b>DN</b>	Used in MPLS-based Layer 3 Virtual Private Networks as defined in RFC 2547 – BGP/MPLS VPNs. When a route learnt from a customer network via OSPF is advertised across a BGP/MPLS VPN using Multiprotocol BGP, and advertised back to a customer network via OSPF, a loop can occur in which the OSPF route is redistributed back to the VPN service provider network via BGP. The DN bit prevents this type of routing loop. When an OSPF router received a Type 3, 5, or 7 LSA with the DN bit set, it cannot use that LSA in the OSPF route calculations.
<b>O</b>	The O bit is set when the originating router supports Type 9, 10, and 11 Opaque LSAs.
<b>DC</b>	The DC bit is set when the originating router supports OSPF over Demand Circuits.
<b>L</b>	Indicates whether the OSPF packet contains a <b>Link-Local Signaling (LLS)</b> data block. This bit is set only in Hello and DBD packets. If the OSPF packet is cryptographically authenticated, the LLS data block must also be cryptographically authenticated. <b>Reference:</b> RFC 4813 – OSPF Link-Local Signaling.
<b>N</b>	The N bit is used only in Hello packets. The N bit is set when the originating router supports Type-7 NSSA-External-LSAs. Neighboring routers with mismatched N bit value will not form neighbor relationship. This restriction ensures that all OSPF routers within an area support NSSA capabilities. When the N bit is set to 1, the E bit must be 0.
<b>P</b>	The P bit is used only in Type-7 NSSA-External-LSA headers. Due to this reason, the N and P bits can share the same position in the Options field. The P (Propagate) bit is set to inform the NSSA ABR to translate Type-7 LSAs into Type-5 LSAs.
<b>MC</b>	The MC bit is set when the originating router supports Multicast extensions to OSPF (MOSPF). <b>Note:</b> Cisco does not support MOSPF, mainly due to the reasons that it uses a dense-mode multicast forwarding scheme and is protocol dependent. A Cisco router would generate a %OSPF-4-BADLSATYPE error message upon receiving a Type-6 LSA. The <b>ignore lsa mospf</b> router subcommand configures an OSPF router to ignore Type-6 LSAs and therefore prevents the router from generating the error message.

<b>E</b>	The E (ExternalRoutingCapability) bit is set when the originating router is capable of accepting AS External LSAs. It will be set to 1 in all AS External LSAs and in all LSAs originated in the backbone and non-stub areas; and will be set to 0 in all Hellos and LSAs originated within a stub area. Additionally, this bit is used in to Hello packets to indicate the capability of a router interface to send and receive Type-5 AS-External-LSAs. Neighboring routers with mismatched E bit value will not form neighbor relationship. This restriction ensures that all OSPF routers within an area support the stub capabilities.
<b>MT</b>	The MT bit is set when the originating router supports Multitopology OSPF, MT-OSPF. However, MT-OSPF is still under the proposal stage and is not generally adopted yet. Older OSPF specifications specified this bit position as the T bit. The T bit was set when the originating router support TOS-based routing. However, OSPF TOS-based routing has never been deployed; therefore the T bit was also never been used.

## OSPF Link-Local Signaling

- The OSPF Link-Local Signaling feature was introduced by Cisco and is defined in RFC 4813 – OSPF Link-Local Signaling to allow routers to exchange arbitrary data – router capabilities, which may be necessary in certain situations. It is a backward-compatible technique which uses existing standard OSPF packet types without introducing a new OSPF packet type.
- LLS TLV types are maintained by the IANA. OSPF extensions that require a new LLS TVL type must be reviewed by a designated expert from the IETF routing protocol domain. Below lists the currently assigned LLS TLV types:

LLS TLV Type	Name
0	Reserved
1	Extended Options (EO)
2	Cryptographic Authentication
3 – 32767	Reserved for IANA assignment
32768 – 65535	Private Use

- The following bits are assigned for the Extended Options bits field in the Extended Options TLV:

Extended Option Bit	Name	Reference
0x00000001	LSDB Resynchronization (LR-bit)	RFC 4811 – OSPF Out-of-Band Link State Database (LSDB) Resynchronization
0x00000002	Restart Signal (RS-bit)	RFC 4812 – OSPF Restart Signaling

- A usage of Link-Local Signaling is to exchange the capabilities information between Cisco routers to enable OSPF **Non-Stop Forwarding** (NSF) awareness on the OSPF-enabled interfaces. Cisco Catalyst 6500 **Virtual Switching System** (VSS) utilizes NSF to provide high availability.
- The OSPF Per-Interface Link-Local Signaling feature allows us to selectively enable or disable Link-Local Signaling (LLS) for a specific interface regardless of the global (router level) setting. The **ip ospf lls [disable]** interface subcommand takes precedence over the **[no] capability lls** router subcommand. Ex: A router with the **no capability lls** router subcommand configured disables the LLS support at the router level. However, the **ip ospf lls** interface subcommand can be configured to selectively enable LLS support for a particular interface.
- Issue the **no ip ospf lls** interface subcommand to restore the LLS setting of an interface and let the router-level LLS setting (**capability lls**) to decide the LLS setting for the interface.

# The OSPF Neighbor State Machine

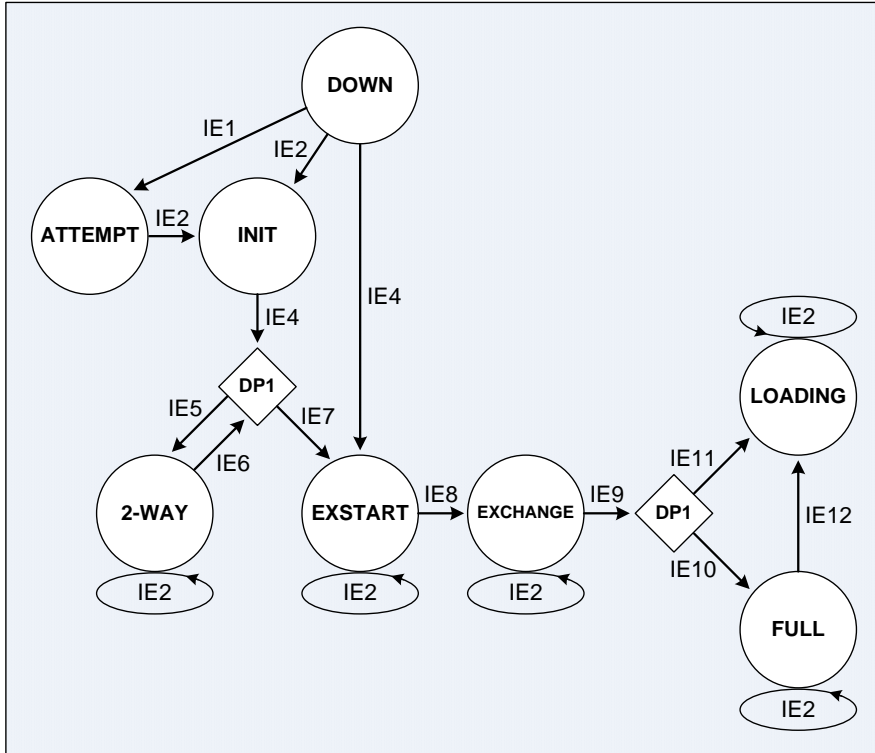


Figure A6-21: OSPF Neighbor State Machine - Normal Transitions

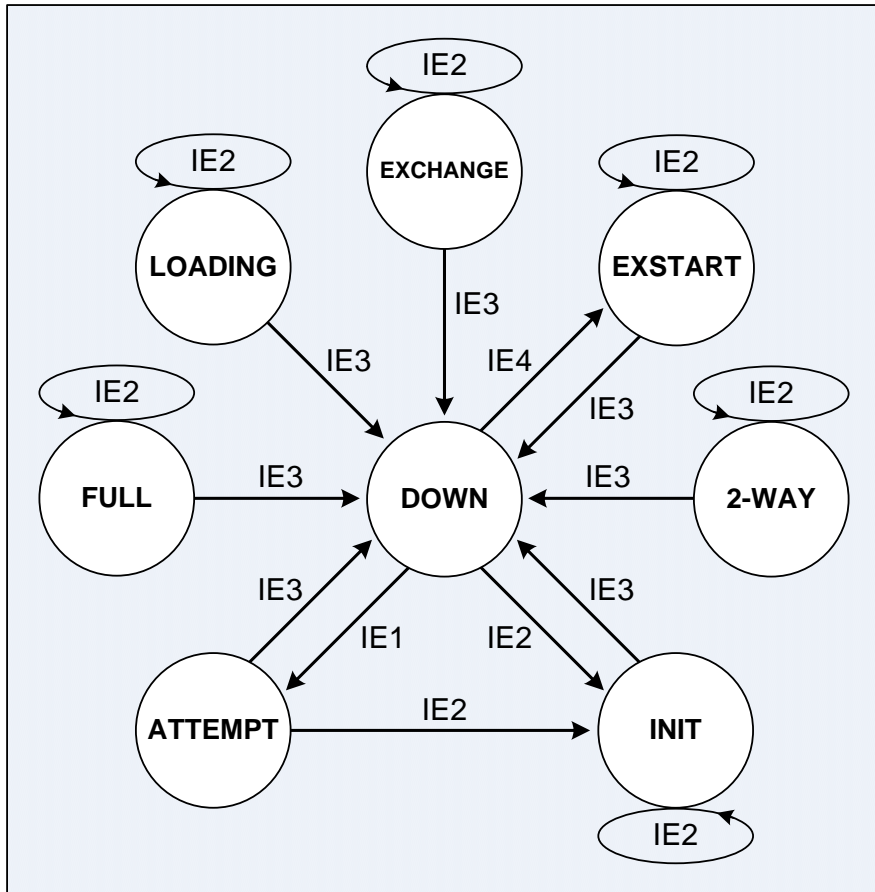
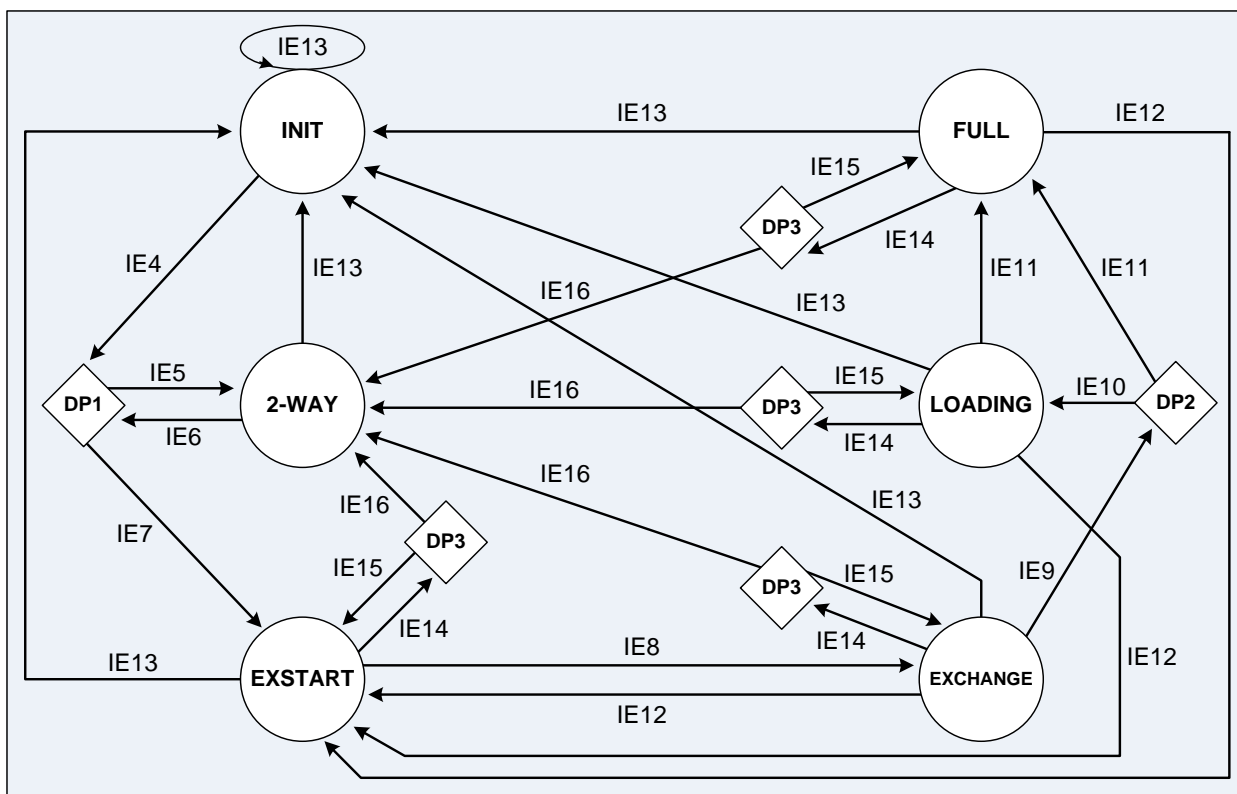


Figure A6-22: OSPF Neighbor State Machine - From DOWN to INIT



**Figure A6-23: OSPF Neighbor State Machine - From INIT to FULL**

- Below describes the input events (IEs) and decision points (DPs) for the OSPF Neighbor State Machine diagrams above:

Input Event	Description
IE1	Applies only for neighbors on NBMA environments. This input event will be triggered under either of the following conditions: <ol style="list-style-type: none"> <li>1) The interface to the NBMA network first becomes active, and the neighbor is eligible for DR election.</li> <li>2) The router becomes either DR or BDR, and the neighbor is not eligible for DR election.</li> </ol>
IE2	A valid Hello packet has been received from the neighbor.
IE3	The neighbor is no longer reachable, as determined by the lower level protocols, by an explicit instruction from the OSPF process itself, or by the expiration of the dead timer.
IE4	The router first sees its own Router ID listed in the neighbor list of the neighbor's Hello packet or receives a DBD packet from the neighbor.
IE5	The neighbor should not become adjacent.
IE6	This input event occurs under either of the following conditions: <ol style="list-style-type: none"> <li>1) The neighbor state first transitions to 2-WAY.</li> <li>2) The interface state changes.</li> </ol>
IE7	An adjacency should be formed with this neighbor.
IE8	The Master-Slave relationship has been established and the initial DD sequence number has been decided.
IE9	The exchange of DBD packets has been completed.
IE10	There are entries in the Link State Request list.
IE11	The Link State Request list is empty.

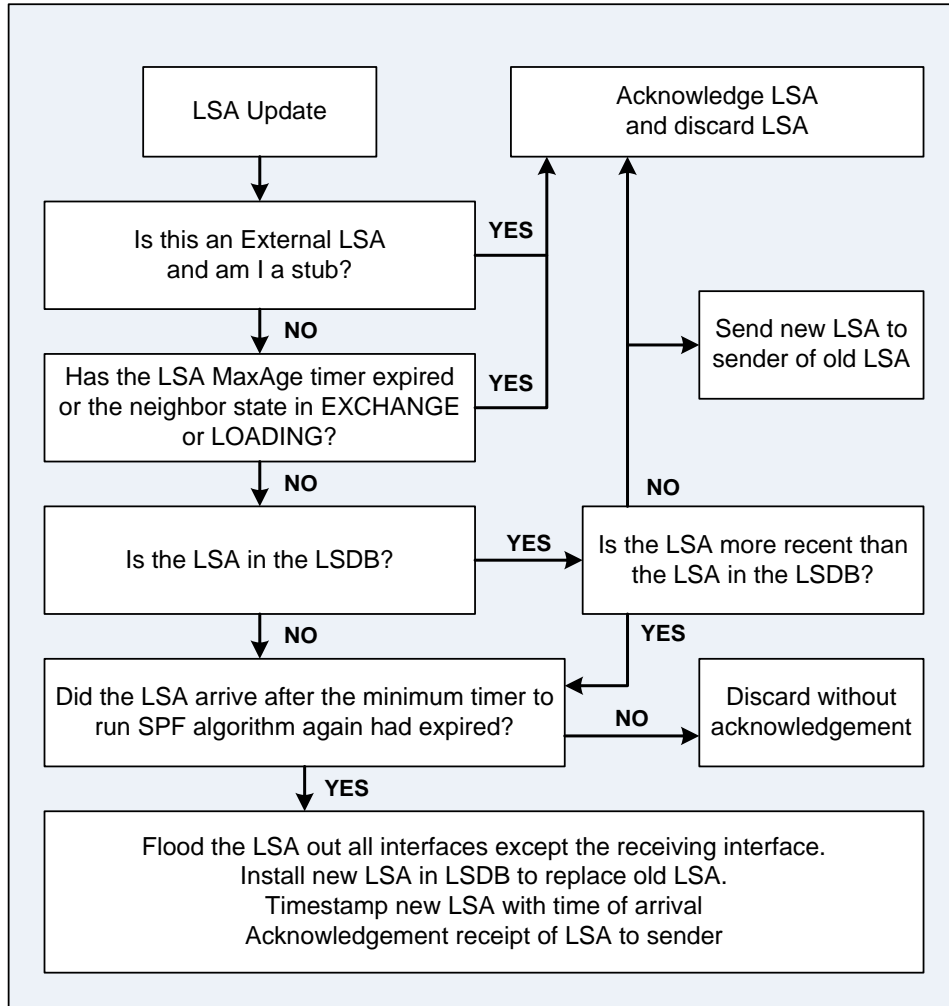
IE12	The adjacency should be broken and then restarted. This input event will be triggered by any of the following conditions: 1) The reception of a DBD packet with an unexpected DD sequence number. 2) The reception of a DBD packet with the Options field set differently than the Options field of the last DBD packet. 3) The reception of a DBD packet, other than the first packet, in which the Init bit is set. 4) The reception of a Link State Request packet for an LSA that is not in the database.
IE13	A Hello packet has been received from the neighbor in which the Router ID of the receiving router is not in the neighbor list of the Hello packet.
IE14	Occurs when the interface state changes.
IE15	The existing or forming adjacency with the neighbor should continue.
IE16	The existing or forming adjacency with the neighbor should not continue.
DP1	Should an adjacency be established with the neighbor? An adjacency should be formed if one or more following conditions is true: 1) The network type is point-to-point. 2) The network type is point-to-multipoint. 3) The network type is virtual link. 4) The router is the DR for the network on which the neighbor is located. 5) The router is the BDR for the network on which the neighbor is located. 6) The neighbor is the DR. 7) The neighbor is the BDR.
DP2	Is the Link State Request list of the neighbor is empty?
DP3	Should the existing or forming adjacency with the neighbor continue?

## OSPF TOS-based Routing

- The **Type of Service** (TOS) field in the IP header can affect the path of a packet. 5 TOS values have been defined: normal service, minimize monetary cost, maximize reliability, maximize throughput, and minimize delay. The TOS of a packet allow a router to choose an appropriate path for the packet. To implement TOS routing, a router would keep separate routing tables for each TOS value. When forwarding a packet, the router would first choose a routing table based on the TOS of the packet, and then perform the normal routing table lookup.
- The TOS bits in the IP header should not be confused with the IP Precedence bits, which are collocated in the same byte of the IP header with the TOS bits. The IP precedence label of a packet does not affect the path of the packet but instead specifies transmission priority at each router hop. The IP Precedence bits remain in IPv6, recast as the Priority bits.
- TOS routing has rarely been used in the Internet. Only 2 Internet routing protocols have ever supported the calculation of separate paths for each TOS value – OSPF and IS-IS. Due to very little deployment of TOS routing, TOS has recently been removed from the OSPF specification. For the same reason, TOS has also been omitted from IPv6.



## Updating the OSPF Link-State Database



**Figure A6-24:** Updating the OSPF Link-State Database

## The show ip ospf interface EXEC Command



Figure A6-25: Network Setup for the **show ip ospf interface** EXEC Command

- Below shows the output of the **show ip ospf interface** [*intf-type intf-num*] command on RT1. The output of the command varies upon the different type of interfaces.

```
RT1#sh ip ospf interface fa0/0
FastEthernet0/0 is up, line protocol is up
  Internet Address 192.168.1.1/24, Area 0
  Process ID 100, Router ID 10.10.10.1, Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State BDR, Priority 1
  Designated Router (ID) 10.10.10.2, Interface address 192.168.1.2
  Backup Designated router (ID) 10.10.10.1, Interface address 192.168.1.1
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
  oob-resync timeout 40
  Hello due in 00:00:05
  Index 1/1, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 10.10.10.2 (Designated Router)
  Suppress hello for 0 neighbor(s)
RT1#
```

- Below describes the various information that is being revealed in the mentioned command:

Field	Description
<b>Interface State</b>	The 1st line of the output displays the Layer 1 and Layer 2 states of an interface. The interface should be in the up / up state.
<b>IP Address and Area</b>	Indicates the IP address configured on an interface and the OSPF area in which the interface resides in.
<b>Process ID</b>	Indicates the OSPF process to which the interface belongs. The OSPF Process ID is locally significant therefore 2 neighboring OSPF routers with different Process IDs can establish adjacency. An OSPF router can run multiple OSPF processes, and the Process IDs are being used to differentiate the OSPF processes.
<b>Router ID</b>	The OSPF Router ID is selected upon the start of an OSPF process. Once the Router ID is elected, it does not change unless the OSPF process restarts upon issuing the <b>clear ip ospf process</b> command.
<b>Network Type</b>	Indicates the network type (eg: BROADCAST, NON_BROADCAST, POINT_TO_MULTIPOINT, POINT_TO_POINT) of the interface. Routers with different network type interfaces are able to become neighbor and established FULL adjacency; however, inconsistency link-state information affects the operation of OSPF and prevents routes from being installed into the routing table.
<b>Cost</b>	Indicates the OSPF metric associated with the interface. The <b>ip ospf cost {cost}</b> interface subcommand specifies the cost of an interface.

<b>Transmit Delay</b>	Indicates the transmission and propagation delay of an interface. The LS Age of an LSA is incremented by this timer value in order to accommodate transmission and propagation delay of an interface. The transmit delay timer is important on low speed links where the transmission delay is significant. The default value is 1 second.
<b>State</b>	Defines the state of the link to either one of DR, BDR, DROTHER, WAITING, POINT_TO_POINT, and POINT_TO_MULTIPOINT.
<b>Priority</b>	Indicates the OSPF priority on the interface that is being used when electing the DR and BDR on a broadcast network media. The router with the highest priority becomes the DR. If the priorities are same, the router with the highest Router ID becomes the DR. The default value is 1. A router with a priority of 0 never participates in the DR/BDR election process and is known as a DROTHER.
<b>Designated Router</b>	Indicates the Router ID of the DR on a broadcast network.
<b>DR Interface Address</b>	Indicates the physical IP address of the DR that connects to the broadcast network.
<b>Backup DR</b>	Indicates the Router ID of the BDR on a broadcast network.
<b>BDR Interface Address</b>	Indicates the physical IP address of the BDR that connects to the broadcast network.
<b>Timer Intervals</b>	<p><b>Hello</b> – Interval time in seconds for sending OSPF Hello packets.</p> <p><b>Dead</b> – Time in seconds to wait before declaring a neighbor is dead.</p> <p><b>Wait</b> – Time to wait (wait period) before electing the DR.</p> <p><b>Retransmit</b> – Time to wait before retransmit a LSA when it has not been acknowledged. This ensures reliable flooding.</p> <p><b>OOB-Resync Timeout</b> – Utilized by RFC4811 OSPF Out-of-Band LSDB Resynchronization to limit the amount of time allowed for 2 neighboring routers to resynchronize their LSDBs upon an OOB-Resync event.</p> <p><b>Hello Due In</b> – A Hello packet will be sent upon this timer elapsed.</p>
<b>Index</b>	Indicates the index of the interface flood lists for the area and autonomous system / OSPF routing domain. Indicated as area/as.
<b>Flood Queue Length</b>	Indicates the number of LSAs waiting to be flooded out an interface.
<b>Next</b>	Indicates the pointer to the flooding lists for next the LSAs to be flooded. Indicated by the index.
<b>Last Flood Scan Length</b>	Indicates the size of the last flooding list of LSAs flooded and the maximum size of the flooding list. 1 LSA is transmitted at a time when OSPF pacing is enabled.
<b>Last Flood Scan Time</b>	Indicates the time spent in the last flooding and the maximum time ever spent in flooding.
<b>Neighbor Count</b>	Indicates the number of OSPF neighbors discovered on an interface.
<b>Adjacent Neighbor Count</b>	Indicates the number of OSPF neighbors that have established FULL adjacency with the router. Adjacency means that their databases are fully synchronized.
<b>Suppress Hello</b>	When an OSPF demand circuit is established over a dialup link that incur usage-based costs, periodic OSPF Hello packets are suppressed to prevent the link from remains always up.

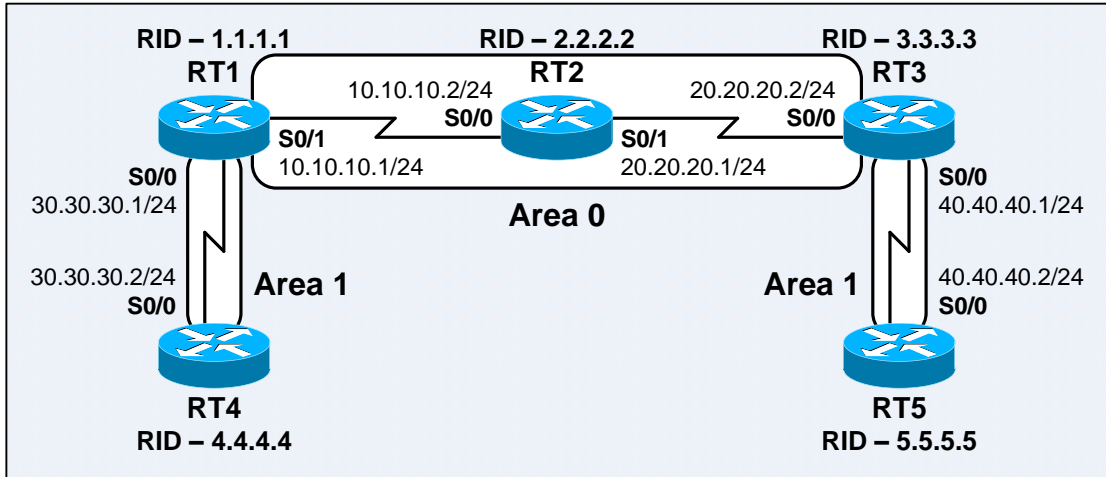
## OSPF Neighbor Relationship over Non-compatible OSPF Network Types

- OSPF routers with non-compatible OSPF network types can establish neighbor relationship and exchange routing information by adjusting the hello and dead intervals. OSPF network types that use a DR (**broadcast** and **non-broadcast**); and OSPF network types that do not use a DR (**point-to-multipoint** and **point-to-point**), can establish neighbor relationship with each other and function properly. However, OSPF routers with mixed DR and non-DR types will not function properly even if they have established FULL adjacency with each other. The "Adv Router is not-reachable" messages, which is due to a discrepancy in the OSPF LSDB, can be seen in the OSPF LSDBs when mixing DR and non-DR types.
- Below lists the possible working scenarios between OSPF routers:
  - i) Broadcast to Broadcast
  - ii) Non-Broadcast to Non-Broadcast
  - iii) Point-to-Point to Point-to-Point
  - iv) Point-to-Multipoint to Point-to-Multipoint
  - v) Broadcast to Non-Broadcast (adjust the hello and dead timers).  
**Note:** Both network types performs DR/BDR election, therefore they are compatible with each other, but not compatible with other network types.
  - vi) Point-to-Point to Point-to-Multipoint (adjust the hello and dead timers).

## OSPF Point-to-Multipoint Network Type and /32 Routes

- **Q:** Why the **point-to-multipoint** OSPF NBMA network type generates /32 routes and how to stop them from being advertised?  
**A:** The behavior of the point-to-multipoint advertises each node out as a /32 host route and suppress the advertisement of the network itself. Point-to-multipoint advertises nodes to overcome possible reachability issues between devices that are on the same logical subnet but do not have direct communication, eg: spoke-to-spoke communication over a hub-and-spoke network setup. The OSPF standard mentions that OSPF point-to-multipoint and loopback network types do not advertise the network itself but advertise a host route for each node. Suppressing the /32 host routes and advertise only the network can be achieved by either configures an OSPF network type other than point-to-multipoint; or configure the network to be in its own area, followed by using the **area range** command to summarize the /32 host routes so other routers receive only the summarized routes.

## Discontiguous Non-Zero OSPF Areas



**Figure A6-26:** Network Setup for OSPF Discontiguous Non-Zero Area

- The figure above shows a sample network setup with Area 1 separated by the Backbone Area 0. It is legal to have multiple discontiguous non-zero areas with the same Area ID, and it is not necessary to implement a virtual link or GRE tunnel to setup them in a contiguous manner.
- Below shows the routing table and OSPF LSDB on RT2:

```

RT2#sh ip route

Gateway of last resort is not set

    20.0.0.0/24 is subnetted, 1 subnets
C       20.20.20.0 is directly connected, Serial0/1
    40.0.0.0/24 is subnetted, 1 subnets
O IA    40.40.40.0 [110/128] via 20.20.20.2, 00:00:36, Serial0/1
    10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, Serial0/0
    30.0.0.0/24 is subnetted, 1 subnets
O IA    30.30.30.0 [110/128] via 10.10.10.1, 00:00:36, Serial0/0
RT2#
RT2#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address         Interface
3.3.3.3          0     FULL/-          00:00:37    20.20.20.2     Serial0/1
1.1.1.1          0     FULL/-          00:00:37    10.10.10.1     Serial0/0
RT2#
RT2#sh ip ospf database

                OSPF Router with ID (2.2.2.2) (Process ID 100)

                Router Link States (Area 0)

Link ID         ADV Router      Age             Seq#           Checksum Link count
1.1.1.1         1.1.1.1        47             0x80000002    0x001144 2
2.2.2.2         2.2.2.2        50             0x80000001    0x00F637 4
3.3.3.3         3.3.3.3        46             0x80000002    0x004ABD 2

                Summary Net Link States (Area 0)

Link ID         ADV Router      Age             Seq#           Checksum
30.30.30.0     1.1.1.1        48             0x80000001    0x00B1EC
40.40.40.0     3.3.3.3        48             0x80000001    0x000C6C
RT2#

```

- Below shows the routing table and OSPF LSDB on RT1:

```

RT1#sh ip route

Gateway of last resort is not set

    20.0.0.0/24 is subnetted, 1 subnets
O       20.20.20.0 [110/128] via 10.10.10.2, 00:00:35, Serial0/1
    40.0.0.0/24 is subnetted, 1 subnets
O IA   40.40.40.0 [110/192] via 10.10.10.2, 00:00:35, Serial0/1
    10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, Serial0/1
    30.0.0.0/24 is subnetted, 1 subnets
C       30.30.30.0 is directly connected, Serial0/0
RT1#
RT1#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address      Interface
2.2.2.2          0    FULL/ -         00:00:30   10.10.10.2   Serial0/1
4.4.4.4          0    FULL/ -         00:00:38   30.30.30.2   Serial0/0
RT1#
RT1#sh ip ospf database

                OSPF Router with ID (1.1.1.1) (Process ID 100)

                Router Link States (Area 0)

Link ID        ADV Router    Age           Seq#           Checksum Link count
1.1.1.1        1.1.1.1      44           0x80000002    0x001144  2
2.2.2.2        2.2.2.2      49           0x80000001    0x00F637  4
3.3.3.3        3.3.3.3      45           0x80000002    0x004ABD  2

                Summary Net Link States (Area 0)

Link ID        ADV Router    Age           Seq#           Checksum
30.30.30.0    1.1.1.1      45           0x80000001    0x00B1EC
40.40.40.0    3.3.3.3      47           0x80000001    0x000C6C

                Router Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum Link count
1.1.1.1        1.1.1.1      49           0x80000001    0x00DEF6  2
4.4.4.4        4.4.4.4      46           0x80000002    0x009C2C  2

                Summary Net Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum
10.10.10.0    1.1.1.1      45           0x80000001    0x008456
20.20.20.0    1.1.1.1      35           0x80000001    0x009DDE
40.40.40.0    1.1.1.1      35           0x80000001    0x004DB2
RT1#

```

- When an ABR builds LSAs for area 0, it takes the routing information from each of its other areas and includes them into Type-3 Summary LSAs, which do not contain any area information. Therefore, other backbone routers simply don't know what areas these destinations are belong to. The routers only know that in order to reach these destinations, the next hop is a given ABR.

- Below shows the routing table and OSPF LSDB on RT3:

```

RT3#sh ip route

Gateway of last resort is not set

    20.0.0.0/24 is subnetted, 1 subnets
C       20.20.20.0 is directly connected, Serial0/0
    40.0.0.0/24 is subnetted, 1 subnets
C       40.40.40.0 is directly connected, Serial0/1
    10.0.0.0/24 is subnetted, 1 subnets
O       10.10.10.0 [110/128] via 20.20.20.1, 00:00:38, Serial0/0
    30.0.0.0/24 is subnetted, 1 subnets
O IA   30.30.30.0 [110/192] via 20.20.20.1, 00:00:38, Serial0/0
RT3#
RT3#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address        Interface
2.2.2.2          0     FULL/ -         00:00:36   20.20.20.1    Serial0/0
5.5.5.5          0     FULL/ -         00:00:33   40.40.40.2    Serial0/1
RT3#
RT3#sh ip ospf database

                OSPF Router with ID (3.3.3.3) (Process ID 100)

                Router Link States (Area 0)

Link ID          ADV Router      Age             Seq#            Checksum Link count
1.1.1.1          1.1.1.1        49             0x80000002     0x001144  2
2.2.2.2          2.2.2.2        53             0x80000001     0x00F637  4
3.3.3.3          3.3.3.3        47             0x80000002     0x004ABD  2

                Summary Net Link States (Area 0)

Link ID          ADV Router      Age             Seq#            Checksum
30.30.30.0      1.1.1.1        51             0x80000001     0x00B1EC
40.40.40.0      3.3.3.3        48             0x80000001     0x000C6C

                Router Link States (Area 1)

Link ID          ADV Router      Age             Seq#            Checksum Link count
3.3.3.3          3.3.3.3        53             0x80000001     0x002C59  2
5.5.5.5          5.5.5.5        51             0x80000002     0x005824  2

                Summary Net Link States (Area 1)

Link ID          ADV Router      Age             Seq#            Checksum
10.10.10.0      3.3.3.3        39             0x80000001     0x00CAC7
20.20.20.0      3.3.3.3        49             0x80000001     0x00DED5
30.30.30.0      3.3.3.3        39             0x80000001     0x007A9B
RT3#

```

- Below shows the routing table and OSPF LSDB on RT4 and RT5:

```

RT4#sh ip route

Gateway of last resort is not set

    20.0.0.0/24 is subnetted, 1 subnets
O IA   20.20.20.0 [110/192] via 30.30.30.1, 00:00:40, Serial0/0
    40.0.0.0/24 is subnetted, 1 subnets
O IA   40.40.40.0 [110/256] via 30.30.30.1, 00:00:40, Serial0/0
    10.0.0.0/24 is subnetted, 1 subnets
O IA   10.10.10.0 [110/128] via 30.30.30.1, 00:00:42, Serial0/0
    30.0.0.0/24 is subnetted, 1 subnets
C      30.30.30.0 is directly connected, Serial0/0
RT4#
RT4#sh ip ospf database

        OSPF Router with ID (4.4.4.4) (Process ID 100)

        Router Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum Link count
1.1.1.1        1.1.1.1      56            0x80000001    0x00DEF6  2
4.4.4.4        4.4.4.4      51            0x80000002    0x009C2C  2

Summary Net Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum
10.10.10.0     1.1.1.1      52            0x80000001    0x008456
20.20.20.0     1.1.1.1      41            0x80000001    0x009DDE
40.40.40.0     1.1.1.1      41            0x80000001    0x004DB2
RT4#
=====
RT5#sh ip route

Gateway of last resort is not set

    20.0.0.0/24 is subnetted, 1 subnets
O IA   20.20.20.0 [110/128] via 40.40.40.1, 00:00:44, Serial0/0
    40.0.0.0/24 is subnetted, 1 subnets
C      40.40.40.0 is directly connected, Serial0/0
    10.0.0.0/24 is subnetted, 1 subnets
O IA   10.10.10.0 [110/192] via 40.40.40.1, 00:00:42, Serial0/0
    30.0.0.0/24 is subnetted, 1 subnets
O IA   30.30.30.0 [110/256] via 40.40.40.1, 00:00:42, Serial0/0
RT5#
RT5#sh ip ospf database

        OSPF Router with ID (5.5.5.5) (Process ID 100)

        Router Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum Link count
3.3.3.3        3.3.3.3      58            0x80000001    0x002C59  2
5.5.5.5        5.5.5.5      54            0x80000002    0x005824  2

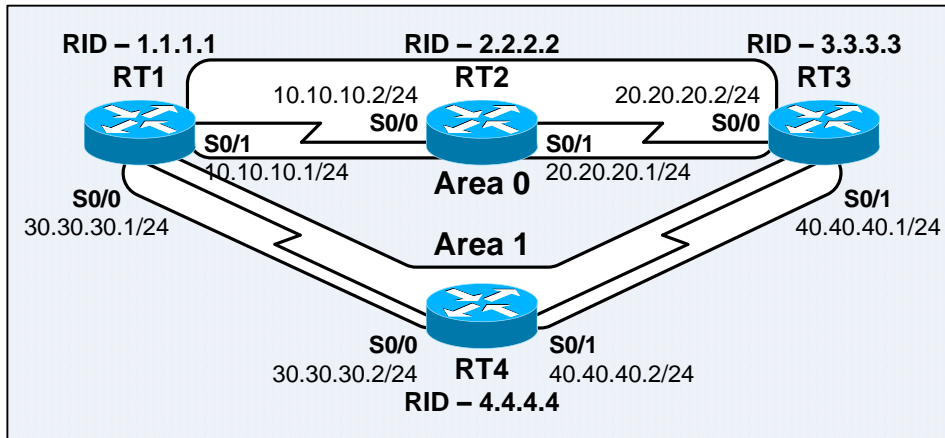
Summary Net Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum
10.10.10.0     3.3.3.3      43            0x80000001    0x00CAC7
20.20.20.0     3.3.3.3      53            0x80000001    0x00DED5
30.30.30.0     3.3.3.3      43            0x80000001    0x007A9B
RT5#

```



## OSPF Type-3 Network-Summary-LSAs Generations and Deletions



**Figure A6-27:** Network Setup for OSPF Type-3 Network-Summary-LSAs Generations and Deletions

- The following scenarios were setup to observe the generations and deletions of OSPF Type-3 Network-Summary-LSAs in a looped network topology:
  - 1) OSPF is configured on RT1, RT2, and RT3. RT4 not participating in OSPF routing yet. Area 1 is separated by Area 0.
  - 2) RT4 participates in OSPF routing after includes Serial0/0 in the **router ospf** configuration.
  - 3) RT4 includes Serial0/1 in the **router ospf** configuration.
  
- Below shows the routing table and OSPF LSDB on RT1 upon Scenario 1. RT1 calculates the route to 40.40.40.0/24 based on the (Area 1 → Area 0) Type-3 LSA generated by RT3. Note that RT1 also generates a (Area 0 → Area 1) Type-3 LSA into Area 1 on its Serial0/0. This is due to RT1 assumes that it is the only router that resides in Area 1.

```

RT1#sh ip route

Gateway of last resort is not set

--- output omitted ---
    40.0.0.0/24 is subnetted, 1 subnets
O IA 40.40.40.0 [110/192] via 10.10.10.2, 00:00:29, Serial0/1
RT1#
RT1#sh ip ospf database

        OSPF Router with ID (3.3.3.3) (Process ID 100)

--- output omitted ---
          Summary Net Link States (Area 0)

Link ID        ADV Router    Age         Seq#          Checksum
30.30.30.0     1.1.1.1      39         0x80000001   0x00B1EC
40.40.40.0     3.3.3.3      40         0x80000001   0x000C6C

--- output omitted ---
          Summary Net Link States (Area 1)

Link ID        ADV Router    Age         Seq#          Checksum
10.10.10.0     1.1.1.1      39         0x80000001   0x008456
20.20.20.0     1.1.1.1      29         0x80000001   0x009DDE
40.40.40.0     1.1.1.1      29         0x80000001   0x004DB2
RT1#
    
```

- The similar behaviors can be observed on RT3 as well – RT3 advertises 30.30.30.0/24 into Area 1 which is known via RT1 when RT1 advertises the network from Area 1 to Area 0.

```

RT3#sh ip route

Gateway of last resort is not set

--- output omitted ---
    30.0.0.0/24 is subnetted, 1 subnets
O IA    30.30.30.0 [110/192] via 20.20.20.1, 00:00:31, Serial0/0
RT3#
RT3#sh ip ospf database

        OSPF Router with ID (3.3.3.3) (Process ID 100)

--- output omitted ---
                Summary Net Link States (Area 0)

Link ID          ADV Router      Age              Seq#             Checksum
30.30.30.0      1.1.1.1        44              0x80000001      0x00B1EC
40.40.40.0      3.3.3.3        41              0x80000001      0x000C6C

--- output omitted ---
                Summary Net Link States (Area 1)

Link ID          ADV Router      Age              Seq#             Checksum
10.10.10.0      3.3.3.3        41              0x80000001      0x00CAC7
20.20.20.0      3.3.3.3        41              0x80000001      0x00DED5
30.30.30.0      3.3.3.3        31              0x80000001      0x007A9B
RT3#

```

- **Conclusion:** RT1 and RT3 advertise the Type-3 LSAs learnt from each other via Area 0 as Type-3 LSAs into Area 1.
- Below shows the routing table and OSPF LSDB on RT4 upon Scenario 2. It receives a Type-3 LSA for 40.40.40.0/24 from RT1. Note that the routing tables and OSPF LSDBs on RT1, RT2, and RT3 are identical as upon Scenario 1.

```

RT4#sh ip route

Gateway of last resort is not set

    20.0.0.0/24 is subnetted, 1 subnets
O IA    20.20.20.0 [110/192] via 30.30.30.1, 00:00:58, Serial0/0
    40.0.0.0/24 is subnetted, 1 subnets
C       40.40.40.0 is directly connected, Serial0/1
    10.0.0.0/24 is subnetted, 1 subnets
O IA    10.10.10.0 [110/128] via 30.30.30.1, 00:00:58, Serial0/0
    30.0.0.0/24 is subnetted, 1 subnets
C       30.30.30.0 is directly connected, Serial0/0
RT4#sh ip ospf database

        OSPF Router with ID (4.4.4.4) (Process ID 100)

--- output omitted ---
                Summary Net Link States (Area 1)

Link ID          ADV Router      Age              Seq#             Checksum
10.10.10.0      1.1.1.1        277             0x80000001      0x008456
20.20.20.0      1.1.1.1        267             0x80000001      0x009DDE
40.40.40.0      1.1.1.1        267             0x80000001      0x004DB2
RT4#

```

- Below shows the routing table and OSPF LSDB on RT4 upon Scenario 3. Take note upon the Type-3 LSAs advertised by RT1 and RT3 from Area 0 into Area 1. RT1 and RT3 no longer advertising 30.30.30.0/24 and 40.40.40.0/24 into Area 1, as RT1, RT3, and RT4 recognize each other in Area 1 via Type-1 Router-LSAs in Area 1.

```

RT4#sh ip route

Gateway of last resort is not set

    20.0.0.0/24 is subnetted, 1 subnets
O IA   20.20.20.0 [110/128] via 40.40.40.1, 00:01:02, Serial0/1
    40.0.0.0/24 is subnetted, 1 subnets
C      40.40.40.0 is directly connected, Serial0/1
    10.0.0.0/24 is subnetted, 1 subnets
O IA   10.10.10.0 [110/128] via 30.30.30.1, 00:01:02, Serial0/0
    30.0.0.0/24 is subnetted, 1 subnets
C      30.30.30.0 is directly connected, Serial0/0
RT4#
RT4#sh ip ospf database

        OSPF Router with ID (4.4.4.4) (Process ID 100)

        Router Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum Link count
1.1.1.1        1.1.1.1      311          0x80000002    0x00DCF7 2
3.3.3.3        3.3.3.3      77           0x80000002    0x000880 2
4.4.4.4        4.4.4.4      77           0x80000002    0x00C466 4

        Summary Net Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum
10.10.10.0     1.1.1.1      515          0x80000001    0x008456
10.10.10.0     3.3.3.3      514          0x80000001    0x00CAC7
20.20.20.0     1.1.1.1      505          0x80000001    0x009DDE
20.20.20.0     3.3.3.3      514          0x80000001    0x00DED5
RT4#

```

## OSPF Link-State Database Overload Protection

- A misconfigured OSPF router may generate large numbers of LSAs and these excessive LSAs can drain the CPU and memory resources on other routers. OSPF LSDB Overload Protection can be configured with the **max-lsa** *max-num* [*threshold-percentage*] [**warning-only** | **ignore-time** *minutes* , **ignore-count** *count-num* , **reset-time** *minutes*] OSPF router subcommand which available on Cisco IOS Release 12.3(7)T and later to protect against the mentioned issue by defining the maximum number of received (non-self-generated) LSAs that an OSPF process will receive and keep in the OSPF LSDB.

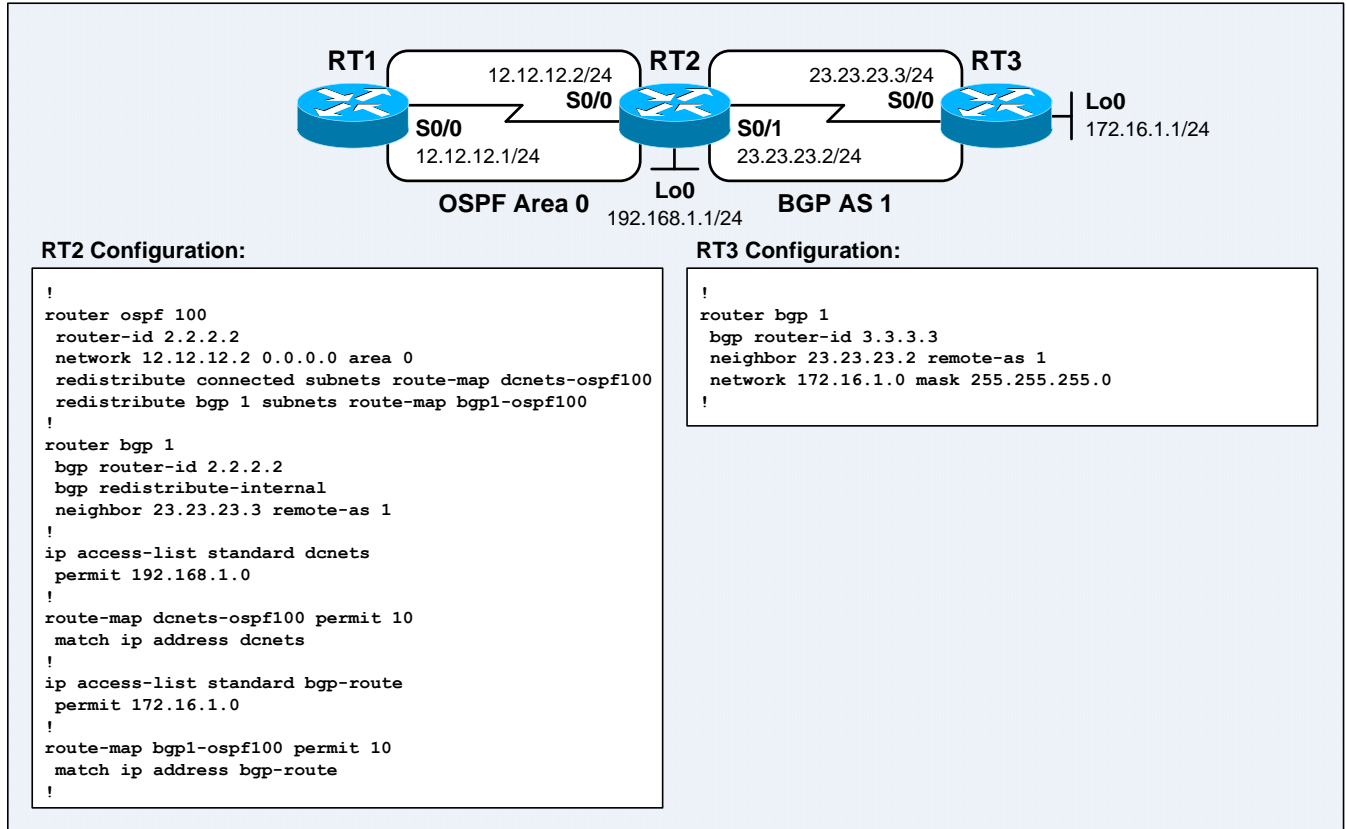
- **Note:** When the **max-lsa** OSPF router subcommand is entered for the first time or when any parameter is changed, the OSPF process would undergo a soft-reset procedure.

- Below describes the parameters of the **max-lsa** OSPF router subcommand:

Parameter	Description
<b>max-num</b>	The maximum number of received (non-self-generated LSAs) that an OSPF process will receive and keep in the OSPF LSDB.
<b>threshold-percentage</b>	(Optional) The percentage of the maximum LSA number as specified by the <i>max-num</i> parameter, at which a warning message is logged. Default is 75%.
<b>warning-only</b>	(Optional) Specifies that only a warning message is sent when the maximum LSA limit is exceeded. The OSPF process never enters into the ignore state and continues in its normal operation. Disabled by default.
<b>ignore-time</b>	(Optional) Specifies the period (in minutes) to remains in the ignore state when the maximum LSA limit is exceeded. Default is 5 minutes.
<b>ignore-count</b>	(Optional) Specifies the number of times that an OSPF process can consecutively be placed into the ignore state before it remains in the ignore state permanently and requires manual intervention. Default is 5 times.
<b>reset-time</b>	(Optional) Specifies the period (in minutes) in which the ignore state counter is reset to 0 if the OSPF process remains normal for the defined period after returned from the ignore state back to normal state. Default is 10 minutes.

- An error message is logged when the number of received (non-self-generated) LSAs in the LSDB has exceeded the configured threshold value. If the threshold being exceeded for 1 minute, the OSPF process will enter into the **ignore state** which will tear down all OSPF adjacencies and clear the OSPF LSDB. No OSPF packets will be sent nor received by interfaces that belong to the OSPF process for the period that is defined by the **ignore-time** parameter.
- To prevent an OSPF process from endlessly switching from the normal state of operation to ignore state after it returns from the ignore state to the normal state of operation for 1 minute due to the maximum LSA limit is exceeded, the OSPF process keeps a counter for the number of times the process went into the ignore state – **ignore count**. The OSPF process remains in the ignore state permanently once the *count-num* parameter is exceeded. The **clear ip ospf** privileged command must be issued to recover the OSPF process to the normal state of operation.
- When an OSPF router is being placed into the permanent ignore state, try to identify the router that is generating the excessive LSAs and correct the problem. It is also recommended to increase the limit that has been configured by the **max-lsa** command before try to bring the router back to normal operation.

## OSPF External Route Default Metrics



**Figure A6-28:** Network Setup for OSPF External Route Default Metrics

- A default metric of 1 will be used for redistributed BGP routes; while a default metric of 20 will be used for routes redistributed from other routing protocols, including static routes and directly connected interfaces. Below shows the routing table on RT1 as the proof of the statement above:

```
RT1#sh ip route
```

```
Gateway of last resort is not set
```

```

      172.16.0.0/24 is subnetted, 1 subnets
O E2 172.16.1.0 [110/1] via 12.12.12.2, 00:00:19, Serial0/0
      12.0.0.0/24 is subnetted, 1 subnets
C      12.12.12.0 is directly connected, Serial0/0
O E2 192.168.1.0/24 [110/20] via 12.12.12.2, 00:00:49, Serial0/0
RT1#

```

```
RT1#sh ip route 172.16.1.0
```

```

Routing entry for 172.16.1.0/24
  Known via "ospf 100", distance 110, metric 1, type extern 2, forward metric 64
  Last update from 12.12.12.2 on Serial0/0, 00:00:19 ago
  Routing Descriptor Blocks:
  * 12.12.12.2, from 2.2.2.2, 00:00:19 ago, via Serial0/0
    Route metric is 1, traffic share count is 1
RT1#

```

```
RT1#
```

```
RT1#sh ip route 192.168.1.0
```

```

Routing entry for 192.168.1.0/24
  Known via "ospf 100", distance 110, metric 20, type extern 2, forward metric 64
  Last update from 12.12.12.2 on Serial0/0, 00:00:49 ago
  Routing Descriptor Blocks:
  * 12.12.12.2, from 2.2.2.2, 00:00:49 ago, via Serial0/0
    Route metric is 20, traffic share count is 1
RT1#

```

```
RT1#
```

## OSPF Forward Metric in External Type-2 (E2) Route Selection

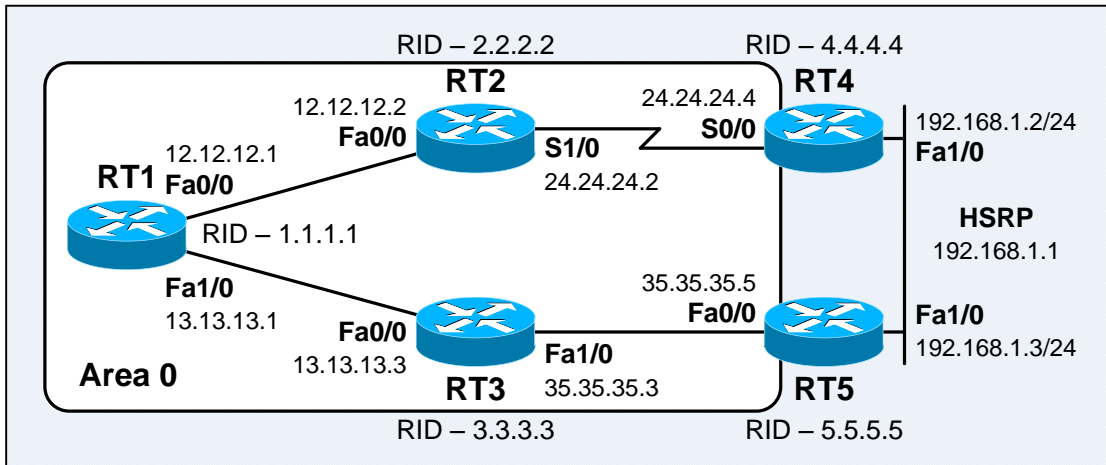


Figure A6-29: Network Setup for OSPF External Type-2 (E2) Route Selection

- RT4 and RT5 redistribute their directly connected network 192.168.1.0/24 into the OSPF routing domain as External Type-2 (E2) route. RT1 selects the path via RT3 instead of RT2 as the **forward metric** to the E2 route is 2 (the metric between RT1-RT3 and RT3-RT5 links is 1). Forward metric is the cost to reach an ASBR. Note that RT2 also selects the path via RT1-RT3 as it has a lower forward metric to reach RT5, the nearest ASBR in the OSPF routing domain.
- Below shows the routing table and OSPF LSDB on RT1:

```

RT1#sh ip route

Gateway of last resort is not set

 35.0.0.0/24 is subnetted, 1 subnets
O   35.35.35.0 [110/2] via 13.13.13.3, 00:00:36, FastEthernet1/0
 24.0.0.0/24 is subnetted, 1 subnets
O   24.24.24.0 [110/65] via 12.12.12.2, 00:00:36, FastEthernet0/0
 12.0.0.0/24 is subnetted, 1 subnets
C   12.12.12.0 is directly connected, FastEthernet0/0
O E2 192.168.1.0/24 [110/20] via 13.13.13.3, 00:00:36, FastEthernet1/0
 13.0.0.0/24 is subnetted, 1 subnets
C   13.13.13.0 is directly connected, FastEthernet1/0
RT1#
RT1#sh ip ospf database

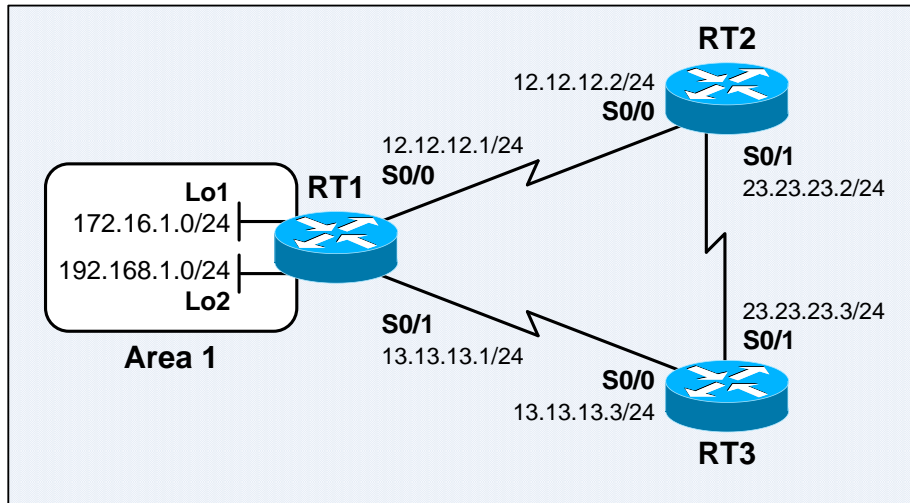
--- output omitted ---

                        Type-5 AS External Link States

Link ID          ADV Router    Age         Seq#          Checksum Tag
192.168.1.0     4.4.4.4      91         0x80000001   0x00B273 0
192.168.1.0     5.5.5.5      91         0x80000001   0x00948D 0
RT1#
RT1#sh ip route 192.168.1.1
Routing entry for 192.168.1.0/24
  Known via "ospf 100", distance 110, metric 20, type extern 2, forward metric 2
  Last update from 13.13.13.3 on FastEthernet1/0, 00:00:36 ago
  Routing Descriptor Blocks:
    * 13.13.13.3, from 5.5.5.5, 00:00:36 ago, via FastEthernet1/0
      Route metric is 20, traffic share count is 1
RT1#

```

## OSPF Route Filtering within an Area



**Figure A6-30:** Network Setup for OSPF Route Filtering within an Area

- Performing OSPF route filtering within an area does not affect routes as they enter the OSPF topology database, but the IP routing table instead; and on only the router on which the route filtering is configured. Route filtering on link-state routing protocols often result in different routing tables (but same topology database) on routers, which would then introduce routing loops or routing black holes to the network if not implemented carefully.
- OSPF route filtering within an area can be achieved using distribute lists (in conjunction with access lists, prefix lists, and route maps), or modifying the administrative distance.
- The routing table and OSPF topology database on RT3 before implementing route filtering:

```
RT3#sh ip route
```

```
Gateway of last resort is not set
```

```

    23.0.0.0/24 is subnetted, 1 subnets
C       23.23.23.0 is directly connected, Serial0/1
    172.16.0.0/24 is subnetted, 1 subnets
O IA    172.16.1.0 [110/65] via 13.13.13.1, 00:00:42, Serial0/0
    12.0.0.0/24 is subnetted, 1 subnets
O       12.12.12.0 [110/128] via 23.23.23.2, 00:00:42, Serial0/1
        [110/128] via 13.13.13.1, 00:00:42, Serial0/0
O IA    192.168.1.0/24 [110/65] via 13.13.13.1, 00:00:42, Serial0/0
    13.0.0.0/24 is subnetted, 1 subnets
C       13.13.13.0 is directly connected, Serial0/0

```

```
RT3#
```

```
RT3#sh ip ospf database
```

```
OSPF Router with ID (3.3.3.3) (Process ID 100)
```

```
Router Link States (Area 0)
```

Link ID	ADV Router	Age	Seq#	Checksum	Link count
1.1.1.1	1.1.1.1	48	0x80000003	0x00DD70	4
2.2.2.2	2.2.2.2	48	0x80000001	0x000807	4
3.3.3.3	3.3.3.3	47	0x80000002	0x00BC45	4

```
Summary Net Link States (Area 0)
```

Link ID	ADV Router	Age	Seq#	Checksum
172.16.1.0	1.1.1.1	59	0x80000001	0x00E495
192.168.1.0	1.1.1.1	59	0x80000001	0x00B815

```
RT3#
```

- Below shows the routing table and OSPF topology database on RT3 after implemented route filtering to filter 192.168.1.0/24 from entering the routing table. Route filtering configuration using the **distribute-list** router subcommand does not modify the OSPF topology database and does not prevent the LSAs from being propagated to other routers!

```

RT3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT3(config)#access-list 1 permit 172.16.1.0 0.0.0.255
RT3(config)#router ospf 100
RT3(config-router)#distribute-list ?
<1-199>      IP access list number
<1300-2699>  IP expanded access list number
WORD         Access-list name
gateway      Filtering incoming updates based on gateway
prefix       Filter prefixes in routing updates
route-map    Filter prefixes based on the route-map

RT3(config-router)#distribute-list 1 in
RT3(config-router)#end
RT3#
RT3#sh ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address      Interface
2.2.2.2          0    FULL/ -         00:00:32   23.23.23.2   Serial0/1
1.1.1.1          0    FULL/ -         00:00:38   13.13.13.1   Serial0/0
RT3#
RT3#sh ip route

Gateway of last resort is not set

    23.0.0.0/24 is subnetted, 1 subnets
C       23.23.23.0 is directly connected, Serial0/1
    172.16.0.0/24 is subnetted, 1 subnets
O IA    172.16.1.0 [110/65] via 13.13.13.1, 00:00:08, Serial0/0
    13.0.0.0/24 is subnetted, 1 subnets
C       13.13.13.0 is directly connected, Serial0/0
RT3#
RT3#sh ip ospf database

                OSPF Router with ID (3.3.3.3) (Process ID 100)

                Router Link States (Area 0)

Link ID          ADV Router      Age             Seq#             Checksum Link count
1.1.1.1          1.1.1.1        198             0x80000003      0x00DD70  4
2.2.2.2          2.2.2.2        198             0x80000001      0x000807  4
3.3.3.3          3.3.3.3        196             0x80000002      0x00BC45  4

                Summary Net Link States (Area 0)

Link ID          ADV Router      Age             Seq#             Checksum
172.16.1.0      1.1.1.1        209             0x80000001      0x00E495
192.168.1.0     1.1.1.1        209             0x80000001      0x00B815
RT3#

```



- Below shows a routing black hole issue occurred in the sample network when the link between RT1 and RT2 is failed. RT2 still forward packets destined to 192.168.1.0/24 via RT3 but apparently RT3 does not have the route to the destination network due to route filtering!

```

RT2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT2(config)#int s0/0
RT2(config-if)#shut
RT2(config-if)#
00:04:33: %OSPF-5-ADJCHG: Process 100, Nbr 1.1.1.1 on Serial0/0 from FULL to
DOWN, Neighbor Down: Interface down or detached
--- output omitted ---
RT2(config-if)#^Z
RT2#
RT2#sh ip ospf database

                OSPF Router with ID (2.2.2.2) (Process ID 100)

                Router Link States (Area 0)

Link ID        ADV Router    Age           Seq#           Checksum Link count
1.1.1.1        1.1.1.1      29            0x80000004    0x0050EC  2
2.2.2.2        2.2.2.2      61            0x80000002    0x00D624  2
3.3.3.3        3.3.3.3      324           0x80000002    0x00BC45  4

                Summary Net Link States (Area 0)

Link ID        ADV Router    Age           Seq#           Checksum
172.16.1.0     1.1.1.1      334           0x80000001    0x00E495
192.168.1.0    1.1.1.1      334           0x80000001    0x00B815
RT2#
RT2#sh ip route

Gateway of last resort is not set

    23.0.0.0/24 is subnetted, 1 subnets
C       23.23.23.0 is directly connected, Serial0/1
    172.16.0.0/24 is subnetted, 1 subnets
O IA    172.16.1.0 [110/129] via 23.23.23.3, 00:00:33, Serial0/1
O IA 192.168.1.0/24 [110/129] via 23.23.23.3, 00:00:33, Serial0/1
    13.0.0.0/24 is subnetted, 1 subnets
O       13.13.13.0 [110/128] via 23.23.23.3, 00:00:33, Serial0/1
RT2#
RT2#ping 172.16.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/91/140 ms
RT2#
RT2#debug ip icmp
ICMP packet debugging is on
RT2#
RT2#ping 192.168.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
U
00:06:06: ICMP: dst (23.23.23.2) host unreachable rcv from 23.23.23.3.U
00:06:08: ICMP: dst (23.23.23.2) host unreachable rcv from 23.23.23.3.U
Success rate is 0 percent (0/5)
RT2#
00:06:10: ICMP: dst (23.23.23.2) host unreachable rcv from 23.23.23.3
RT2#

```

- Below shows another alternative route filtering configuration on RT3 to achieve the same result – filter 192.168.1.0/24 from entering the routing table, using a distribute list in conjunction with route map and prefix list.

```

RT3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT3(config)#ip prefix-list Area0-permit permit 172.16.1.0/24
RT3(config)#
RT3(config)#route-map Area0-Filter
RT3(config-route-map)#match ip address prefix-list Area0-permit
RT3(config-route-map)#
RT3(config-route-map)#router ospf 100
RT3(config-router)#distribute-list route-map Area0-Filter in
RT3(config-router)#end
RT3#
RT3#sh ip route

Gateway of last resort is not set

    23.0.0.0/24 is subnetted, 1 subnets
C       23.23.23.0 is directly connected, Serial0/1
    172.16.0.0/24 is subnetted, 1 subnets
O IA    172.16.1.0 [110/65] via 13.13.13.1, 00:00:03, Serial0/0
    13.0.0.0/24 is subnetted, 1 subnets
C       13.13.13.0 is directly connected, Serial0/0
RT3#
RT3#sh ip ospf database

                OSPF Router with ID (3.3.3.3) (Process ID 100)

                Router Link States (Area 0)

Link ID        ADV Router    Age           Seq#           Checksum Link count
1.1.1.1        1.1.1.1      41            0x80000002    0x00DF6F 4
2.2.2.2        2.2.2.2      43            0x80000001    0x000807 4
3.3.3.3        3.3.3.3      40            0x80000002    0x00BC45 4

                Summary Net Link States (Area 0)

Link ID        ADV Router    Age           Seq#           Checksum
172.16.1.0     1.1.1.1      42            0x80000001    0x00E495
192.168.1.0    1.1.1.1      42            0x80000001    0x00B815
RT3#
RT3#sh route-map
route-map Area0-Filter, permit, sequence 10
  Match clauses:
    ip address prefix-lists: Area0-permit
  Set clauses:
  Policy routing matches: 0 packets, 0 bytes
RT3#
RT3#sh ip prefix-list
ip prefix-list Area0-permit: 1 entries
  seq 5 permit 172.16.1.0/24
RT3#

```

- Below shows another alternative route filtering configuration on RT3 to achieve the same result – filter 192.168.1.0/24 from entering the routing table, by modifying the administrative distance.

```

RT3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT3(config)#access-list 1 permit 192.168.1.0 0.0.0.255
RT3(config)#
RT3(config)#router ospf 100
RT3(config-router)#distance 255 0.0.0.0 255.255.255.255 1
RT3(config-router)#end
RT3#
RT3#sh ip route

Gateway of last resort is not set

    23.0.0.0/24 is subnetted, 1 subnets
C       23.23.23.0 is directly connected, Serial0/1
    172.16.0.0/24 is subnetted, 1 subnets
O IA    172.16.1.0 [110/65] via 13.13.13.1, 00:00:04, Serial0/0
    12.0.0.0/24 is subnetted, 1 subnets
O       12.12.12.0 [110/128] via 23.23.23.2, 00:00:04, Serial0/1
        [110/128] via 13.13.13.1, 00:00:04, Serial0/0
    13.0.0.0/24 is subnetted, 1 subnets
C       13.13.13.0 is directly connected, Serial0/0
RT3#
RT3#sh ip ospf database

                OSPF Router with ID (3.3.3.3) (Process ID 100)

                Router Link States (Area 0)

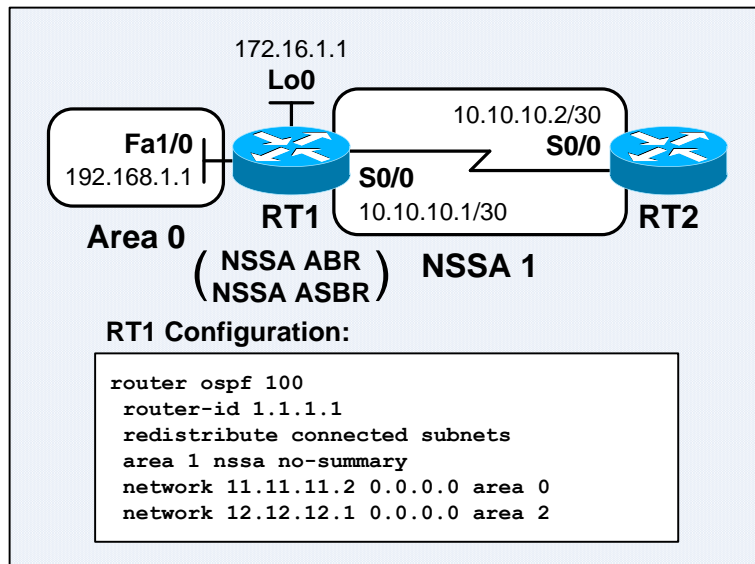
Link ID        ADV Router    Age           Seq#           Checksum Link count
1.1.1.1        1.1.1.1      94            0x80000002    0x00DF6F  4
2.2.2.2        2.2.2.2      98            0x80000001    0x000807  4
3.3.3.3        3.3.3.3      93            0x80000002    0x00BC45  4

                Summary Net Link States (Area 0)

Link ID        ADV Router    Age           Seq#           Checksum
172.16.1.0     1.1.1.1      95            0x80000001    0x00E495
192.168.1.0    1.1.1.1      95            0x80000001    0x00B815
RT3#

```

## The area *area-id* nssa no-redistribution OSPF Router Subcommand



**Figure A6-31:** Network Setup for the **area *area-id* nssa no-redistribution** OSPF Router Subcommand

- The **no-redistribution** keyword is being used in situations where there is no need to inject external routes into an NSSA as Type-7 LSAs, eg: when an NSSA ASBR is also an NSSA ABR.
- Below shows the routing table and OSPF LSDB on RT2 before implementing the **no-redistribution** keyword on RT1:

```

RT2#sh ip route

Gateway of last resort is 10.10.10.1 to network 0.0.0.0

    172.16.0.0/32 is subnetted, 1 subnets
O N2   172.16.1.1 [110/20] via 10.10.10.1, 00:00:10, Serial10/0
    10.0.0.0/30 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, Serial10/0
O*IA 0.0.0.0/0 [110/65] via 10.10.10.1, 00:00:12, Serial10/0
RT2#
RT2#sh ip ospf database

                OSPF Router with ID (2.2.2.2) (Process ID 100)

                Router Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum Link count
1.1.1.1        1.1.1.1      17           0x80000002    0x006EE1  2
2.2.2.2        2.2.2.2      20           0x80000001    0x000748  2

                Summary Net Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum
0.0.0.0        1.1.1.1      21           0x80000001    0x001B17

                Type-7 AS External Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum Tag
172.16.1.1    1.1.1.1      15           0x80000001    0x0013C8  0
RT2#
RT2#debug ip routing
IP routing debugging is on
RT2#
    
```

- Below shows the configuration implemented on RT1 as well as the routing table and OSPF LSDB on RT2 after implemented the **no-redistribution** keyword on RT1:

```

RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#router ospf 100
RT1(config-router)#area 1 nssa no-redistribution
RT1(config-router)#do sh run | b router ospf
router ospf 100
  router-id 1.1.1.1
  log-adjacency-changes
  area 1 nssa no-redistribution no-summary
  redistribute connected subnets
  network 10.10.10.1 0.0.0.0 area 1
  network 192.168.1.1 0.0.0.0 area 0
!
=====
RT2#
00:00:42: RT: del 0.0.0.0 via 10.10.10.1, ospf metric [110/65]
00:00:42: RT: delete network route to 0.0.0.0
00:00:42: RT: NET-RED 0.0.0.0/0
00:00:42: RT: NET-RED 0.0.0.0/0
00:00:43: RT: del 172.16.1.1/32 via 10.10.10.1, ospf metric [110/20]
00:00:43: RT: delete subnet route to 172.16.1.1/32
00:00:43: RT: NET-RED 172.16.1.1/32
00:00:43: RT: delete network route to 172.16.0.0
00:00:43: RT: NET-RED 172.16.0.0/16
00:00:47: RT: add 0.0.0.0/0 via 10.10.10.1, ospf metric [110/65]
00:00:47: RT: NET-RED 0.0.0.0/0
00:00:47: RT: default path is now 0.0.0.0 via 10.10.10.1
00:00:47: RT: new default network 0.0.0.0
00:00:47: RT: NET-RED 0.0.0.0/0
RT2#
RT2#sh ip route

Gateway of last resort is 10.10.10.1 to network 0.0.0.0

    10.0.0.0/30 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, Serial0/0
O*IA 0.0.0.0/0 [110/65] via 10.10.10.1, 00:00:04, Serial0/0
RT2#
RT2#sh ip ospf database

                OSPF Router with ID (2.2.2.2) (Process ID 100)

                Router Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum Link count
1.1.1.1        1.1.1.1      46           0x80000002    0x006EE1  2
2.2.2.2        2.2.2.2      50           0x80000001    0x000748  2

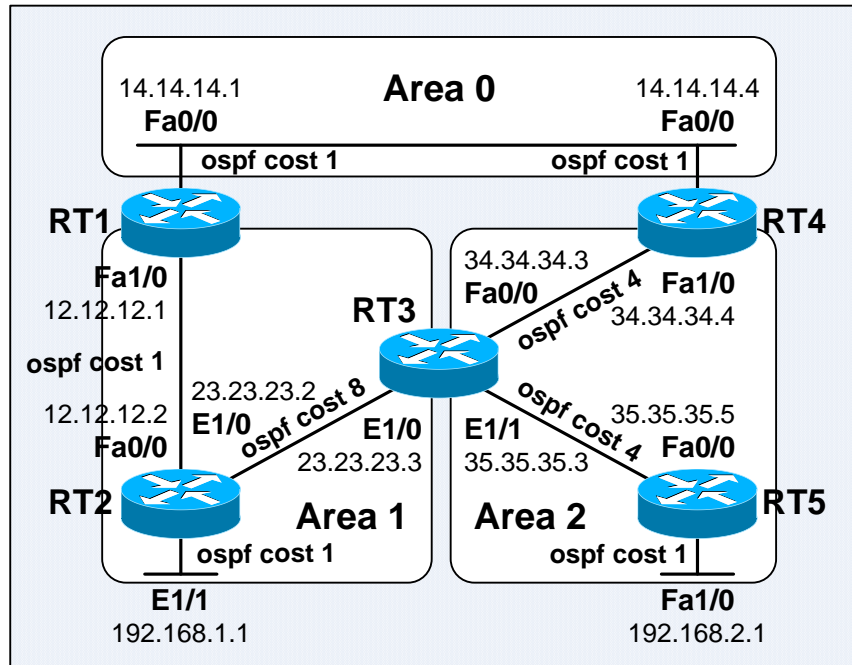
                Summary Net Link States (Area 1)

Link ID        ADV Router    Age           Seq#           Checksum
0.0.0.0        1.1.1.1      6            0x80000002    0x001918
RT2#

```

**Note:** There was a network interruption (~ 5 seconds) upon the default route when implementing the **no-redistribution** keyword on RT1.

## When OSPF Inter-Area Routing Bypasses the Backbone Area



**Figure A6-32:** Inter-Area Asymmetry Routing

- According to RFC 2328 – OSPF Version 2, an **area border router (ABR)** is a router attached to multiple areas, and has the main function to provide its attached areas with Type-3 and Type-4 LSAs that describe routes and ASBRs in other areas, as well as perform actual inter-area routing.
- Based on RFC 2328, RT3 will identify itself as an ABR by setting the B bit in its Router-LSA, and generates Type-3 Summary-LSAs for the destinations in its directly attached areas into Areas 1 and 2 – originates Summary-LSAs for Area 1 to Area 2; and vice versa.
- The Cisco's implementation of ABRs is based on RFC 3509 – Alternative Implementations of OSPF Area Border Routers, which redefines an ABR as a router that has more than one area actively attached and one of them is the backbone area! RT3 is no longer considered as an ABR; this is proven when the output of the **show ip ospf database router self-originate EXEC** command on RT3 does not have the "Area Border Router" message as in RT1 and RT4. Cisco using this approach to prevent loops due to the origination of Summary-LSAs in situations where a router considers itself as an ABR but doesn't have an active backbone connection. Deployment of the Cisco's implementation of ABRs based on RFC 3509 is fully compatible and can coexist with standard OSPF domains without problems.
- Deployment of either OSPF-standard or RFC3509-based ABRs can lead to unexpected routing asymmetry as discussed in this section.
- RT2 will have the inter-area route to 192.168.2.0/24 via RT1 (ABR); while RT5 will have the inter-area route to 192.168.1.0/24 via RT4 (ABR). Since RT4 is only reachable via RT3, the traffic initiated from 192.168.2.0/24 and destined to 192.168.1.0/24 will pass through RT3. Due to RT3 has an intra-area route to 192.168.1.0/24 via RT2, RT3 will route the traffic directly to RT2 instead of RT4, as intra-area routes are always preferred over inter-area routes.  
**Note:** The OSPF cost from RT3 to 192.168.1.0/24 via RT2 is **9** while the OSPF cost via RT4 is **7**.

- The return traffic from 192.168.1.0/24 back to 192.168.2.0/24 will pass through RT2 and routed to RT1 instead of RT3, as RT2 do not have any inter-area routes via RT3. As a result, routing asymmetry occurs and the more optimal path through the backbone area is not being used as the traffic from 192.168.1.0/24 to 192.168.2.0/24 will go through the backbone area while traffic from 192.168.2.0/24 to 192.168.1.0/24 will cross the areas directly through RT3. Below shows the output on RT3 which proves that asymmetric routing has occurred:

```

RT3#sh int | in is up|rate
FastEthernet0/0 is up, line protocol is up
 30 second input rate 18000 bits/sec, 20 packets/sec
 30 second output rate 0 bits/sec, 0 packets/sec
Ethernet1/0 is up, line protocol is up
 30 second input rate 0 bits/sec, 0 packets/sec
 30 second output rate 18000 bits/sec, 20 packets/sec
Ethernet1/1 is up, line protocol is up
 30 second input rate 18000 bits/sec, 20 packets/sec
 30 second output rate 18000 bits/sec, 20 packets/sec
RT3#

```

- The RFC3509-based-ABR RT3 is not really functioning as an ABR as it does not originate Summary-LSAs into its attached areas, but inter-area traffic can still pass through it.

### OSPF Stub Areas Attack Session

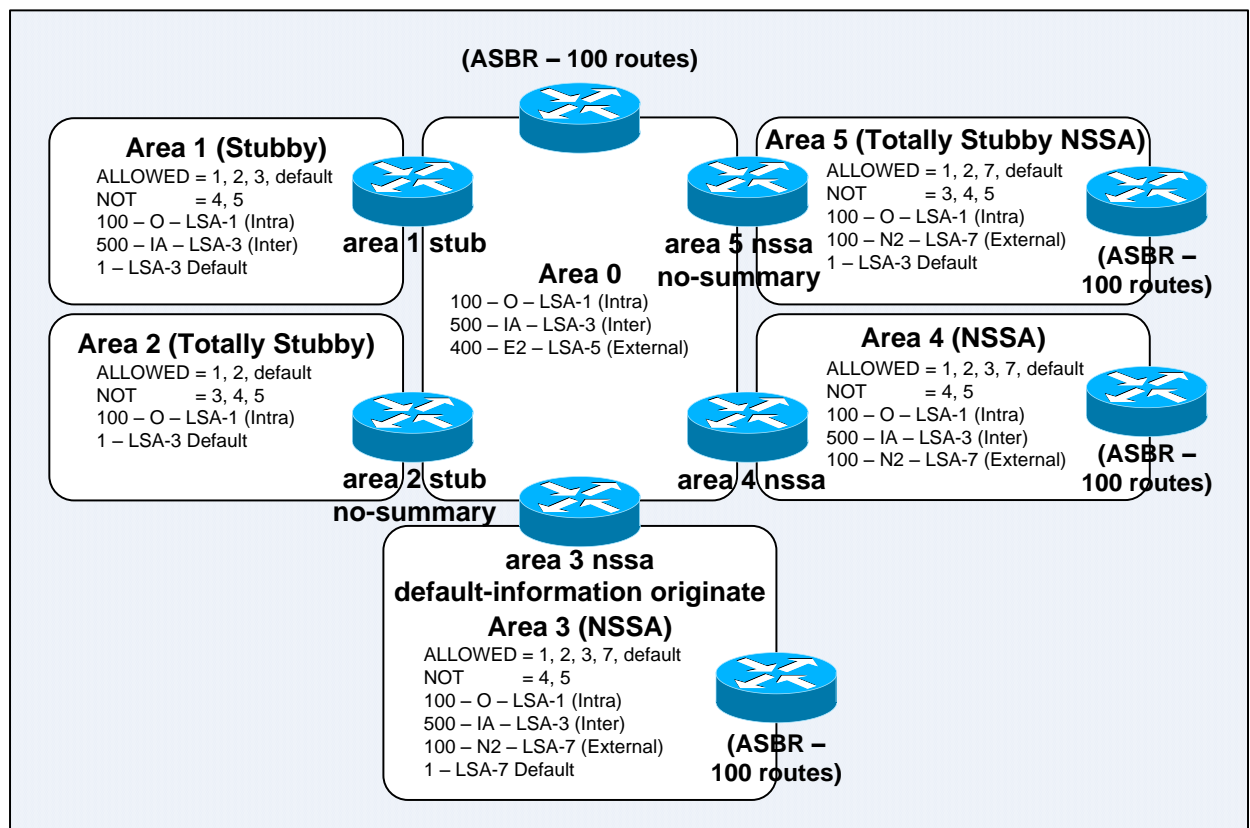
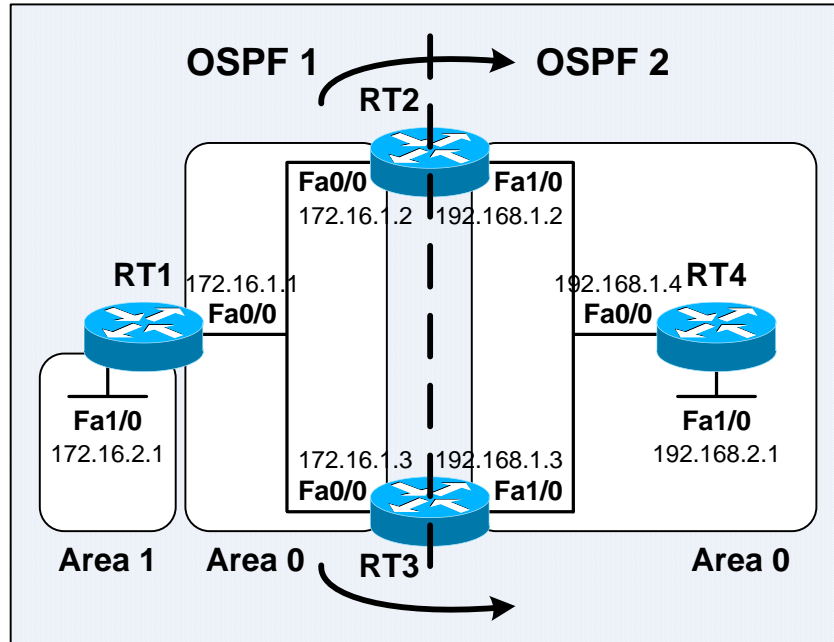


Figure A6-33: OSPF Stub Areas Attack Session

- The figure above is used to test the understanding of the features of different types of stub areas, eg: the generation of the default routes upon different types of stub areas, the number of inter-area OSPF routes that will be received for different types of stub areas, etc.

## Suboptimal Routing when Redistributing between OSPF Processes



**Figure A6-34:** Suboptimal Routing when Redistributing between OSPF Processes

- When redistributing between different OSPF processes in multiple points on a network, it is possible to enter into situations of suboptimal routing or even worse, a routing loop.
- The figure above shows a sample OSPF redistribution network. Both RT2 and RT3 are running 2 OSPF processes and redistributing from OSPF process 1 into OSPF process 2.
- RT1 is an ABR that advertises 172.16.2.0/24 as an inter-area (IA) OSPF route to RT2 and RT3. RT2 and RT3 are redistributing this route from OSPF process 1 into OSPF process 2. Therefore, both RT2 and RT3 will learn about 172.16.2.0/24 as IA route via OSPF process 1 and as E2 route via OSPF process 2.
- Since IA routes are always preferred over E1/E2 routes, the routing tables of RT2 and RT3 should show 172.16.2.0/24 as an IA route with RT1 as the next hop. However, there are chances that the route will be installed into the routing table as an E2 route.

```
RT2#sh ip route
```

```
Gateway of last resort is not set
```

```
172.16.0.0/24 is subnetted, 2 subnets
```

```
C 172.16.1.0 is directly connected, FastEthernet0/0
```

```
O IA 172.16.2.0 [110/2] via 172.16.1.1, 00:00:36, FastEthernet0/0
```

```
C 192.168.1.0/24 is directly connected, FastEthernet1/0
```

```
O 192.168.2.0/24 [110/2] via 192.168.1.4, 00:01:06, FastEthernet1/0
```

```
RT2#
```

```
RT3#sh ip route
```

```
Gateway of last resort is not set
```

```
172.16.0.0/24 is subnetted, 2 subnets
```

```
C 172.16.1.0 is directly connected, FastEthernet0/0
```

```
O E2 172.16.2.0 [110/2] via 192.168.1.2, 00:00:58, FastEthernet1/0
```

```
C 192.168.1.0/24 is directly connected, FastEthernet1/0
```

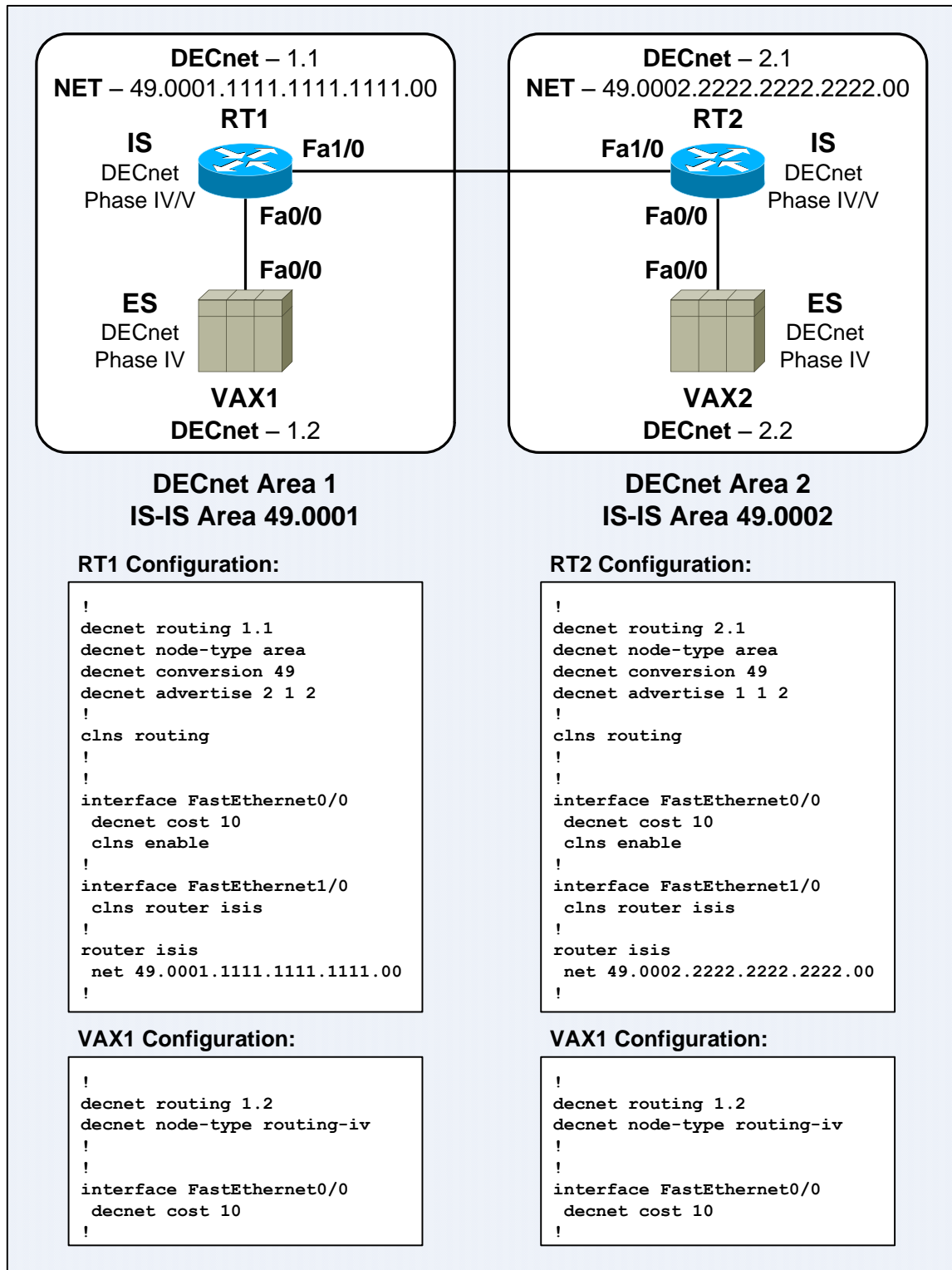
```
O 192.168.2.0/24 [110/2] via 192.168.1.4, 00:01:08, FastEthernet1/0
```

```
RT3#
```



- **Why does this problem occur?**  
When enabling multiple OSPF processes on a router, the processes are independent from the software perspective. When OSPF runs as in a single OSPF process, it always prefers internal routes over external routes. However, OSPF does not perform route selection between processes. For instance, OSPF metrics and route types are not being considered when deciding the route from which process should be installed into the routing table. There is no interaction between different OSPF processes, and the tie-breaker is administrative distance. Since both OSPF processes have a default AD of 110, the first process that installs the route into the routing table will succeed against another process.
- The 1st solution is define different AD values for routes from different OSPF processes, so that the routes of certain OSPF processes are preferred over the routes of other processes by human intention, and not a matter of chance or luck. By implementing the **distance ospf external 115** OSPF router subcommand under the OSPF process 2 on both routers, which increase the AD of the external routes in OSPF process 2, internal routes learnt via OSPF process 1 will always be preferred over external routes redistributed from OSPF 1 into OSPF 2.
- If the main reason of implementing the different OSPF processes is to filter certain routes, the **OSPF ABR Type-3 LSA Filtering** feature should be considered to archive the same result. Instead of implementing a second OSPF process, the routers and links reside in OSPF process 2 as in the sample network, could be configured as another area in the OSPF process 1, followed by implementing Type-3 LSA filtering on RT2 and RT3, the ABRs.

## DECnet Inter-Phase IV-Area Routing through DECnet Phase V Routing (IS-IS)



**Figure A6-35:** DECnet Inter-Phase IV-Area Routing through DECnet Phase V Routing (IS-IS)

- Below verifies the DECnet configuration on VAX1. VAX1 (1.2) is able to reach VAX2 (2.2):

```

VAX1#sh decnet
Global DECnet parameters for network 0:
  Local address is 1.2, node type is routing-iv
  Nearest Level-2 router is 1.1
Maximum node is 1023, maximum area is 63, maximum visits is 63
Maximum paths is 1, path split mode is normal
Local maximum cost is 1022, maximum hops is 30
Area maximum cost is 1022, maximum hops is 30
Static routes *NOT* being sent in routing updates
Cluster-alias routes *NOT* being sent in routing updates
VAX1#
VAX1#sh decnet neighbors
Net Node      Interface      MAC address    Flags
0 1.1         FastEthernet0/0 aa00.0400.0104 V
VAX1#
VAX1#sh decnet route
Node      Cost  Hops      Next Hop to Node      Expires  Prio
* (Area)  10   1         FastEthernet0/0 -> 1.1
*1.1     10   1         FastEthernet0/0 -> 1.1      34      64  V
*1.2     0    0         (Local) -> 1.2
VAX1#
VAX1#debug decnet packets
DECnet packet forwarding debugging is on
VAX1#
VAX1#ping 2.2

Type escape sequence to abort.
Sending 5, 100-byte DECnet echos to atg 0 area.node 2.2, timeout is 5
seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 88/136/252 ms
VAX1#
00:02:15: DNET-PKT: Packet fwded from 1.2 to 2.2, via 1.1, snpa
aa00.0400.0104, FastEthernet0/0
00:02:16: Source address: 2.2
00:02:16: DNET: echo response from 2.2

00:02:16: DNET-PKT: Packet fwded from 1.2 to 2.2, via 1.1, snpa
aa00.0400.0104, FastEthernet0/0
00:02:16: Source address: 2.2
00:02:16: DNET: echo response from 2.2

00:02:16: DNET-PKT: Packet fwded from 1.2 to 2.2, via 1.1, snpa
aa00.0400.0104, FastEthernet0/0
00:02:16: Source address: 2.2
00:02:16: DNET: echo response from 2.2

00:02:16: DNET-PKT: Packet fwded from 1.2 to 2.2, via 1.1, snpa
aa00.0400.0104, FastEthernet0/0
00:02:16: Source address: 2.2
00:02:16: DNET: echo response from 2.2

VAX1#

```

## Pure CLNS Routing Network

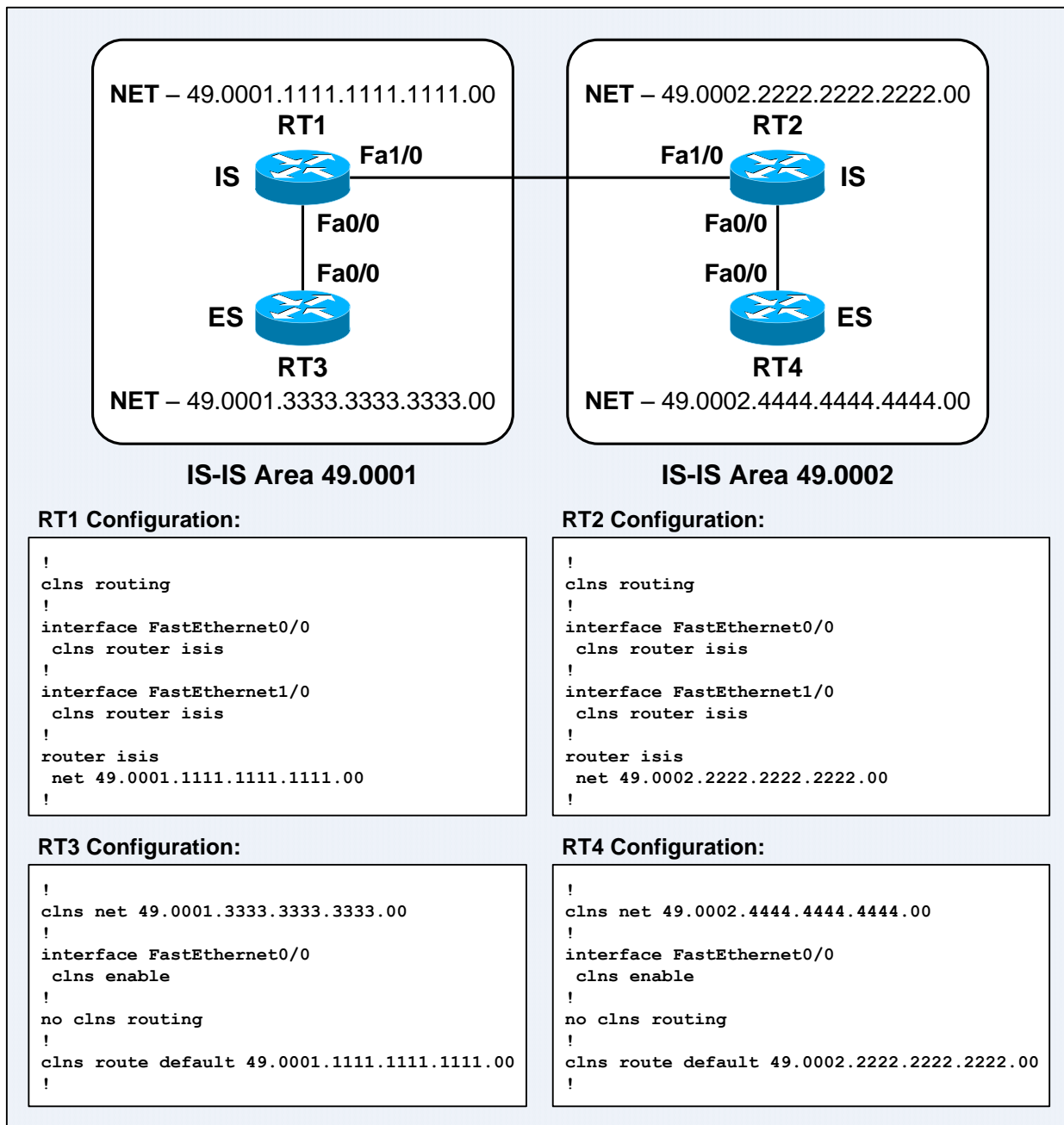


Figure A6-36: Network Setup for Pure CLNS Routing

- Below verifies the CLNS configuration on RT3. RT3 (49.0001.3333.3333.3333.00) is able to reach RT4 (49.0002.4444.4444.4444.00):

```

RT3#sh clns
Global CLNS Information:
  1 Interfaces Enabled for CLNS
  NET: 49.0001.3333.3333.3333.00
  Configuration Timer: 60, Default Holding Timer: 300, Packet Lifetime 64
  ERPDU's requested on locally generated packets
  End system operation only (CLNS forwarding not allowed)
RT3#
RT3#sh clns interface
FastEthernet0/0 is up, line protocol is up
  Checksums enabled, MTU 1497, Encapsulation SAP
  ERPDU's enabled, min. interval 10 msec.
  CLNS fast switching enabled
  CLNS SSE switching disabled
  DEC compatibility mode OFF for this interface
  Next ESH/ISH in 42 seconds
RT3#
RT3#sh clns neighbors

System Id      Interface      SNPA              State  Holdtime  Type
Protocol
1111.1111.1111 Fa0/0         cc00.0f38.0000    Up     279       IS    ES-IS
RT3#
RT3#sh clns route
Codes: C - connected, S - static, d - DecnetIV
       I - ISO-IGRP,  i - IS-IS,  e - ES-IS
       B - BGP,       b - eBGP-neighbor

C  49.0001.3333.3333.3333.00 [1/0], Local NET

S  Default Prefix [10/0]
via 49.0001.1111.1111.1111.00
RT3#
RT3#debug clns packets
CLNS packets debugging is on
RT3#
RT3#ping 49.0002.4444.4444.4444.00

Type escape sequence to abort.
Sending 5, 100-byte CLNS Echos with timeout 2 seconds
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 92/130/160 ms
RT3#
00:01:36: CLNS: Originating packet, size 100
00:01:36:          from 49.0001.3333.3333.3333.00
          to 49.0002.4444.4444.4444.00
          via 1111.1111.1111 (FastEthernet0/0 cc00.0f38.0000)
00:01:36: CLNS: Echo Reply PDU received on FastEthernet0/0!
00:01:36: CLNS: Originating packet, size 100
00:01:36:          from 49.0001.3333.3333.3333.00
          to 49.0002.4444.4444.4444.00
          via 1111.1111.1111 (FastEthernet0/0 cc00.0f38.0000)
00:01:36: CLNS: Echo Reply PDU received on FastEthernet0/0!
--- output omitted ---

```

- The **which-route** {*nsap-addr* | *clns-name*} privileged command displays the OSI routing table in which the specified CLNS destination is found. This command determines the next-hop router and is important when troubleshooting configuration with multiple processes running.

```

RT1#which-route 49.0001.1111.1111.1111.00
Route look-up for destination 49.0001.1111.1111.1111.00
  Found route in IS-IS level-1 routing table - destination is local
RT1#
RT1#which-route 49.0001.2222.2222.2222.00
Route look-up for destination 49.0001.2222.2222.2222.00
  Route not found
RT1#
RT1#which-route 49.0001.3333.3333.3333.00
Route look-up for destination 49.0001.3333.3333.3333.00
  Found route in IS-IS level-1 routing table

Adjacency entry used:
System Id      Interface      SNPA              State  Holdtime  Type
Protocol
3333.3333.3333 Fa0/0          cc02.03fc.0000    Up     277       IS    ES-IS
  Area Address(es): 49.0001
  Uptime: 00:01:59
RT1#
RT1#which-route 49.0002.4444.4444.4444.00
Route look-up for destination 49.0002.4444.4444.4444.00
  Found route in CLNS L2 prefix routing table

Route entry used:
i 49.0002 [110/10]
  via RT2, FastEthernet1/0

Adjacency entry used:
System Id      Interface      SNPA              State  Holdtime  Type
Protocol
RT2            Fa1/0          cc01.03fc.0010    Up     8          L2    IS-IS
  Area Address(es): 49.0002
  Uptime: 00:02:16
  NSF capable
RT1#which-route 49.0002
Route look-up for destination 49.0002
  Found route in CLNS L2 prefix routing table

Route entry used:
i 49.0002 [110/10]
  via RT2, FastEthernet1/0

Adjacency entry used:
System Id      Interface      SNPA              State  Holdtime  Type
Protocol
RT2            Fa1/0          cc01.03fc.0010    Up     9          L2    IS-IS
  Area Address(es): 49.0002
  Uptime: 00:02:27
  NSF capable
RT1#

```

- Route information can reside in the following table:
  - IS-IS Level 1 routing table
  - ISO IGRP System ID or area routing table
  - Prefix routing table (IS-IS Level 2 routes, ISO IGRP domain routes, and static routes)
  - Adjacency table

## Why IS-IS?

- Below lists some good reasons to prefer IS-IS over the better-known OSPF as an interior gateway protocol (IGP):
  - **Better flooding scheme on broadcast media.** OSPF routers must first multicast LSUs to the DR and BDR; while IS-IS routers directly multicast LSPs to all ISs on the broadcast network – routers can receive link-state updates directly from the originator rather than waiting until the DR gets it and resends it. The OSPF DROther1 → DR → DROther2 link-state update flow also unnecessary consumes network bandwidth due to the duplicate LSUs.
  - **Reduced LSP traffic in unstable networks.** OSPF DR uses the acknowledge–retransmit scheme to reliably deliver LSUs to routers on a broadcast network. IS-IS DIS periodically multicasts a CSNP that contains the complete summary of its LSDB. When an IS notices that it is missing an LSP after compared the contents of the CSNP against its LSDB, it will request the missing LSP from the DIS using PSNP. The OSPF approach generates a lot of extra traffic when a network is unstable due to the high volumes of LSUs and LSAs; while the IS-IS approach generates some extra traffic all the time due to the periodic CSNPs. It is preferably to have a little extra traffic when the network is stable rather than a lot of extra traffic when the network is unstable.
  - **Link-state basis.** OSPF is only a link-state routing protocol within an area; it uses a distance-vector approach when computing inter-area routes. IS-IS maintains separate LSDBs for the L1 and L2 topologies and utilizes SPF algorithm to compute routes for both routing levels.
  - **Hierarchical flexibility.** OSPF requires extra design effort due to the special *backbone area* in which all ABRs must be connected to it in order to participate in inter-area routing. IS-IS doesn't have this built-in design limitation; it employs a more general concept of L1 and L2 routing levels.
  - **Simplicity.** OSPF has more than 10 LSA types; while IS-IS has only 2 for each routing level – node and pseudonode. OSPF has virtual links; IS-IS doesn't need them as any router can participate in L2 routing by simply forming an L2 adjacency with another L2 router.
  - **Robust failure mode.** IS-IS LSPs employ an Overload bit that is used to inform other routers don't route traffic to the originating router when the bit is set. IS-IS uses this bit to prevent routing failures due to forwarding packets to a router that experiencing problems. The Overload bit can be used in conjunction with BGP. The bit is set while BGP is converging in order to prevent black-holing of transit traffic. Ex: When an IS-IS router is an Internet backbone router and is reloaded, the IGP is able to converge very fast but BGP will take some time. Without a mechanism like the Overload bit, a lot of packets may be sent toward the router after the IGP has converged, but the packets will be black-holed if they are destined to networks that haven't been received through BGP updates.
  - **Better SPF calculation and LSP generation back-off strategy.** IS-IS employs exponential back-off to control SPF calculation and LSP generation. OSPF does not have such capability for long time until the LSA Throttling and LSA Group Pacing features were introduced.
  - **LSP aging.** OSPF LSAs count up to MaxAge and are then refreshed; IS-IS LSPs count down based on the Remaining Lifetime which has the advantage that the aging time can differ per LSP as it is being set by the originator. In OSPF, undesirable network spikes happen when all LSAs are expiring and getting refreshed at the same time. The OSPF LSA Group Pacing feature artificially spaces out LSA reflooding.
  - **Better adjacency-forming policy.** An OSPF router refuses to form adjacency if any of the numerous parameters in a Hello packet received from a neighbor doesn't match with its own. IS-IS does not employ such dumb policy.
- Besides that, failure of the DR in OSPF is more serious than failure of the DIS in IS-IS because the OSPF DR keeps track of many states about this DROther has which LSAs. The IS-IS DIS doesn't maintain such states therefore the recovery is faster than the failure of OSPF DR.

## The Impact of the clear ip route Privileged Command

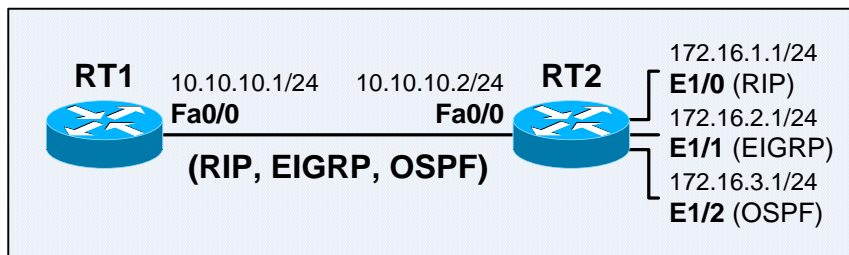


Figure A6-37: Network Setup for the clear ip route Privileged Command

- Below shows the routing table on RT1:

```
RT1#sh ip route

Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 3 subnets
R       172.16.1.0 [120/1] via 10.10.10.2, 00:00:16, FastEthernet0/0
D       172.16.2.0 [90/284160] via 10.10.10.2, 00:02:06, FastEthernet0/0
O       172.16.3.0 [110/11] via 10.10.10.2, 00:01:14, FastEthernet0/0
    10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0
RT1#
RT1#sh ip route summary
IP routing table name is Default-IP-Routing-Table(0)
Route Source      Networks      Subnets      Overhead      Memory (bytes)
connected         0             1             64            152
static            0             0             0             0
eigrp 100         0             1             64            152
ospf 100          0             1             64            152
  Intra-area: 1 Inter-area: 0 External-1: 0 External-2: 0
  NSSA External-1: 0 NSSA External-2: 0
rip               0             1             64            152
internal          2             2             0             2344
Total             2             4             256           2952
RT1#
```

- Below shows the impact of clearing the RIPv2 route – 172.16.1.0/24. Packet drops were observed.

```
RT1#debug ip rip
RIP protocol debugging is on
RT1#
RT1#debug ip routing
IP routing debugging is on
RT1#
RT1#sh ip route

Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 3 subnets
R       172.16.1.0 [120/1] via 10.10.10.2, 00:00:02, FastEthernet0/0
D       172.16.2.0 [90/284160] via 10.10.10.2, 00:02:51, FastEthernet0/0
O       172.16.3.0 [110/11] via 10.10.10.2, 00:01:59, FastEthernet0/0
    10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0
RT1#
```





- Below shows the impact of clearing the EIGRP route – 172.16.2.0/24. No packet drop was observed.

```

RT1#debug ip eigrp
IP-EIGRP Route Events debugging is on
RT1#
RT1#debug ip routing
IP routing debugging is on
RT1#
RT1#sh ip route

Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 3 subnets
R       172.16.1.0 [120/1] via 10.10.10.2, 00:00:24, FastEthernet0/0
D       172.16.2.0 [90/284160] via 10.10.10.2, 00:05:03, FastEthernet0/0
O       172.16.3.0 [110/11] via 10.10.10.2, 00:04:11, FastEthernet0/0
    10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0
RT1#
RT1#sh ip eigrp neighbors
IP-EIGRP neighbors for process 100
H   Address                Interface           Hold Uptime    SRTT   RTO   Q   Seq
                               (sec)              (ms)          Cnt Num
0   10.10.10.2              Fa0/0              12 00:05:06    20    200  0   1
RT1#
RT1#sh ip eigrp topology
IP-EIGRP Topology Table for AS(100)/ID(10.10.10.1)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.10.10.0/24, 1 successors, FD is 28160
     via Connected, FastEthernet0/0
P 172.16.2.0/24, 1 successors, FD is 284160
     via 10.10.10.2 (284160/281600), FastEthernet0/0
RT1#
RT1#clear ip route 172.16.2.0 255.255.255.0
RT1#
RT1#sh ip route

Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 3 subnets
R       172.16.1.0 [120/1] via 10.10.10.2, 00:00:14, FastEthernet0/0
D       172.16.2.0 [90/284160] via 10.10.10.2, 00:00:00, FastEthernet0/0
O       172.16.3.0 [110/11] via 10.10.10.2, 00:04:28, FastEthernet0/0
    10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0
RT1#
RT1#
00:05:27: RT: del 172.16.2.0/24 via 10.10.10.2, eigrp metric [90/284160]
00:05:27: RT: delete subnet route to 172.16.2.0/24
00:05:27: RT: NET-RED 172.16.2.0/24
00:05:27: RT: add 172.16.2.0/24 via 10.10.10.2, eigrp metric [90/284160]
00:05:27: RT: NET-RED 172.16.2.0/24
RT1#

```





## Using Extended Access Lists with Route Maps and Distribute Lists

- Standard access lists are often being used in conjunction with distribute lists for route filtering. Using extended access lists with distribute lists is supported; however, the syntax is quite confusing because they behave differently when being referred by route maps and distribute lists.
- When using an extended access list for a route map, it acts like a prefix list, which means that it can match both the address and subnet masks of IP prefixes. IP prefixes of 10.0.0.0/8 and 10.0.0.0/16 can be differentiated by defining that the address must be 10.0.0.0 and the subnet mask must be /8. The syntax of prefix list to achieve this is very straightforward as below:

```
ip prefix-list prefix-test01 permit 10.0.0.0/8
```

- The extended access lists to achieve the same result as the prefix list above are as below. Note that the extended access lists are no longer matching any source and destination address pairs, but instead matching the address and subnet mask – this means that the address must be exactly 10.0.0.0 and the subnet mask must be exactly 255.0.0.0.

```
access-list 101 permit ip host 10.0.0.0 host 255.0.0.0
                        --- OR ---
access-list 101 permit ip 10.0.0.0 0.255.255.255 host 255.0.0.0
```

**Note:** The standard access list to achieve the similar result is:

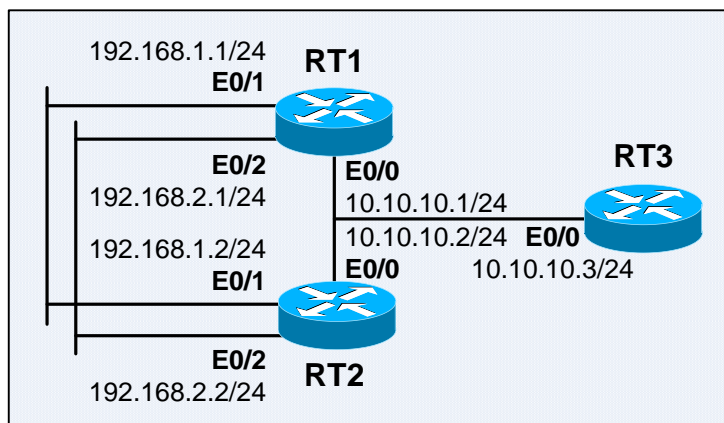
```
access-list 1 permit 10.0.0.0 0.255.255.255
```

- The extended access list above can perform fuzzy binary matches by changing the **host** keyword to a wildcard mask. The configuration below matches any address that starts with 192.168. and has a subnet mask of /24, which matches 192.168.0.0/24, 192.168.100.0/24, etc.

```
access-list 102 permit ip 192.168.0.0 0.0.255.255 host 255.255.255.0
```

The source address part of the access list matches the destination address of the route; while the destination address part of the access list – the network mask in fact, indicates the range.

- The confusion lies upon when extended access lists are being referred by distribute lists. The source and destination fields in the extended ACL syntax match the update source of the route and the network address respectively. This provides the mechanism to control which networks are being received, and more importantly from whom the networks were received from.



**Figure A6-38:** Network Setup for Extended Access Lists with Distribute Lists

- Below shows that RT3 learned the 2 prefixes twice, once from RT1 and once from RT2.

```

RT3#sh ip route rip
R    192.168.1.0/24 [120/1] via 10.10.10.1, 00:00:02, Ethernet0/0
                                [120/1] via 10.10.10.2, 00:00:20, Ethernet0/0
R    192.168.2.0/24 [120/1] via 10.10.10.1, 00:00:02, Ethernet0/0
                                [120/1] via 10.10.10.2, 00:00:20, Ethernet0/0
RT3#

```

- Below implement an extended access list along with the distribute list on RT3 to only receive the 192.168.1.0/24 and 192.168.2.0/24 prefixes from RT1 and RT2 respectively.

```

RT3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT3(config)#access-list 101 permit ip host 10.10.10.1 host 192.168.1.0
RT3(config)#access-list 101 permit ip host 10.10.10.2 host 192.168.2.0
RT3(config)#
RT3(config)#router rip
RT3(config-router)#distribute-list 101 in Ethernet0/0
RT3(config-router)#end
RT3#
RT3#clear ip route *
RT3#sh ip route rip
R    192.168.1.0/24 [120/1] via 10.10.10.1, 00:00:01, Ethernet0/0
R    192.168.2.0/24 [120/1] via 10.10.10.2, 00:00:02, Ethernet0/0
RT3#

```

- Now that RT3 has only one entry for each prefix, with 192.168.1.0/24 coming only from RT1 and 192.168.2.0/24 coming only from RT2. The disadvantage on this configuration is that it is unable to differentiate prefixes based on their subnet masks, eg: unable to control to receive only 172.16.0.0/16 from RT1 and 172.16.0.0/24 from RT2. Prefix list along with distribute list should be implemented using the **distribute-list prefix {ip-prefix-name}** router subcommand to achieve such requirement.

- Below shows some other examples of prefix lists and their equivalent extended access lists.

**Ex #1:** Matches only the 172.16.10.0/24 prefix.

- ip prefix-list prefix-test01 permit 172.16.10.0/24
- access-list 101 permit host 172.16.0.0 host 255.255.255.0

**Ex #2:** Matches subnets from 172.16.0.0/16; the entire class B range.

- ip prefix-list prefix-test01 permit 172.16.0.0/16 le 32
- access-list 101 permit ip 172.16.0.0 0.0.255.255 255.255.0.0 0.0.255.255

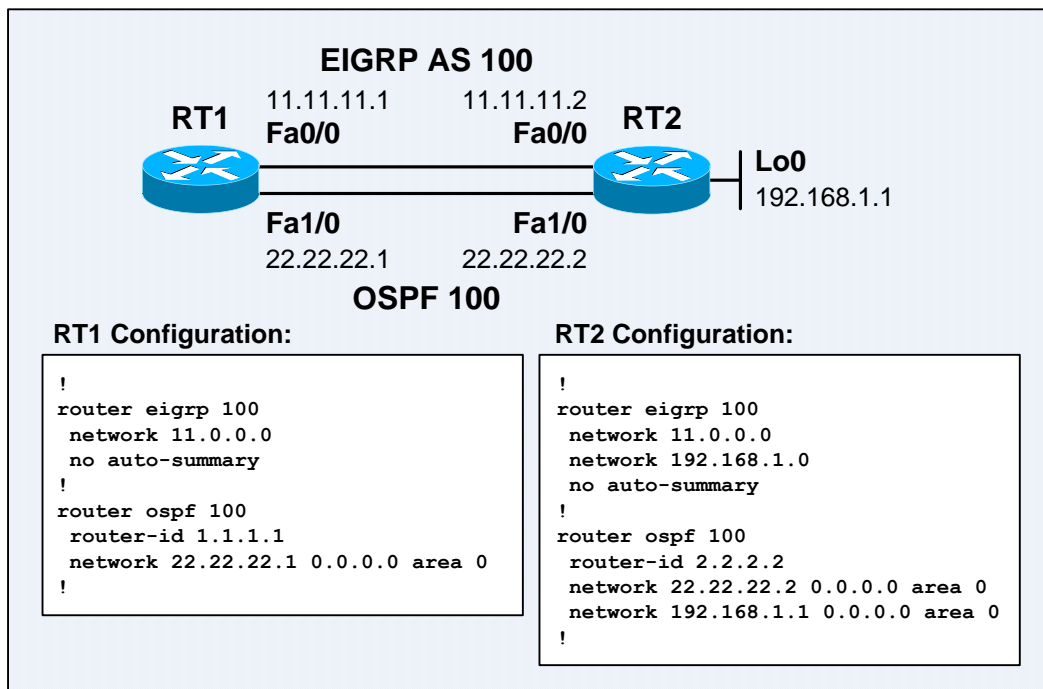
**Ex #3:** Matches subnets from 172.16.0.0/16 with a prefix length between /24 and /32.

- ip prefix-list prefix-test01 permit 172.16.0.0/16 ge 24
- access-list 101 permit ip 172.16.0.0 0.0.255.255 255.255.255.0 0.0.0.255

**Ex #4:** Matches subnets from 172.16.0.0/16 with a prefix length between /24 and /30.

- ip prefix-list prefix-test01 permit 172.16.0.0/16 ge 24 le 30
  - access-list 101 permit ip 172.16.0.0 0.0.255.255 255.255.255.0 0.0.0.252
- (NOT) access-list 101 permit ip 172.16.0.0 0.0.255.255 **255.255.255.3 0.0.0.252**

## Administrative Distance Load Balancing



**Figure A6-39: Administrative Distance Load Balancing**

- The network setup above is being setup to answer an interesting question. EIGRP is enabled for the upper path while OSPF is enabled for the lower path. RT2 Lo0 is enabled for both EIGRP and OSPF.
- **Q:** The administrative distance values for EIGRP and OSPF are 90 and 120 respectively. If the administrative distance of OSPF is modified to be the same as EIGRP – 90 on RT1, will RT1 insert both EIGRP and OSPF routes to 192.168.1.0/24 into its IP routing table?
- **A:** No. If there are multiple routes with the same longest-match and same administrative distance but are learned from different routing protocols. Cisco IOS will consider the default administrative distance of the routing protocols and select the route with the lower default administrative distance.
- A common question related to this topic is are we able to load balance across 2 different MPLS providers in which we run EBGP with the 1st provider and run OSPF with the 2nd provider? 😊
- Let the battle begins by first checking the routing table on RT1:

```
RT1#sh ip route
```

```
Gateway of last resort is not set
```

```

    22.0.0.0/30 is subnetted, 1 subnets
C       22.22.22.0 is directly connected, FastEthernet1/0
    11.0.0.0/30 is subnetted, 1 subnets
C       11.11.11.0 is directly connected, FastEthernet0/0
```

```
D 192.168.1.0/24 [90/156160] via 11.11.11.2, 00:01:30, FastEthernet0/0
```

```
RT1#
```

- Let the battle begins by first checking the routing table on RT1:

```
RT1#debug ip routing
IP routing debugging is on
RT1#
RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#router ospf 100
RT1(config-router)#distance 89
RT1(config-router)#
00:02:25: RT: closer admin distance for 192.168.1.0, flushing 1 routes
00:02:25: RT: NET-RED 192.168.1.0/24
00:02:25: RT: add 192.168.1.0/24 via 22.22.22.2, ospf metric [89/2]
00:02:25: RT: NET-RED 192.168.1.0/24
RT1(config-router)#
! OSPF wins!!! :-)
RT1(config-router)#do sh ip route

Gateway of last resort is not set

    22.0.0.0/30 is subnetted, 1 subnets
C       22.22.22.0 is directly connected, FastEthernet1/0
    11.0.0.0/30 is subnetted, 1 subnets
C       11.11.11.0 is directly connected, FastEthernet0/0
O 192.168.1.0/24 [89/2] via 22.22.22.2, 00:00:09, FastEthernet1/0
RT1(config-router)#
RT1(config-router)#distance 90
RT1(config-router)#
00:02:39: RT: delete route to 192.168.1.0/24
00:02:39: RT: NET-RED 192.168.1.0/24
00:02:39: RT: add 192.168.1.0/24 via 11.11.11.2, eigrp metric [90/156160]
00:02:39: RT: NET-RED 192.168.1.0/24
RT1(config-router)#
! EIGRP wins again!!! :-)
RT1(config-router)#do sh ip route

Gateway of last resort is not set

    22.0.0.0/30 is subnetted, 1 subnets
C       22.22.22.0 is directly connected, FastEthernet1/0
    11.0.0.0/30 is subnetted, 1 subnets
C       11.11.11.0 is directly connected, FastEthernet0/0
D 192.168.1.0/24 [90/156160] via 11.11.11.2, 00:00:05, FastEthernet0/0
RT1(config-router)#
RT1(config-router)#exit
! Static routing enters the battle!!! :-)
RT1(config)#ip route 192.168.1.0 255.255.255.0 22.22.22.2 90
RT1(config)#
00:03:07: RT: closer admin distance for 192.168.1.0, flushing 1 routes
00:03:07: RT: NET-RED 192.168.1.0/24
00:03:07: RT: add 192.168.1.0/24 via 22.22.22.2, static metric [90/0]
00:03:07: RT: NET-RED 192.168.1.0/24
RT1(config)#
! Here comes the final winner!!! :-)
RT1(config)#do sh ip route

Gateway of last resort is not set

    22.0.0.0/30 is subnetted, 1 subnets
C       22.22.22.0 is directly connected, FastEthernet1/0
    11.0.0.0/30 is subnetted, 1 subnets
C       11.11.11.0 is directly connected, FastEthernet0/0
S 192.168.1.0/24 [90/0] via 22.22.22.2
RT1(config)#
```



## The ip route-cache policy Interface Subcommand

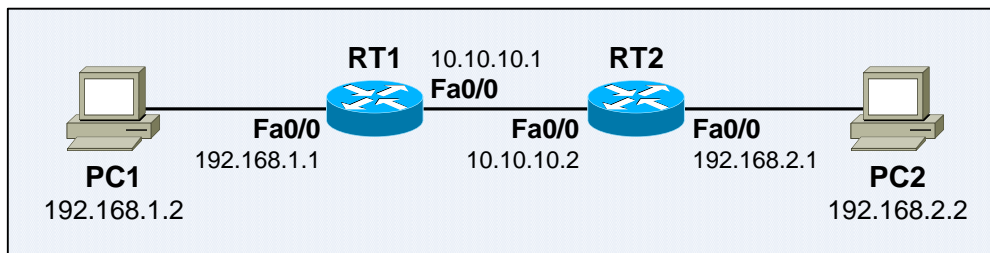


Figure A6-40: Network Setup for the **ip route-cache policy** Interface Subcommand

- The **ip route-cache policy** interface subcommand enables **Fast-Switched Policy-Based Routing (FSPBR)** for an interface. PBR must first be configured before enabling FSPBR. FSPBR is disabled by default. However, this command is not required if Cisco Express Forwarding (CEF) is already enabled, as PBR packets are CEF-switched by default.
- FSPBR supports all the **match** commands and most of the **set** actions. Below lists the restrictions:
  - The **set ip default next-hop** action is not supported.
  - The **set interface** action is supported only over point-to-point links, unless a route cache entry exists for the interface specified in the **set interface** action in the route map. The route cache is the portion of router memory assigned for faster routing decisions. Additionally, the routing table is consulted to determine the path to a destination upon process switching; while Cisco IOS does not make this check during fast switching because fast switching utilizes route cache for routing decisions – Cisco IOS blindly forwards the packet out from the specified interface. Issue the **show ip cache policy EXEC** command to display the cache entries in the policy route cache.
- PC1 (192.168.1.2) was performing continuous ping to PC2 (192.168.2.2) during the capture for the usage of the **ip route-cache policy** interface subcommand. Below shows that normal packets were being CEF-switched by default.

```

RT1#sh int stats
FastEthernet0/0
  Switching path  Pkts In  Chars In  Pkts Out  Chars Out
  Processor       8        2029     22        2833
  Route cache  149    16986  146    16644
  Total          157      19015   168      19477
FastEthernet1/0
  Switching path  Pkts In  Chars In  Pkts Out  Chars Out
  Processor       8        1993     22        2833
  Route cache  146    16644  149    16986
  Total          154      18637   171      19819
RT1#
RT1#sh int stats
FastEthernet0/0
  Switching path  Pkts In  Chars In  Pkts Out  Chars Out
  Processor       8        2029     23        2893
  Route cache  234    26676  231    26334
  Total          242      28705   254      29227
FastEthernet1/0
  Switching path  Pkts In  Chars In  Pkts Out  Chars Out
  Processor       8        1993     23        2893
  Route cache  231    26334  234    26676
  Total          239      28327   257      29569
RT1#
  
```

- Below configures a PBR and applies it to RT1 Fa0/0.

```

RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#access-list 101 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
RT1(config)#route-map Test_PBR
RT1(config-route-map)#match ip address 101
RT1(config-route-map)#set ip next-hop 10.10.10.2
RT1(config-route-map)#do sh route-map
route-map Test_PBR, permit, sequence 10
  Match clauses:
    ip address (access-lists): 101
  Set clauses:
    ip next-hop 10.10.10.2
  Policy routing matches: 0 packets, 0 bytes
RT1(config-route-map)#
RT1(config-route-map)#int fa0/0
RT1(config-if)#ip policy route-map Test_PBR
! Below shows that the PBR is in effect.
RT1(config-if)#do sh route-map
route-map Test_PBR, permit, sequence 10
  Match clauses:
    ip address (access-lists): 101
  Set clauses:
    ip next-hop 10.10.10.2
Policy routing matches: 28 packets, 3192 bytes
RT1(config-if)#
RT1(config-if)#do sh route-map
route-map Test_PBR, permit, sequence 10
  Match clauses:
    ip address (access-lists): 101
  Set clauses:
    ip next-hop 10.10.10.2
Policy routing matches: 50 packets, 5700 bytes
RT1(config-if)#
! Below shows that the PBR packets were being CEF-switched.
RT1(config-if)#do sh int stats
FastEthernet0/0
      Switching path   Pkts In   Chars In   Pkts Out   Chars Out
      Processor         10        2747         35         4229
      Route cache      1137      129618      1134      129276
      Total           1147      132365      1169      133505
FastEthernet1/0
      Switching path   Pkts In   Chars In   Pkts Out   Chars Out
      Processor         10        2729         35         4229
      Route cache      1134      129276      1137      129618
      Total           1144      132005      1172      133847
RT1(config-if)#
RT1(config-if)#do sh int stats
FastEthernet0/0
      Switching path   Pkts In   Chars In   Pkts Out   Chars Out
      Processor         10        2747         35         4229
      Route cache      1199      136686      1197      136458
      Total           1209      139433      1232      140687
FastEthernet1/0
      Switching path   Pkts In   Chars In   Pkts Out   Chars Out
      Processor         10        2729         35         4229
      Route cache      1197      136458      1199      136686
      Total           1207      139187      1234      140915
RT1(config-if)#

```

- Below shows that the PBR packets were being process-switched after disabled CEF on RT1 Fa0/0.

```

RT1(config-if)#no ip route-cache cef
RT1(config-if)#do sh int stats
FastEthernet0/0
  Switching path  Pkts In  Chars In  Pkts Out  Chars Out
    Processor      108      13919     40         4583
  Route cache     1404     160056    1498       170772
  Total           1512     173975    1538       175355
FastEthernet1/0
  Switching path  Pkts In  Chars In  Pkts Out  Chars Out
    Processor      11       2843     137        15641
  Route cache     1498     170772    1404       160056
  Total           1509     173615    1541       175697
RT1(config-if)#
RT1(config-if)#do sh int stats
FastEthernet0/0
  Switching path  Pkts In  Chars In  Pkts Out  Chars Out
    Processor      182      22355     41         4643
  Route cache     1404     160056    1572       179208
  Total           1586     182411    1613       183851
FastEthernet1/0
  Switching path  Pkts In  Chars In  Pkts Out  Chars Out
    Processor      11       2843     212        24137
  Route cache     1572     179208    1404       160056
  Total           1583     182051    1616       184193
RT1(config-if)#

```

- Below configures the **Fast Switching for Policy-Based Routing (FSPBR)** on RT1 Fa0/0.

```

RT1(config-if)#do sh ip cache policy
Total adds 0, total deletes 0

Type Routemap/sequence      Age      Interface      Next Hop
RT1(config-if)#
RT1(config-if)#ip route-cache policy
RT1(config-if)#do sh ip cache policy
Total adds 1, total deletes 0

Type Routemap/sequence      Age      Interface      Next Hop
NH Test_PBR/10           00:00:05  FastEthernet1/0  10.10.10.2
RT1(config-if)#

```

- Below shows that the PBR packets were being Fast-switched after configured the **Fast-Switched Policy-Based Routing (FSPBR)** on RT1 Fa0/0.

```

RT1(config-if)#do sh int stats
FastEthernet0/0
    Switching path  Pkts In   Chars In   Pkts Out   Chars Out
    Processor       373      44374     45         5191
    Route cache    1481     168834    1839      209646
    Total           1854     213208    1884      214837
FastEthernet1/0
    Switching path  Pkts In   Chars In   Pkts Out   Chars Out
    Processor       12       3211     406        46345
    Route cache     1839     209646    1481       168834
    Total           1851     212857    1887       215179
RT1(config-if)#
RT1(config-if)#do sh int stats
FastEthernet0/0
    Switching path  Pkts In   Chars In   Pkts Out   Chars Out
    Processor       373      44374     45         5191
    Route cache    1514     172596    1873      213522
    Total           1887     216970    1918      218713
FastEthernet1/0
    Switching path  Pkts In   Chars In   Pkts Out   Chars Out
    Processor       12       3211     406        46345
    Route cache     1873     213522    1514       172596
    Total           1885     216733    1920       218941
RT1(config-if)#

```

## The BGP Notification Message Error Codes and Error Subcodes

Error Code	Error	Error Subcode	Subcode Detail
1	Message Header Error	1	Connection Not Synchronized
		2	Bad Message Length
		3	Bad Message Type
2	Open Message Error	1	Unsupported Version Number
		2	Bad Peer AS
		3	Bad BGP Identifier
		4	Unsupported Optional Parameter
		5	Authentication Failure (deprecated)
		6	Unacceptable Hold Time
		7	Unsupported Capability
3	Update Message Error	1	Malformed Attribute List
		2	Unrecognized Well-known Attribute
		3	Missing well-known attribute
		4	Attribute Flags Error
		5	Attribute Length Error
		6	Invalid ORIGIN Attribute
		7	AS routing loop (deprecated)
		8	Invalid NEXT_HOP Attribute
		9	Optional Attribute Error
		10	Invalid Network Field
		11	Malformed AS_PATH
4	Hold Timer Expired	0	–
5	Finite State Machine Error	0	–
6	Cease (for fatal errors besides the ones already listed)	1	Maximum Number of Prefixes Reached
		2	Administrative Shutdown
		3	Peer De-configured
		4	Administrative Reset
		5	Connection Rejected
		6	Other Configuration Change
		7	Connection Collision Resolution
		8	Out of Resources

### References:

<http://www.iana.org/assignments/bgp-parameters/bgp-parameters.xhtml>

## Cracking the BGP ORIGIN Codes

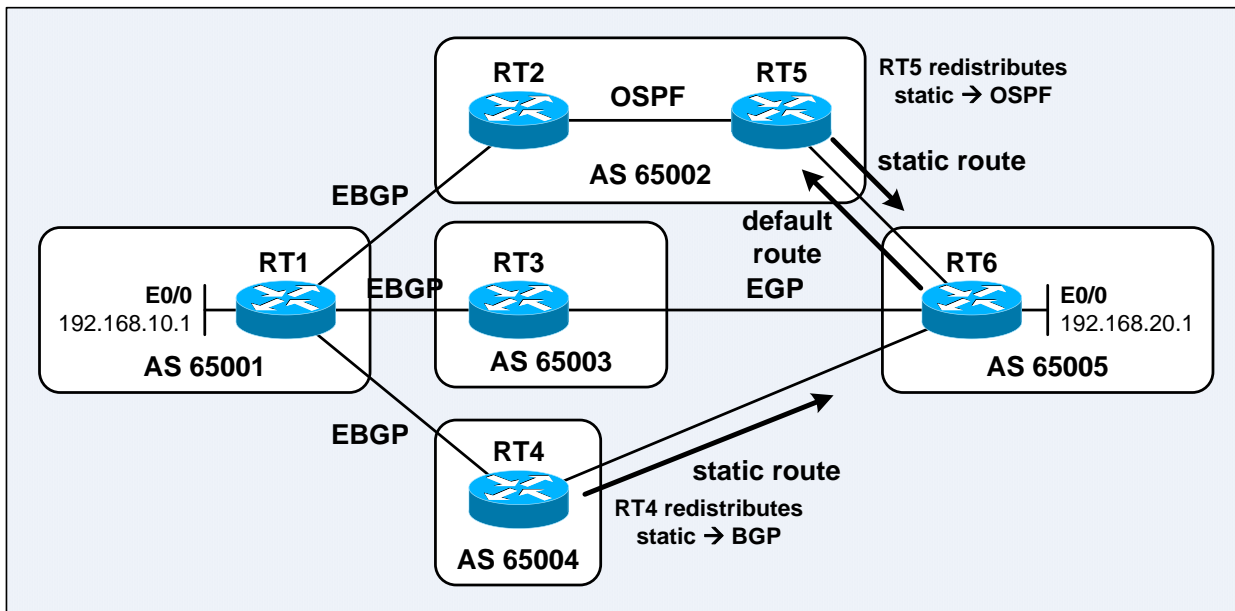


Figure A6-41: Network Setup for BGP ORIGIN Attribute

- The network above is setup to investigate how BGP generates the BGP ORIGIN codes. RT2 advertises the route to 192.168.20.0/24 to RT1 with an origin of IGP, as the BGP route is originated within the AS – RT5 redistributes the static route into OSPF and advertises it to RT2. RT3 advertises the route to 192.168.20.0/24 to RT1 with an origin of EGP, as the EGP route learned from AS 65005 is redistributed into BGP. RT4 advertises the route to 192.168.20.0/24 to RT1 with an origin of incomplete, as the static route is actually being redistributed into BGP.

```
RT1#sh ip bgp summary
```

```
--- output omitted ---
```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
2.2.2.2	4	65002	9	9	3	0	0	00:04:10	1
3.3.3.3	4	65003	9	10	3	0	0	00:04:02	1
4.4.4.4	4	65004	9	10	3	0	0	00:04:07	1

```
RT1#
```

```
RT1#sh ip route 192.168.0.0 255.255.0.0 longer-prefixes
```

```
Gateway of last resort is not set
```

```
C 192.168.10.0/24 is directly connected, Ethernet0/0
```

```
B 192.168.20.0/24 [20/20] via 2.2.2.2, 00:04:00
```

```
RT1#
```

```
RT1#sh ip bgp
```

```
BGP table version is 3, local router ID is 1.1.1.1
```

```
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,  
r RIB-failure, S Stale
```

```
Origin codes: i - IGP, e - EGP, ? - incomplete
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 192.168.10.0	0.0.0.0	0		32768	i
* 192.168.20.0	3.3.3.3	1		0	65003 65005 e
*	4.4.4.4	0		0	65004 ?
*>	2.2.2.2	20		0	65002 i

```
RT1#
```

- RT3 managed to advertise the EGP received from RT6 to RT1 with the BGP ORIGIN code of **e**.

```

RT3#sh ip route 192.168.0.0 255.255.0.0 longer-prefixes

Gateway of last resort is not set

B    192.168.10.0/24 [200/0] via 1.1.1.1, 00:04:39
E    192.168.20.0/24 [140/1] via 36.36.36.6, 00:00:39, Ethernet0/1
RT3#
RT3#sh ip egp
Local autonomous system is 65003

  EGP Neighbor      FAS/LAS  State   SndSeq RcvSeq Hello  Poll j/k Flags
*36.36.36.6        65005/65003 UP       0       1       0    60   180   3 Perm, Act
RT3#
RT3#sh ip bgp
BGP table version is 4, local router ID is 3.3.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 192.168.10.0     1.1.1.1           0         0 65001 i
*> 192.168.20.0     36.36.36.6        1        32768 65005 e
*                   1.1.1.1           0         0 65001 65002 i
RT3#
RT3#sh ip bgp neighbors 1.1.1.1 advertised-routes
BGP table version is 4, local router ID is 3.3.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 192.168.20.0     36.36.36.6        1        32768 65005 e
RT3#

```

- Below verifies that the network is operational through an extended ping on RT1.

```

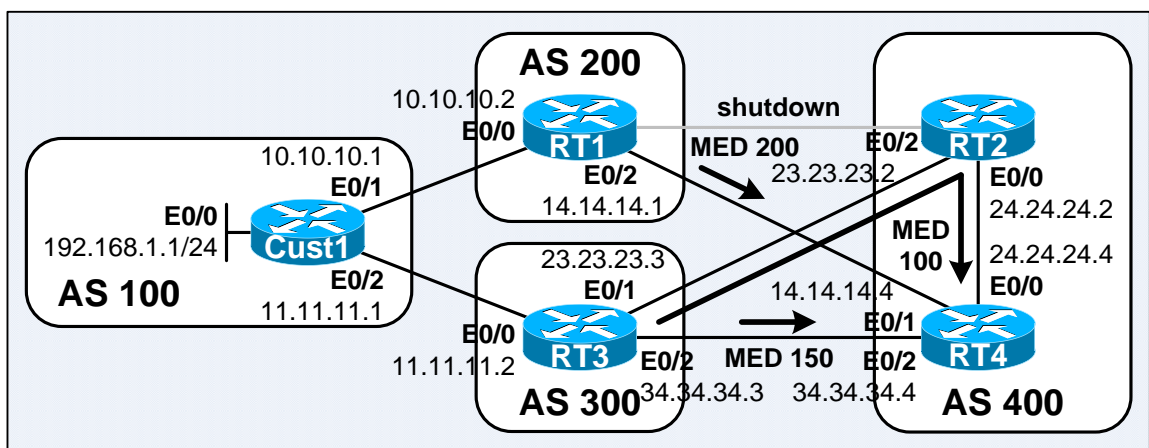
RT1#ping 192.168.20.1 source 192.168.10.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.20.1, timeout is 2 seconds:
Packet sent with a source address of 192.168.10.1
!!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 88/111/144 ms
RT1#

```

## The **bgp deterministic-med** and **bgp always-compare-med** Commands

- Below explains how the **bgp deterministic-med** and **bgp always-compare-med** BGP router subcommands affect and influence the MED-based BGP best path selection process. Both commands are not enabled by default; and both commands are separate and independent – enabling one does not automatically enable the other.
- The **bgp deterministic-med** command ensures the comparison of the MED attribute when choosing the paths advertised by different peers in the **same AS**.  
The **bgp always-compare-med** command ensures the comparison of the MED attribute when choosing the paths advertised by different peers in **different ASes**.
- The **bgp always-compare-med** command is useful when multiple ISPs or enterprises agree upon a uniform policy for setting the MED, eg: ISP1 sets the MED for Network X to 100; ISP2 sets the MED for Network X to 200. Both ISPs agree that ISP1 has the better path to Net X.



**Figure A6-42:** Network Setup for MED-based Path Manipulation

- Consider the following BGP routes for network 192.168.1.0/24:
  - **Entry #1** – AS\_PATH 300 100, MED 150, external, NEXT\_HOP 3.3.3.3, RID 3.3.3.3
  - **Entry #2** – AS\_PATH 200 100, MED 200, external, NEXT\_HOP 1.1.1.1, RID 1.1.1.1
  - **Entry #3** – AS\_PATH 300 100, MED 100, internal, NEXT\_HOP 2.2.2.2, RID 2.2.2.2
 The order in which the BGP routes were received is Entry #3, Entry #2, and Entry #1. Entry #3 and Entry #1 are the oldest and newest entries in the BGP routing table respectively.
- When the BGP process receives multiple routes to a particular destination, it lists them in the reverse order that they were received – from the newest to the oldest. The BGP process then compares the routes in pairs, starting with the newest entry and moving towards the oldest entry (starting at the top of the list and moving down) – Entry #1 and Entry #2 are compared first, the better of them is then compared to Entry #3, and so on...



- **Scenario #1: Both commands are disabled**

Entry #1 and Entry #2 are compared first. Entry #2 is selected as it has a lower BGP Router ID. The MED is not checked as the paths are originated from neighbors in different neighboring ASes. Entry #2 and Entry #3 are compared next. Entry #2 is selected as the best path as it is an EBG route.

```
RT4#sh ip bgp
BGP table version is 3, local router ID is 4.4.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*  192.168.1.0      3.3.3.3            150          0 300 100 i
*> 1.1.1.1          1.1.1.1            200          0 200 100 i
*  i                2.2.2.2            100          100  0 300 100 i
RT4#
```

- **Scenario #2: bgp deterministic-med is disabled; bgp always-compare-med is enabled**

Entry #1 and Entry #2 are compared first. These entries are from different neighboring ASes, but the MED is used in the comparison as the **bgp always-compare-med** command is enabled. Entry #1 is selected as it has a lower MED. Entry #1 and Entry #3 are compared next. The MED is used in the comparison again. Entry #3 is selected as the best path as it has a lower MED.

```
RT4#sh ip bgp
BGP table version is 3, local router ID is 4.4.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*  192.168.1.0      3.3.3.3            150          0 300 100 i
*                   1.1.1.1            200          0 200 100 i
*>i 2.2.2.2          2.2.2.2            100          100  0 300 100 i
RT4#
```

- **Scenario #3: bgp deterministic-med is enabled; bgp always-compare-med is disabled**

When the **bgp deterministic-med** command is enabled, the routes received from the same AS are grouped together, and the best entries of each group are then compared to select the best path. The BGP routing table on RT4 looks like this:

- **Entry #1** – AS\_PATH 200 100, MED 200, external, RID 1.1.1.1
- **Entry #2** – AS\_PATH 300 100, MED 100, internal, RID 2.2.2.2
- **Entry #3** – AS\_PATH 300 100, MED 150, external, RID 3.3.3.3

Entry #1 is the best of its group as it is the only route originated from AS 200.

Entry #2 is the best for AS 300 as it has the lowest MED.

Entry #1 and Entry #2 are compared eventually. Since the entries are from different ASes and the **bgp always-compare-med** command is not enabled, the MED is not considered in the comparison. Entry #1 is selected as the best path as it is an EBG route.

```
RT4#sh ip bgp
BGP table version is 5, local router ID is 4.4.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 192.168.1.0      1.1.1.1            200          0 200 100 i
*  i                2.2.2.2            100          100  0 300 100 i
*                   3.3.3.3            150          0 300 100 i
RT4#
```

- **Scenario #4: Both commands are enabled**

The comparisons for this scenario is same as the previous scenario; expect for the last comparison between Entry #1 and Entry #2, in which the MED is considered in the comparison as the **bgp always-compare-med** command is enabled.

Entry #2 is selected as the best path as it has a lower MED.

```

RT4#sh ip bgp
BGP table version is 6, local router ID is 4.4.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*  192.168.1.0      1.1.1.1            200          0 200 100 i
*>i 2.2.2.2          2.2.2.2            100          0 100 100 i
*  3.3.3.3          3.3.3.3            150          0 300 100 i
RT4#

```

- It is recommended to implement the **bgp deterministic-med** command in all new network rollouts. For existing networks, the command must either be deployed on all routers at the same time; or incrementally, with care to avoid possible IBGP routing loops.
- If **bgp always-compare-med** is enabled, MEDs are compared for all paths. This option must be enabled on BGP routers across an entire AS; otherwise routing loops can occur.

### The Propagation of the MED Attribute

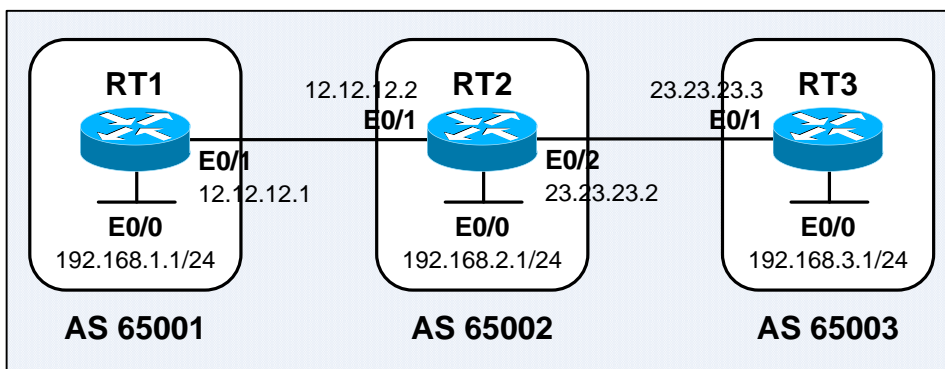


Figure A6-43: The Propagation of the MED Attribute

- The network setup above is used to demonstrate the propagation of the MED attribute across ASes. The MED value for 192.168.1.0/24 originated from AS 65001 only propagated to AS 65002. Therefore the MED for the 192.168.1.0/24 is not shown in the BGP routing table on RT3.

```

RT3#sh ip bgp
BGP table version is 4, local router ID is 192.168.3.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 192.168.1.0      23.23.23.2          0          0 65002 65001 i
*> 192.168.2.0      23.23.23.2          0          0 65002 i
*> 192.168.3.0      0.0.0.0             0          0 32768 i
RT3#

```

## BGP RIB Failure

- When BGP tries to install the best-path prefix into the **Routing Information Base (RIB)**, it might be rejected due to there is an IGP or static route with lower administrative distance already exists in the IP routing table.

A BGP route that is not being installed into the RIB is also known as an **inactive route**.

**Note:** Routing Information Base (RIB) is simply another name for the IP routing table.

- The **show ip bgp rib-failure EXEC** command displays the BGP routes that failed to be installed into the RIB, but are installed into the BGP table as RIB-failure routes.

**Note:** The network topology that is used for this article is Page 223 - Advanced BGP Network.

```

RT2#sh ip bgp
BGP table version is 9, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network                Next Hop                Metric LocPrf Weight Path
*> 192.168.1.0            1.1.1.1                  0             0 65001 i
*> 192.168.2.0            0.0.0.0                  0             32768 i
r>i192.168.3.0            3.3.3.3                  0            100      0 i
r>i192.168.4.0            4.4.4.4                  0            100      0 i
RT2#
RT2#sh ip route 192.168.0.0 255.255.0.0 longer-prefixes

Gateway of last resort is not set

B   192.168.1.0/24 [20/0] via 1.1.1.1, 00:00:55
C   192.168.2.0/24 is directly connected, Ethernet0/0
O   192.168.3.0/24 [110/20] via 23.23.23.3, 00:01:05, Ethernet0/2
O   192.168.4.0/24 [110/20] via 24.24.24.4, 00:01:05, Ethernet0/3
RT2#
RT2#sh ip bgp rib-failure
Network                Next Hop                RIB-failure          RIB-NH Matches
192.168.3.0            3.3.3.3                Higher admin distance n/a
192.168.4.0            4.4.4.4                Higher admin distance n/a
RT2#

```

- MISC notes for the output of the **show ip bgp rib-failure EXEC** command:
  - An entry of 0.0.0.0 of the Next Hop column indicates that the router has some non-BGP routes to the particular network. \*TBC\*
  - The RIB-NH Matches column indicates whether the next hops of the BGP and IP routes are the same. It is valid (Yes / No) only when the Higher admin distance reason appears in the RIB-failure column and the **bgp suppress-inactive** BGP router subcommand is configured for the address family that is being used.

Yes	The next hop of the route in the RIB is same as the next hop of the BGP route or recurses down to the same adjacency as the next hop of the BGP route.
No	The next hop of the route in the RIB recurses down differently from the next hop of the BGP route. BGP route that has a different next hop than the next hop of the IP route will not be advertised to other BGP neighbors (both IBGP and EBGP) – it is being suppressed.
n/a	<b>bgp suppress-inactive</b> is not configured for the address family that is being used.

## BGP Backdoor Routes

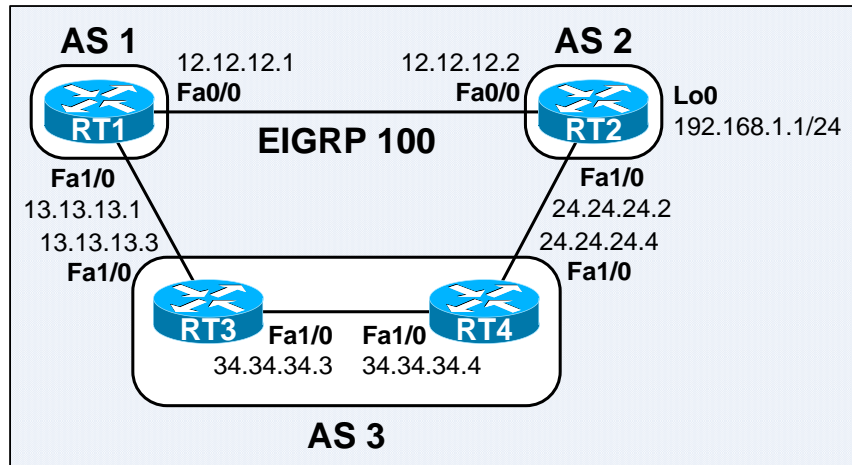


Figure A6-44: BGP Backdoor Routes

- The **backdoor** option of the **network** BGP router subcommand changes the administrative distance of EBGP to allow IGP routes favored over EBGP routes for specific network numbers.
- The figure above shows that AS 1 and AS 2 are running an IGP (EIGRP) on the private link between them, and are running EBGP with AS 3. RT1 in AS 1 will receive advertisements about 192.168.1.0/24 from AS 3 via EBGP with an administrative distance of 20 and from AS 2 via EIGRP with an administrative distance of 90. RT1 will use the EBGP route via AS 3 to reach 192.168.1.0/24 as the lower administrative distance is preferred.
- Below shows that implementing the **network 192.168.1.0 backdoor** BGP router subcommand on RT1 changes the administrative distance of the EBGP route 192.168.1.0/24 from 20 to 200, and eventually makes the EIGRP route with an administrative distance of 90 more preferred. Note that the command will not cause BGP on RT1 to generate an advertisement for that network.

```
RT1#sh ip route

Gateway of last resort is not set

    12.0.0.0/24 is subnetted, 1 subnets
C       12.12.12.0 is directly connected, FastEthernet0/0
B 192.168.1.0/24 [20/0] via 13.13.13.3, 00:00:10
    13.0.0.0/24 is subnetted, 1 subnets
C       13.13.13.0 is directly connected, FastEthernet1/0
RT1#
RT1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT1(config)#router bgp 1
RT1(config-router)#network 192.168.1.0 backdoor
RT1(config-router)#end
RT1#
RT1#sh ip route

Gateway of last resort is not set

    12.0.0.0/24 is subnetted, 1 subnets
C       12.12.12.0 is directly connected, FastEthernet0/0
D 192.168.1.0/24 [90/156160] via 12.12.12.2, 00:00:05, FastEthernet0/0
    13.0.0.0/24 is subnetted, 1 subnets
C       13.13.13.0 is directly connected, FastEthernet1/0
RT1#
```

## BGP Route Reflection and Peer Group

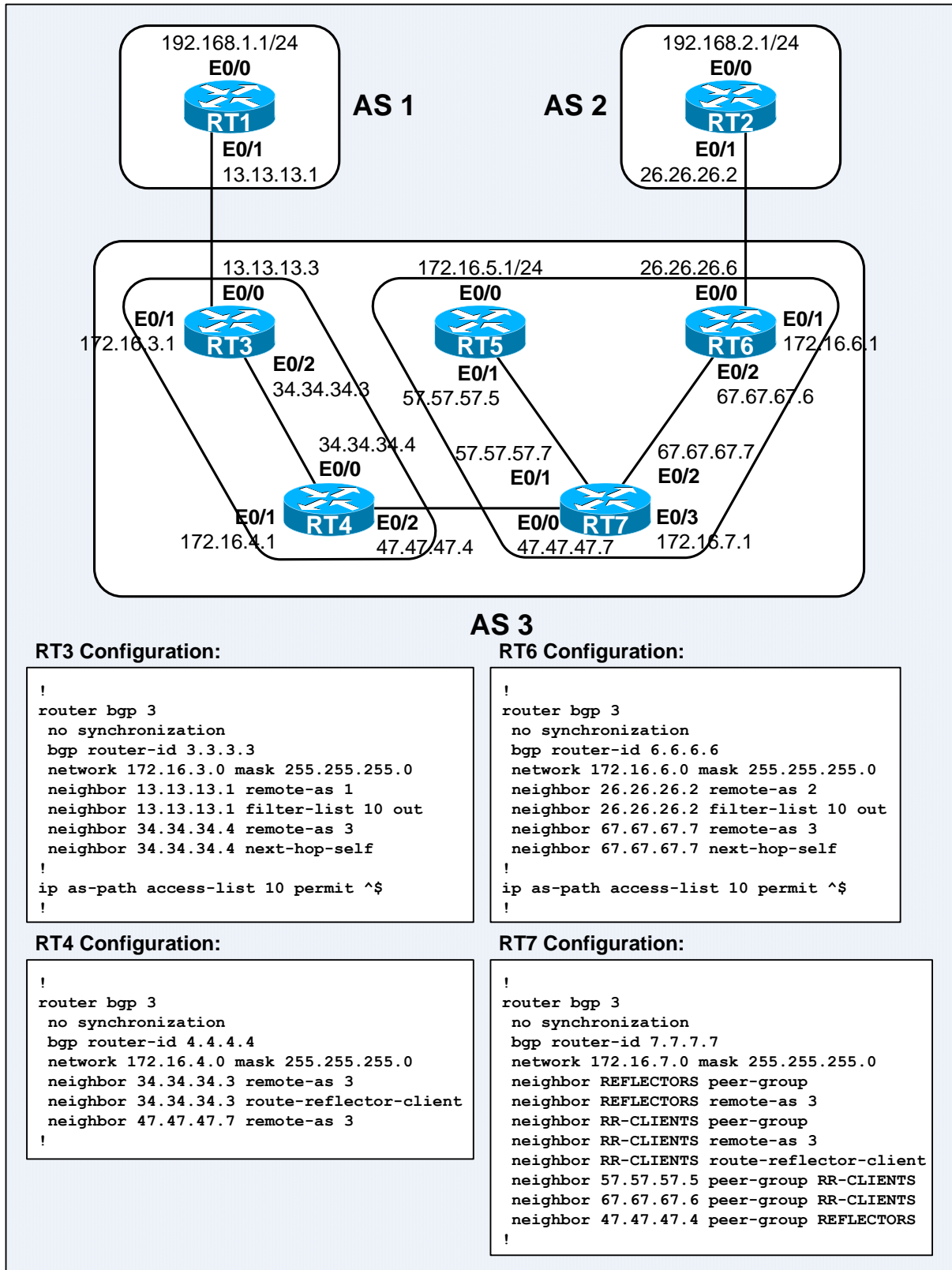


Figure A6-45: BGP Route Reflection and Peer Group

- This example illustrates a practical use of route reflectors and peer groups. RT3 and RT4 form a route reflector cluster, where RT4 is the route reflector; RT5, RT6, and RT7 form another cluster where RT7 is the route reflector. RT4 and RT7 are part of a peer group called REFLECTORS; if there are other route reflectors, all should establish full-mesh IBGP peering. RT7 groups all its route reflector clients in a peer group called RR-CLIENTS, where common policies can be applied.
- From RT5 and RT6's perspective, the IBGP peering session with RT7 is a normal IBGP session – the client need not know that it is a client.
- Below shows the BGP table on RT3 for all the routes in the network. Take notes upon the originator and cluster list for the different routes.

```

RT3#sh ip bgp 192.168.1.0
BGP routing table entry for 192.168.1.0/24, version 2
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
    34.34.34.4
    1
      13.13.13.1 from 13.13.13.1 (1.1.1.1)
        Origin IGP, metric 0, localpref 100, valid, external, best
RT3#
RT3#sh ip bgp 192.168.2.0
BGP routing table entry for 192.168.2.0/24, version 8
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
    2
      67.67.67.6 (metric 30) from 34.34.34.4 (4.4.4.4)
        Origin IGP, metric 0, localpref 100, valid, internal, best
        Originator: 6.6.6.6, Cluster list: 4.4.4.4, 7.7.7.7
RT3#
RT3#sh ip bgp 172.16.3.0
BGP routing table entry for 172.16.3.0/24, version 3
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
    13.13.13.1 34.34.34.4
  Local
    0.0.0.0 from 0.0.0.0 (3.3.3.3)
      Origin IGP, metric 0, localpref 100, weight 32768, valid, sourced, local,
best
RT3#
RT3#sh ip bgp 172.16.4.0
BGP routing table entry for 172.16.4.0/24, version 4
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
    13.13.13.1
  Local
    34.34.34.4 from 34.34.34.4 (4.4.4.4)
      Origin IGP, metric 0, localpref 100, valid, internal, best
RT3#
RT3#sh ip bgp 172.16.5.0
BGP routing table entry for 172.16.5.0/24, version 7
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
    13.13.13.1
  Local
    57.57.57.5 (metric 30) from 34.34.34.4 (4.4.4.4)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Originator: 5.5.5.5, Cluster list: 4.4.4.4, 7.7.7.7
RT3#

```

```

RT3#sh ip bgp 172.16.6.0
BGP routing table entry for 172.16.6.0/24, version 6
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
    13.13.13.1
  Local
    67.67.67.6 (metric 30) from 34.34.34.4 (4.4.4.4)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Originator: 6.6.6.6, Cluster list: 4.4.4.4, 7.7.7.7
RT3#
RT3#sh ip bgp 172.16.7.0
BGP routing table entry for 172.16.7.0/24, version 5
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
    13.13.13.1
  Local
    47.47.47.7 (metric 20) from 34.34.34.4 (4.4.4.4)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Originator: 7.7.7.7, Cluster list: 4.4.4.4
RT3#

```

- A route may carry a cluster list that contains the Router IDs of all the route reflectors that it passed through.
- When multiple route reflectors are configured in the same cluster, all the route reflectors must be configured with a common Cluster ID to detect routing loops that might occur between clusters. The Cluster ID is a number identifying a route reflector cluster. Note that it is a good practice to configure a Cluster ID even if using a single route reflector.  
**Ex:** If RT5 were to be configured as a route reflector, the **bgp cluster-id {number}** BGP router subcommand is required on RT5 and RT7. Below shows the syntax of the command.

```

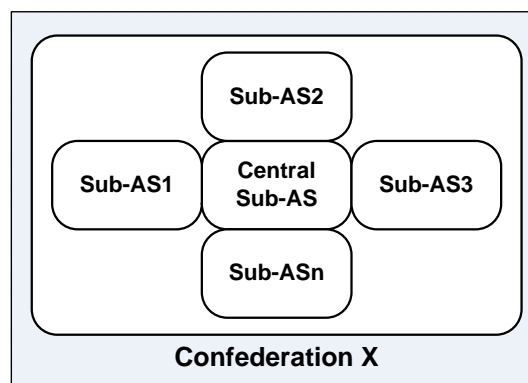
Router(config-router)#bgp cluster-id ?
  <1-4294967295>  Route-Reflector Cluster-id as 32 bit quantity
  A.B.C.D        Route-Reflector Cluster-id in IP address format

Router(config-router)#

```

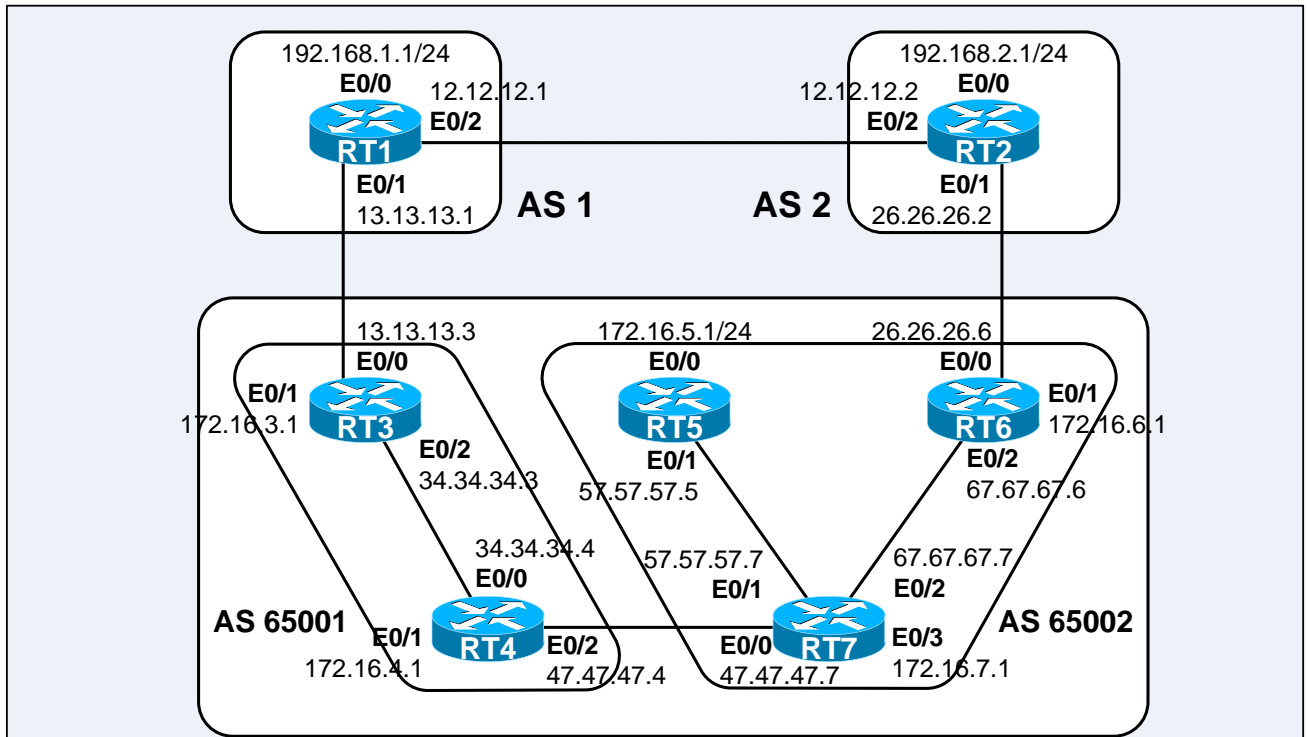
## BGP Confederations

- Confederation is another method to handle the scalability issue of full mesh IBGP within an AS. Confederation is based on the concept that an AS can be broken into multiple smaller sub-ASes. All the IBGP rules still apply inside each sub-AS, eg: all the BGP routers must be fully meshed. EBGP must run between the sub-ASes as they have different ASNs. The ASNs could be chosen from the private AS range (64512 to 65535) to avoid the hassles of applying formal ASNs. Although EBGP is used to exchange routes between sub-ASes of a confederation, routing inside a confederation behaves like IBGP routing in a single AS – the NEXT\_HOP, LOCAL\_PREF, and MED information is preserved throughout the sub-ASes. A confederation still behaves and looks like a single AS or routing domain to the outside world.  
**Note:** Some texts refer to Sub-AS as Member-AS.
- Without confederations, EBGP routes are preferred over IBGP routes in the BGP best path selection algorithm. Confederations introduced a new type of EBGP route between sub-ASes – the confederation external or EIBGP route. BGP prefers routes in the following manner: \*TBC\*  
**EBGP routes to outside the confederation > EIBGP routes > IBGP routes**
- Confederations can easily detect routing loops inside the whole AS because EBGP is run between the sub-ASes. The AS path list is a loop avoidance mechanism used to detect routing updates leaving a sub-AS and attempting to reenter the same sub-AS. A routing update that tries to reenter a sub-AS it originated from is detected when the sub-AS will see its own sub-AS number listed in the AS path of the update.
- The main drawbacks of confederations are the migration from a non-confederation to a confederation design requires major reconfiguration of the routers and a major change upon the logical topology, and do not provide the capabilities for implementing policies between sub-ASes, as the whole AS is still considered one entity. Additionally, since sub-ASs do not influence the overall AS path length – all paths inside a confederation have exactly the same AS path length, suboptimal routing through a confederation could occur without manually tuning the policies, such as using the local preference attribute.
- Choosing and connecting the sub-ASes randomly inside a confederation will lead to problems. Unnecessary processing might occur as a sub-AS can end up receiving duplicate information from sub-ASes throughout the same path. Additionally, suboptimal routing can occur as all paths inside the confederation have exactly the same AS path length. Experience has proven that a centralized confederation architecture, in which all sub-ASes exchange information with each other through a central sub-AS backbone, and each sub-AS interacts with only one other sub-AS, produces a uniform routing through the AS path length and route exchange within confederation, results in the most optimal routing behavior.



**Figure A6-46:** Centralized Confederation Architecture





### AS 3

#### RT3 Configuration:

```
!
router ospf 100
router-id 3.3.3.3
network 13.13.13.0 0.0.0.255 area 1
network 34.34.34.0 0.0.0.255 area 1
passive-interface Ethernet0/0
!
router bgp 65001
no synchronization
bgp router-id 3.3.3.3
bgp confederation identifier 3
network 172.16.3.0 mask 255.255.255.0
neighbor 13.13.13.1 remote-as 1
neighbor 13.13.13.1 filter-list 10 out
neighbor 34.34.34.4 remote-as 65001
!
ip as-path access-list 10 permit ^(\([0-9]*\))*$
!
```

#### RT6 Configuration:

```
!
router ospf 100
router-id 6.6.6.6
network 67.67.67.0 0.0.0.255 area 0
!
router bgp 65002
no synchronization
bgp router-id 6.6.6.6
bgp confederation identifier 3
network 172.16.6.0 mask 255.255.255.0
neighbor 26.26.26.2 remote-as 2
neighbor 26.26.26.2 filter-list 10 out
neighbor 57.57.57.5 remote-as 65002
neighbor 57.57.57.5 next-hop-self
neighbor 67.67.67.7 remote-as 65002
neighbor 67.67.67.7 next-hop-self
!
ip as-path access-list 10 permit ^(\([0-9]*\))*$
!
```

#### RT4 Configuration:

```
!
router ospf 100
router-id 4.4.4.4
network 34.34.34.0 0.0.0.255 area 1
network 47.47.47.0 0.0.0.255 area 0
passive-interface Ethernet0/2
!
router bgp 65001
no synchronization
bgp router-id 4.4.4.4
bgp confederation identifier 3
bgp confederation peers 65002
network 172.16.4.0 mask 255.255.255.0
neighbor 34.34.34.3 remote-as 65001
neighbor 47.47.47.7 remote-as 65002
neighbor 47.47.47.7 next-hop-self
!
```

#### RT7 Configuration:

```
!
router ospf 100
router-id 7.7.7.7
network 47.47.47.0 0.0.0.255 area 0
network 57.57.57.0 0.0.0.255 area 1
network 67.67.67.0 0.0.0.255 area 0
passive-interface Ethernet0/0
!
router bgp 65002
no synchronization
bgp router-id 7.7.7.7
bgp confederation identifier 3
bgp confederation peers 65001
network 172.16.7.0 mask 255.255.255.0
neighbor SUB-AS_65002 peer-group
neighbor SUB-AS_65002 remote-as 65002
neighbor 57.57.57.5 peer-group SUB-AS_65002
neighbor 67.67.67.6 peer-group SUB-AS_65002
neighbor 47.47.47.4 remote-as 65001
neighbor 47.47.47.4 next-hop-self
!
```

Figure A6-47: BGP Confederation

- AS 3 is divided into 2 smaller sub-ASes – AS 65001 and AS 65002. OSPF is used as the IGP in each sub-AS. The OSPF in AS 65001 is running independently of the OSPF in AS 65002, which means that the OSPF area numbers used in AS 65001 can be reused in AS 65002 – an IGP in a sub-AS runs independently of IGP in other sub-ASes.
- RT3 has all its interfaces reside in OSPF area 1. RT3 is running EBGP with RT1 in AS 1 and is running IBGP with RT4 in AS 65001. Take note that RT3 is advertising the DMZ link to AS 1 – 13.13.13.0/24 through OSPF, eliminating the need for the **next-hop-self** command for RT4; and defines interface Ethernet0/0 to AS 1 as a passive interface to avoid forming a neighborship. RT3 uses the **bgp confederation identifier 3** BGP router subcommand to present itself to RT1 as being part of confederation 3. The AS Confederation Identifier is the **externally visible** ASN, and it is the ASN used in OPEN messages and advertised in the AS\_PATH attribute.
- RT4 is the sub-AS 65001 border router that is running EIBGP with RT7 in sub-AS 65002. RT4 is also running IBGP with RT3. RT4 is an OSPF area border router that has its interfaces in areas 0 and 1. RT4 has disabled its OSPF processing on the link to RT7 using a passive interface, and only EIBGP is running on that link. RT4 uses the **bgp confederation peers 65002** BGP router subcommand to preserve all the attributes, eg: local preference and next hop when traversing the EIBGP session to AS 65002, which makes the confederation EBGP session with sub-AS 65002 look like an IBGP session. RT4 uses the **next-hop-self** command to set the next-hop address of routes advertising from RT4 to RT7 to RT4's IP address – 47.47.47.4. Without this command, the next-hop address of the EBGP route from AS 1 (192.168.1.0/24) will be sent from RT4 to RT7 with the external next hop 13.13.13.1, which may become unreachable for routers in sub-AS 65002 depends upon the network design and implementation. **Tip:** When implementing a confederation, include several extra internal AS numbers in the **bgp confederation peers** BGP router subcommand to ease the addition of new sub-ASes in the future without having to reconfigure all BGP routers throughout the network.
- RT7 is also an area border router in areas 0 and 1. Areas 0 and 1 in AS 65002 are totally independent of areas 0 and 1 in AS 65001. The IGP is shielded from each other by EBGP. Full-mesh IBGP sessions are implemented between RT5, RT6, and RT7 in sub-AS 65002.
- RT6 is a border router for confederation 3. RT6 is running EBGP with RT2 in AS 2 and a full mesh IBGP with RT5 and RT7 in sub-AS 65002. RT6 has all its interfaces reside in OSPF area 0. Note that RT6 is not running OSPF on the external link to AS 2 and therefore the next hop for EBGP routes received on RT6 must be set to itself before propagating the routes to RT5 and RT7.
- The AS path access list would match all the routes that originated locally from the confederation.

```

RT3#sh ip bgp regexp ^(\([0-9]*\))*$
BGP table version is 12, local router ID is 3.3.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop           Metric LocPrf Weight Path
*> 172.16.3.0/24    0.0.0.0             0         32768 i
*>i172.16.4.0/24    34.34.34.4          0         100     0 i
*>i172.16.5.0/24    47.47.47.7          0         100     0 (65002) i
*>i172.16.6.0/24    47.47.47.7          0         100     0 (65002) i
*>i172.16.7.0/24    47.47.47.7          0         100     0 (65002) i
RT3#

```

- Below shows how RT1 sees all routes via 2 paths – one via AS 2 and one via AS 3. Note that all the sub-ASes are hidden from RT1. RT1 and RT2 in AS 1 and AS 2 have no visibility upon the sub-ASes inside confederation 3.

```

RT1#sh ip bgp
BGP table version is 13, local router ID is 1.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*   172.16.3.0/24    12.12.12.2                0      2 3 i
*> 172.16.4.0/24    13.13.13.3                0      3 i
*   172.16.5.0/24    12.12.12.2                0      2 3 i
*> 172.16.6.0/24    13.13.13.3                0      3 i
*   172.16.7.0/24    12.12.12.2                0      2 3 i
*> 192.168.1.0      0.0.0.0                0      32768 i
*> 192.168.2.0      12.12.12.2                0      0 2 i
RT1#

```

- Below shows the BGP table on RT3. Note that all the sub-ASes are indicated between parentheses. Any path taken between sub-ASes has an AS path length of 0. Notice how prefix 192.168.2.0/24 is learned via 2 paths – one internal via (65002) 2, and the other external via 1 2. The AS path length of the internal route via (65002) 2 is considered shorter, as the sub-ASes are not being used to determine the AS path length.

```

RT3#sh ip bgp
BGP table version is 9, local router ID is 3.3.3.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.3.0/24    0.0.0.0                0      32768 i
*>i172.16.4.0/24    34.34.34.4                0     100    0 i
*>i172.16.5.0/24    47.47.47.7                0     100    0 (65002) i
*>i172.16.6.0/24    47.47.47.7                0     100    0 (65002) i
*>i172.16.7.0/24    47.47.47.7                0     100    0 (65002) i
*> 192.168.1.0      13.13.13.1                0      0 1 i
*>i192.168.2.0      47.47.47.7                0     100    0 (65002) 2 i
*                   13.13.13.1                0      0 1 2 i
RT3#

```

- Below shows how RT7 considers all routes coming from sub-AS 65001 as confederation external routes – confed-external.

```

RT7#sh ip bgp 192.168.1.0
BGP routing table entry for 192.168.1.0/24, version 5
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Advertised to peer-groups:
    SUB-AS_65002
    (65001) 1
      47.47.47.4 from 47.47.47.4 (4.4.4.4)
        Origin IGP, metric 0, localpref 100, valid, confed-external, best
RT7#
RT7#sh ip bgp 172.16.3.0
BGP routing table entry for 172.16.3.0/24, version 2
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Advertised to peer-groups:
    SUB-AS_65002
    (65001)
      47.47.47.4 from 47.47.47.4 (4.4.4.4)
        Origin IGP, metric 0, localpref 100, valid, confed-external, best
RT7#

```

- Cisco recommends the use of route reflection to solve IBGP mesh scalability issues; and use confederations to run an IGP in a sub-AS independently of IGP in other sub-ASes in order to control the instability of large IGP routing domains. Actual deployments have proven that route reflections are more flexible to implement and maintain. Route reflections can be used in conjunction with confederations, in which an AS can be divided into sub-ASes that each run route reflections internally.

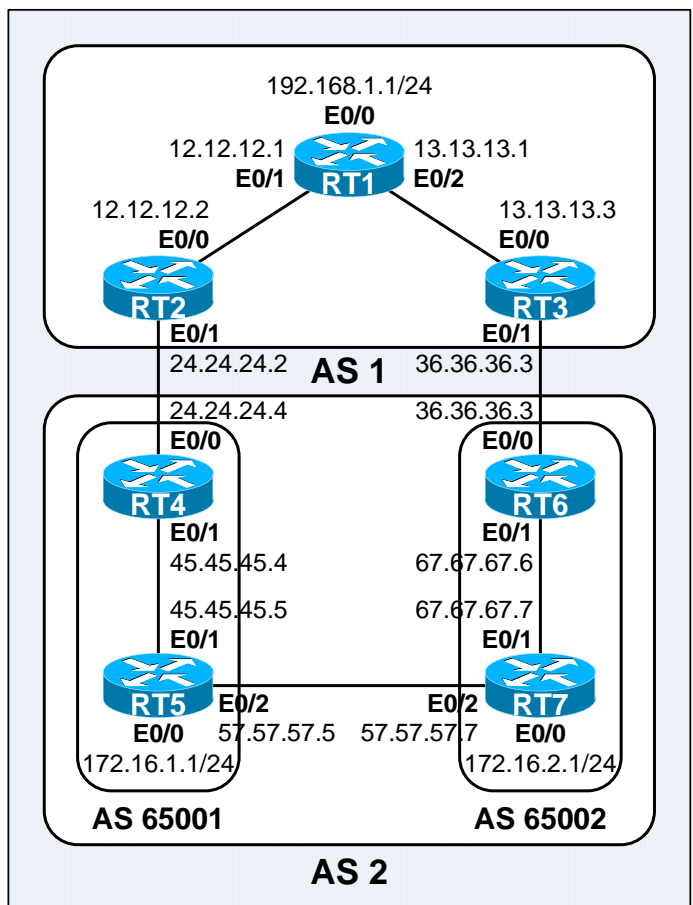


Figure A6-48: BGP Confederation Route Selection

- Below shows the routing towards 192.168.1.0/24 on RT5 and RT7. Note that both the Confederation External and Confederation Internal are treated as internal routes. RT7 selects the path through RT5 instead of RT6 as RT5 has a lower BGP Router ID than RT6. RT7 does not select the path through RT5 due to it is a confed-external route. This is proven by resetting the BGP sessions on RT4, and RT5 will then learn the confed-external route through RT7; but RT5 will eventually select the confed-internal route through RT4 when RT4 is up again.

```

RT5#sh ip bgp
BGP table version is 4, local router ID is 5.5.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.1.0/24    0.0.0.0            0         32768 i
*> 172.16.2.0/24    57.57.57.7         0         100     0 (65002) i
*>i192.168.1.0    45.45.45.4         0         100     0 1 i
RT5#
=====
RT7#sh ip bgp
BGP table version is 5, local router ID is 7.7.7.7
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.1.0/24    57.57.57.5         0         100     0 (65001) i
*> 172.16.2.0/24    0.0.0.0            0         32768 i
*> 192.168.1.0    57.57.57.5         0         100     0 (65001) 1 i
* i                 67.67.67.6         0         100     0 1 i
RT7#
RT7#sh ip bgp 192.168.1.0
BGP routing table entry for 192.168.1.0/24, version 3
Paths: (2 available, best #1, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
    67.67.67.6
    (65001) 1
      57.57.57.5 from 57.57.57.5 (5.5.5.5)
        Origin IGP, metric 0, localpref 100, valid, confed-external, best
    1
      67.67.67.6 from 67.67.67.6 (6.6.6.6)
        Origin IGP, metric 0, localpref 100, valid, confed-internal
RT7#

```

- By manipulating the BGP Router ID of RT4 from 4.4.4.4 to 8.8.8.8, RT5 select the confed-external to 192.168.1.0/24 through RT7 instead of RT4.

```

RT5#sh ip bgp 192.168.1.0
BGP routing table entry for 192.168.1.0/24, version 6
Paths: (2 available, best #2, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
    45.45.45.4
    1
      45.45.45.4 from 45.45.45.4 (8.8.8.8)
        Origin IGP, metric 0, localpref 100, valid, confed-internal
    (65002) 1
      57.57.57.7 from 57.57.57.7 (7.7.7.7)
        Origin IGP, metric 0, localpref 100, valid, confed-external, best
RT5#

```

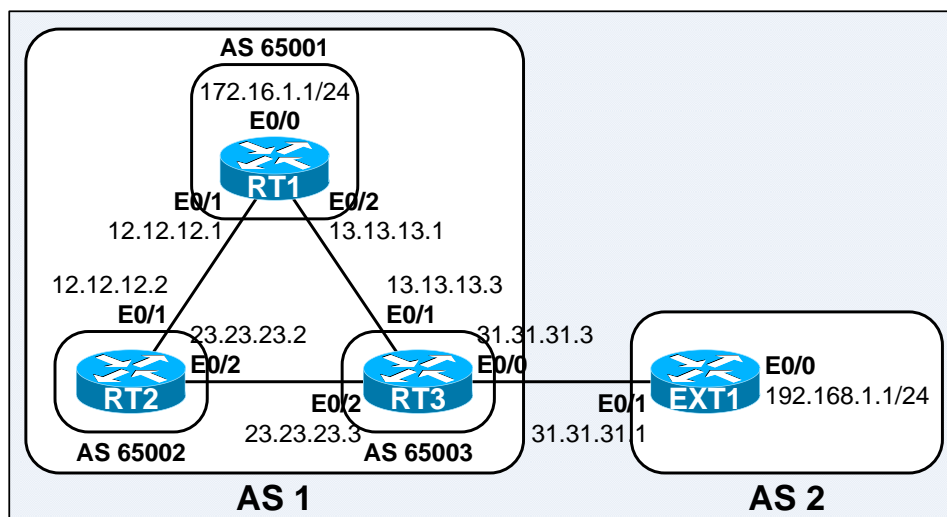
## Differences between BGP Route Reflection and Confederations

- The table below compares how route reflection and confederations handle the scaling problem of the IBGP full mesh requirement.

Feature	BGP Route Reflection	BGP Confederations
<b>Hierarchical Roles</b>	Route reflector and route reflector client roles exist.	Some sub-AS might act as backbone, depending on the network design.
<b>Topology</b>	The AS is divided into clusters. Clusters are formed of route reflectors and their clients. In an AS with route reflector clusters, there can be routers that are not members of any cluster.	The AS becomes a confederation of sub-ASes; and each sub-AS contains a group of BGP routers. All routers outside of sub-ASes must have confederation configuration.
<b>Configuration</b>	Route reflector clients are peered with the route reflectors within each cluster. No configuration is required on route reflector clients. The route reflector clients and the Cluster IDs are configured on the route reflectors. A cluster can contain more than one route reflector.	Configure the confederation identifier (the ASN that appears to external ASes), the local sub-ASN, and the peer sub-ASNs on the routers within the confederation.
<b>Peering</b>	Route reflectors are peered with each other in full-mesh. Within a cluster, the route reflector client only peers with the route reflectors.	Sub-ASes do not require full-mesh EIBGP peering. Full-mesh IBGP peering is required within a sub-AS. Route reflection can be implemented to solve the IBGP scalability issue.
<b>Communication between peers</b>	IBGP is used between route reflectors and their clients using new attributes specific to route reflection.	IBGP is used within each sub-AS. EBGP-like protocol – EIBGP, or Confederation External, is used between sub-ASes. EIBGP enhances the AS_PATH attribute and changes the handling of the NEXT_HOP, LOCAL_PREF, and MED attributes over EBGP.
<b>Changes or Additions to the BGP attributes</b>	Introduce the Originator ID and Cluster ID optional non-transitive attributes. These attributes are contained within an AS.	Enhance the AS_PATH attribute with Type 3 AS_CONFED_SET and Type 4 AS_CONFED_SEQ. These types are contained within the confederation AS and are not advertised to external ASes.
<b>Handling of the NEXT_HOP, LOCAL_PREF, and MED attributes</b>	IBGP is used between route reflectors and their clients; the attributes are preserved and passed as it is.	Although EBGP-like communication occurs between sub-ASes, the attributes are preserved and passes as it is, which is contrary to EBGP.

<b>Re-advertising a learned prefix</b>	Route reflectors re-advertise prefixes learned from a client to other clients or non-client peers. And due to the IBGP full-mesh peering between route reflectors, a route reflector does not re-advertise a prefix learned from a non-client peer to other peers.	As an EBGP-like protocol is used between sub-ASes, prefixes learned from a sub-AS can be re-advertised to other sub-ASes if they are selected as best and being used.
<b>Communication with non member BGP peers</b>	If route reflectors peer with non-route reflector routers in the same AS, the route reflector attributes are ignored due to the nature of the attributes – optional non-transitive.	If a member of the confederation is peering with a BGP peer in another AS, the sub-ASNs in the AS_PATH are suppressed and only the confederation identifier is placed and passed in the AS_PATH attribute.
<b>Support of multiple instances</b>	A router can be a member of multiple clusters, provided that it is configured as a route reflector.	Not possible. A router can only be a member of a single sub-AS or a single confederation.
<b>Uses the multi-hop parameter</b>	Not required.	Might be required. EIBGP is a derivative of EBGP. If the directly connected interface IP address of the neighbor is not configured as the peer IP address, the multi-hop configuration will be required.

- Confederations place traversed sub-ASNs in the AS\_PATH attribute; therefore it is easier to trace the path of the prefix. Route reflectors place only the Originator IDs and Cluster IDs in the AS\_PATH attribute.
- Confederations allow the comparison of the sub-AS path length when multiple routers in different sub-ASes advertise the same prefix, in order to prevent sub-optimal routing within an AS. Below shows a sample scenario in which RT1, RT2, and RT3 are routers in the same confederation and each of them belongs to a separate sub-AS. EXT1 is an EBGP peer with RT3. EXT1 advertises 192.168.1.0/24 to RT3. RT3 re-advertises it to RT1 and RT2. RT1 and RT2 also re-advertise the route to each other. Eventually RT1 has a route to 192.168.1.0/24 from both RT2 and RT3. RT1 may choose the longer path through RT2 to reach 192.168.1.0/24! Most BGP implementations do not consider the sub-AS path length but rather depends upon other factors, eg: BGP Router ID.



**Figure A6-49: Sub-optimal Routing in BGP Confederation**

- Below shows the BGP table and IP routing table on RT1. Note that even though RT1 selects the path through RT2 as shown in the BGP table, RT1 actually forwards the packets destined to 192.168.1.0/24 through 31.31.31.1, and the shortest path to reach it is via OSPF through RT3! This seems like sub-optimal routing but in fact it is not. ☺

```

RT1#sh ip bgp
BGP table version is 11, local router ID is 1.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.1.0/24    0.0.0.0           0         32768 i
*> 192.168.1.0      31.31.31.1        0         100      0 (65002 65003) 2 i
*                   31.31.31.1        0         100      0 (65003) 2 i
RT1#
RT1#sh ip bgp 192.168.1.0
BGP routing table entry for 192.168.1.0/24, version 5
Paths: (2 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
  Advertised to non peer-group peers:
  13.13.13.3
  (65002 65003) 2
    31.31.31.1 (metric 20) from 12.12.12.2 (2.2.2.2)
      Origin IGP, metric 0, localpref 100, valid, confed-external, best
  (65003) 2
    31.31.31.1 (metric 20) from 13.13.13.3 (3.3.3.3)
      Origin IGP, metric 0, localpref 100, valid, confed-external
RT1#
RT1#sh ip route 192.168.1.0
Routing entry for 192.168.1.0/24
  Known via "bgp 65001", distance 200, metric 0
  Tag 65002, type internal
  Last update from 31.31.31.1 00:00:25 ago
  Routing Descriptor Blocks:
  * 31.31.31.1, from 12.12.12.2, 00:02:58 ago
    Route metric is 0, traffic share count is 1
    AS Hops 1

RT1#
RT1#ping 192.168.1.1 source 172.16.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
Packet sent with a source address of 172.16.1.1
!!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/47/84 ms
RT1#

```



- Below shows the BGP table on RT1 when RT3 is configured with the **next-hop-self** settings for RT1 and RT2.

```

RT1#sh ip bgp
BGP table version is 3, local router ID is 1.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.1.0/24    0.0.0.0           0             32768 i
*   192.168.1.0     23.23.23.3        0           100          0 (65002 65003) 2 i
*>   13.13.13.3    13.13.13.3      0          100         0 (65003) 2 i
RT1#
RT1#sh ip bgp 192.168.1.0
BGP routing table entry for 192.168.1.0/24, version 3
Paths: (2 available, best #2, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
    12.12.12.2
    (65002 65003) 2
    23.23.23.3 (metric 20) from 12.12.12.2 (2.2.2.2)
      Origin IGP, metric 0, localpref 100, valid, confed-external
    (65003) 2
    13.13.13.3 from 13.13.13.3 (3.3.3.3)
      Origin IGP, metric 0, localpref 100, valid, confed-external, best
RT1#

```

- When merging a number of ASes into a single big AS, confederations can be used as an interim solution for easy migration.

## BGP Route Map Logic

- A route map is an extremely powerful and versatile tool for route filtering and manipulating BGP attributes. In regards of BGP, route maps are used in the following commands:  
**aggregate-address** address mask **advertise-map** route-map-name  
**aggregate-address** address mask **as-set** route-map-name  
**aggregate-address** address mask **attribute-map** route-map-name  
**aggregate-address** address mask **route-map** route-map-name  
**aggregate-address** address mask **suppress-map** route-map-name  
**bgp dampening route-map** route-map-name  
**neighbor ip-addr advertise-map** route-map-name {**exist-map** | **non-exist-map**} map-name  
**neighbor ip-addr default-originate route-map** route-map-name  
**neighbor ip-addr route-map** route-map-name {**in** | **out**}  
**neighbor ip-addr unsuppress-map** route-map-name  
**redistribute protocol route-map** route-map-name

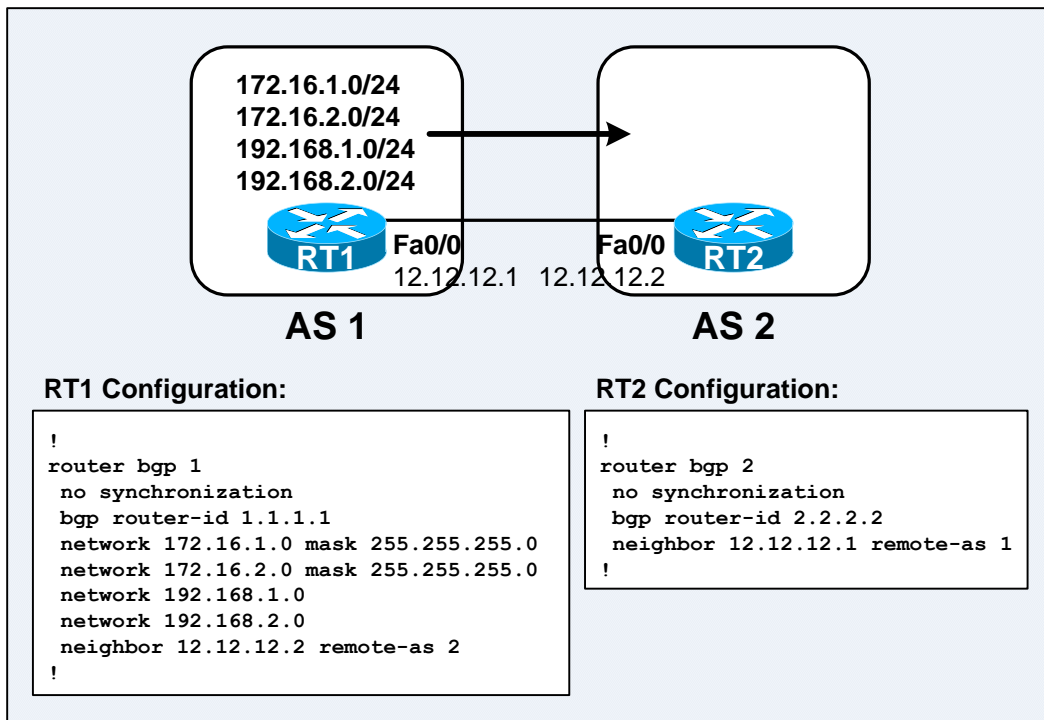


Figure A6-50: Network Setup for BGP Route Map Logic

- The network setup above is used to demonstrate the logic of route maps. Below shows the BGP table on RT2 prior to implementing any route map.

```
RT2#sh ip bgp
BGP table version is 5, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf  Weight  Path
*> 172.16.1.0/24    12.12.12.1         0           0   1   i
*> 172.16.2.0/24    12.12.12.1         0           0   1   i
*> 192.168.1.0      12.12.12.1         0           0   1   i
*> 192.168.2.0      12.12.12.1         0           0   1   i
RT2#
```

- Below shows the effect upon implementing a route map on RT2 but the route map is not defined – all routes are denied.

```

RT2#debug ip bgp updates
BGP updates debugging is on
RT2#
RT2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RT2(config)#router bgp 2
RT2(config-router)#neighbor 12.12.12.1 route-map map01 in
RT2(config-router)#exit
RT2(config)#
RT2(config)#do clear ip bgp 12.12.12.1 in
RT2(config)#
00:01:53: BGP(0): 12.12.12.1 rcvd UPDATE w/ attr: nexthop 12.12.12.1, origin i,
metric 0, path 1
00:01:53: BGP(0): 12.12.12.1 rcvd 172.16.1.0/24 -- DENIED due to: route-map;
00:01:53: BGP(0): no valid path for 172.16.1.0/24
00:01:53: BGP(0): 12.12.12.1 rcvd 172.16.2.0/24 -- DENIED due to: route-map;
00:01:53: BGP(0): no valid path for 172.16.2.0/24
00:01:53: BGP(0): 12.12.12.1 rcvd 192.168.1.0/24 -- DENIED due to: route-map;
00:01:53: BGP(0): no valid path for 192.168.1.0/24
00:01:53: BGP(0): 12.12.12.1 rcvd 192.168.2.0/24 -- DENIED due to: route-map;
00:01:53: BGP(0): no valid path for 192.168.2.0/24
00:01:53: BGP(0): nettable_walker 172.16.1.0/24 no best path
00:01:53: BGP(0): nettable_walker 172.16.2.0/24 no best path
00:01:53: BGP(0): nettable_walker 192.168.1.0/24 no best path
00:01:53: BGP(0): nettable_walker 192.168.2.0/24 no best path
00:01:53: BGP(0): 12.12.12.1 computing updates, afi 0, neighbor version 5, table
version 9, starting at 0.0.0.0
00:01:53: BGP(0): 12.12.12.1 update run completed, afi 0, ran for 4ms, neighbor
version 5, start version 9, throttled to 9
RT2(config)#
RT2(config)#do sh ip bgp

RT2(config)#

```

- Below shows the effect upon implementing an empty **route-map permit** statement on RT2.

```

RT2(config)#route-map map01 permit 10
RT2(config-route-map)#exit
RT2(config)#
RT2(config)#do clear ip bgp 12.12.12.1 in
RT2(config)#
00:02:23: BGP(0): 12.12.12.1 rcvd UPDATE w/ attr: nexthop 12.12.12.1, origin i,
metric 0, path 1
00:02:23: BGP(0): 12.12.12.1 rcvd 172.16.1.0/24
00:02:23: BGP(0): 12.12.12.1 rcvd 172.16.2.0/24
00:02:23: BGP(0): 12.12.12.1 rcvd 192.168.1.0/24
00:02:23: BGP(0): 12.12.12.1 rcvd 192.168.2.0/24
00:02:23: BGP(0): Revise route installing 1 of 1 route for 172.16.1.0/24 ->
12.12.12.1 to main IP table
00:02:23: BGP(0): Revise route installing 1 of 1 route for 172.16.2.0/24 ->
12.12.12.1 to main IP table
00:02:23: BGP(0): Revise route installing 1 of 1 route for 192.168.1.0/24 ->
12.12.12.1 to main IP table
00:02:23: BGP(0): Revise route installing 1 of 1 route for 192.168.2.0/24 ->
12.12.12.1 to main IP table
00:02:23: BGP(0): 12.12.12.1 computing updates, afi 0, neighbor version 9, table
version 13, starting at 0.0.0.0
00:02:23: BGP(0): 12.12.12.1 update run completed, afi 0, ran for 4ms, neighbor
version 9, start version 13, throttled to 13
RT2(config)#

```

```

RT2(config)#do sh ip bgp
BGP table version is 13, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.1.0/24    12.12.12.1         0           0 1 i
*> 172.16.2.0/24    12.12.12.1         0           0 1 i
*> 192.168.1.0      12.12.12.1         0           0 1 i
*> 192.168.2.0      12.12.12.1         0           0 1 i
RT2(config)#

```

The empty form of the **route-map permit** statement allows all routes – equivalent to **permit any** in an IP access list.

- Now changes the route map from **permit** to **deny**.  
The **deny** form of the empty route map denies all routes.

```

RT2(config)#route-map map01 deny 10
RT2(config-route-map)#exit
RT2(config)#
RT2(config)#do clear ip bgp 12.12.12.1 in
RT2(config)#
00:02:53: BGP(0): 12.12.12.1 rcvd UPDATE w/ attr: nexthop 12.12.12.1, origin i,
metric 0, path 1
00:02:53: BGP(0): 12.12.12.1 rcvd 172.16.1.0/24 -- DENIED due to: route-map;
00:02:53: BGP(0): no valid path for 172.16.1.0/24
00:02:53: BGP(0): 12.12.12.1 rcvd 172.16.2.0/24 -- DENIED due to: route-map;
00:02:53: BGP(0): no valid path for 172.16.2.0/24
00:02:53: BGP(0): 12.12.12.1 rcvd 192.168.1.0/24 -- DENIED due to: route-map;
00:02:53: BGP(0): no valid path for 192.168.1.0/24
00:02:53: BGP(0): 12.12.12.1 rcvd 192.168.2.0/24 -- DENIED due to: route-map;
00:02:53: BGP(0): no valid path for 192.168.2.0/24
00:02:53: BGP(0): nettable_walker 172.16.1.0/24 no best path
00:02:53: BGP(0): nettable_walker 172.16.2.0/24 no best path
00:02:53: BGP(0): nettable_walker 192.168.1.0/24 no best path
00:02:53: BGP(0): nettable_walker 192.168.2.0/24 no best path
00:02:53: BGP(0): 12.12.12.1 computing updates, afi 0, neighbor version 13,
table version 17, starting at 0.0.0.0
00:02:53: BGP(0): 12.12.12.1 update run completed, afi 0, ran for 4ms, neighbor
version 13, start version 17, throttled to 17
RT2(config)#
RT2(config)#do sh ip bgp

RT2(config)#

```

- The numbered **route-map** statement can be either a **permit** or **deny**; the **match** clause, if used, can also be either a **permit** or **deny**. Therefore, a route map has 4 basic forms or permutations:
  - **route-map permit, match permit**
  - **route-map permit, match deny**
  - **route-map deny, match permit**
  - **route-map deny, match deny**

- The first 2 forms are commonly used to allow certain routes to be accepted while denying others.

```

RT2(config)#access-list 1 permit 172.16.1.0 0.0.0.255
RT2(config)#route-map map01 permit 10
RT2(config-route-map)#match ip address 1
RT2(config-route-map)#exit
RT2(config)#
RT2(config)#do clear ip bgp 12.12.12.1 in
RT2(config)#
00:03:19: BGP(0): 12.12.12.1 rcvd UPDATE w/ attr: nexthop 12.12.12.1, origin i,
metric 0, path 1
00:03:19: BGP(0): 12.12.12.1 rcvd 172.16.1.0/24
00:03:19: BGP(0): 12.12.12.1 rcvd 172.16.2.0/24 -- DENIED due to: route-map;
00:03:19: BGP(0): 12.12.12.1 rcvd 192.168.1.0/24 -- DENIED due to: route-map;
00:03:19: BGP(0): 12.12.12.1 rcvd 192.168.2.0/24 -- DENIED due to: route-map;
00:03:19: BGP(0): Revise route installing 1 of 1 route for 172.16.1.0/24 ->
12.12.12.1 to main IP table
00:03:20: BGP(0): 12.12.12.1 computing updates, afi 0, neighbor version 17,
table version 18, starting at 0.0.0.0
00:03:20: BGP(0): 12.12.12.1 update run completed, afi 0, ran for 0ms, neighbor
version 17, start version 18, throttled to 18
RT2(config)#
RT2(config)#do sh ip bgp
BGP table version is 18, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.1.0/24    12.12.12.1         0             0 1 i
RT2(config)#

```

- In order to determine when the route map terminates execution upon a match, add another **route-map** statement upon the route map on RT2:

```

RT2(config)#route-map map01 permit 20
RT2(config-route-map)#set metric 10
RT2(config-route-map)#exit
RT2(config)#
RT2(config)#do clear ip bgp 12.12.12.1 in
RT2(config)#
00:03:46: BGP(0): 12.12.12.1 rcvd UPDATE w/ attr: nexthop 12.12.12.1, origin i,
metric 0, path 1
00:03:46: BGP(0): 12.12.12.1 rcvd 172.16.1.0/24...duplicate ignored
00:03:46: BGP(0): 12.12.12.1 rcvd 172.16.2.0/24
00:03:46: BGP(0): 12.12.12.1 rcvd 192.168.1.0/24
00:03:46: BGP(0): 12.12.12.1 rcvd 192.168.2.0/24
00:03:46: BGP(0): Revise route installing 1 of 1 route for 172.16.2.0/24 ->
12.12.12.1 to main IP table
00:03:46: BGP(0): Revise route installing 1 of 1 route for 192.168.1.0/24 ->
12.12.12.1 to main IP table
00:03:46: BGP(0): Revise route installing 1 of 1 route for 192.168.2.0/24 ->
12.12.12.1 to main IP table
00:03:51: BGP(0): 12.12.12.1 computing updates, afi 0, neighbor version 18,
table version 21, starting at 0.0.0.0
00:03:51: BGP(0): 12.12.12.1 update run completed, afi 0, ran for 4ms, neighbor
version 18, start version 21, throttled to 21
RT2(config)#

```

```

RT2(config)#do sh ip bgp
BGP table version is 21, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.1.0/24    12.12.12.1         0      0 1 i
*> 172.16.2.0/24    12.12.12.1        10      0 1 i
*> 192.168.1.0      12.12.12.1        10      0 1 i
*> 192.168.2.0      12.12.12.1        10      0 1 i
RT2(config)#

```

When there is a match using the **permit** statement, the route is accepted and the route map is terminated for that route. When there is a match using a **deny** statement, the route is not accepted, but execution continues and the route is processed by the next route map statement. If there is no other route map statement, all remaining routes are rejected.

- The last 2 forms of a route map contain a **route-map deny** statement. Change the route-map statement to **deny**, but do not change the IP access list.

```

RT2(config)#no route-map map01
RT2(config)#do sh route-map

RT2(config)#
RT2(config)#route-map map01 deny 10
RT2(config-route-map)#match ip address 1
RT2(config-route-map)#exit
RT2(config)#
RT2(config)#do clear ip bgp 12.12.12.1 in
RT2(config)#
00:04:18: BGP(0): 12.12.12.1 rcvd UPDATE w/ attr: nexthop 12.12.12.1, origin i,
metric 0, path 1
00:04:18: BGP(0): 12.12.12.1 rcvd 172.16.1.0/24 -- DENIED due to: route-map;
00:04:18: BGP(0): no valid path for 172.16.1.0/24
00:04:18: BGP(0): 12.12.12.1 rcvd 172.16.2.0/24 -- DENIED due to: route-map;
00:04:18: BGP(0): no valid path for 172.16.2.0/24
00:04:18: BGP(0): 12.12.12.1 rcvd 192.168.1.0/24 -- DENIED due to: route-map;
00:04:18: BGP(0): no valid path for 192.168.1.0/24
00:04:18: BGP(0): 12.12.12.1 rcvd 192.168.2.0/24 -- DENIED due to: route-map;
00:04:18: BGP(0): no valid path for 192.168.2.0/24
00:04:18: BGP(0): nettable_walker 172.16.1.0/24 no best path
00:04:18: BGP(0): nettable_walker 172.16.2.0/24 no best path
00:04:18: BGP(0): nettable_walker 192.168.1.0/24 no best path
00:04:18: BGP(0): nettable_walker 192.168.2.0/24 no best path
00:04:21: BGP(0): 12.12.12.1 computing updates, afi 0, neighbor version 21,
table version 25, starting at 0.0.0.0
00:04:21: BGP(0): 12.12.12.1 update run completed, afi 0, ran for 4ms, neighbor
version 21, start version 25, throttled to 25
RT2(config)#
RT2(config)#do sh ip bgp

RT2(config)#

```

- All the routes are denied. In order to have only 172.16.1.0/24 to be denied by the route map, an empty **route-map permit** statement that acts like a **permit any** is required for other routes.

```

RT2(config)#route-map map01 permit 20
RT2(config-route-map)#exit
RT2(config)#
RT2(config)#do clear ip bgp 12.12.12.1 in
RT2(config)#
00:04:47: BGP(0): 12.12.12.1 rcvd UPDATE w/ attr: nexthop 12.12.12.1, origin i,
metric 0, path 1
00:04:47: BGP(0): 12.12.12.1 rcvd 172.16.1.0/24 -- DENIED due to: route-map;
00:04:47: BGP(0): 12.12.12.1 rcvd 172.16.2.0/24
00:04:47: BGP(0): 12.12.12.1 rcvd 192.168.1.0/24
00:04:47: BGP(0): 12.12.12.1 rcvd 192.168.2.0/24
00:04:47: BGP(0): Revise route installing 1 of 1 route for 172.16.2.0/24 ->
12.12.12.1 to main IP table
00:04:47: BGP(0): Revise route installing 1 of 1 route for 192.168.1.0/24 ->
12.12.12.1 to main IP table
00:04:47: BGP(0): Revise route installing 1 of 1 route for 192.168.2.0/24 ->
12.12.12.1 to main IP table
00:04:47: BGP(0): 12.12.12.1 computing updates, afi 0, neighbor version 25,
table version 28, starting at 0.0.0.0
00:04:47: BGP(0): 12.12.12.1 update run completed, afi 0, ran for 4ms, neighbor
version 25, start version 28, throttled to 28
RT2(config)#
RT2(config)#do sh ip bgp
BGP table version is 28, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.2.0/24    12.12.12.1         0           0 1 i
*> 192.168.1.0      12.12.12.1         0           0 1 i
*> 192.168.2.0      12.12.12.1         0           0 1 i
RT2(config)#

```

- The last scenario is implementing a **deny – deny** and becomes a **permit** for 172.16.1.0/24! ☺

```

RT2(config)#no access-list 1
RT2(config)#access-list 1 deny 172.16.1.0 0.0.0.255
RT2(config)#access-list 1 permit any
RT2(config)#do clear ip bgp 12.12.12.1 in
RT2(config)#
00:05:15: BGP(0): 12.12.12.1 rcvd UPDATE w/ attr: nexthop 12.12.12.1, origin i,
metric 0, path 1
00:05:15: BGP(0): 12.12.12.1 rcvd 172.16.1.0/24
00:05:15: BGP(0): 12.12.12.1 rcvd 172.16.2.0/24 -- DENIED due to: route-map;
00:05:15: BGP(0): no valid path for 172.16.2.0/24
00:05:15: BGP(0): 12.12.12.1 rcvd 192.168.1.0/24 -- DENIED due to: route-map;
00:05:15: BGP(0): no valid path for 192.168.1.0/24
00:05:15: BGP(0): 12.12.12.1 rcvd 192.168.2.0/24 -- DENIED due to: route-map;
00:05:15: BGP(0): no valid path for 192.168.2.0/24
00:05:15: BGP(0): Revise route installing 1 of 1 route for 172.16.1.0/24 ->
12.12.12.1 to main IP table
00:05:15: BGP(0): nettable_walker 172.16.2.0/24 no best path
00:05:15: BGP(0): nettable_walker 192.168.1.0/24 no best path
00:05:15: BGP(0): nettable_walker 192.168.2.0/24 no best path
00:05:15: BGP(0): 12.12.12.1 computing updates, afi 0, neighbor version 28,
table version 32, starting at 0.0.0.0
00:05:15: BGP(0): 12.12.12.1 update run completed, afi 0, ran for 4ms, neighbor
version 28, start version 32, throttled to 32
RT2(config)#
RT2(config)#do sh ip bgp
BGP table version is 32, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
* > 172.16.1.0/24    12.12.12.1         0           0 1 i
RT2(config)#
RT2(config)#do sh route-map
route-map map01, deny, sequence 10
  Match clauses:
    ip address (access-lists): 1
  Set clauses:
    Policy routing matches: 0 packets, 0 bytes
route-map map01, permit, sequence 20
  Match clauses:
    Set clauses:
    Policy routing matches: 0 packets, 0 bytes
RT2(config)#
RT2(config)#do sh access-list 1
Standard IP access list 1
  10 deny 172.16.1.0, wildcard bits 0.0.0.255 (1 match)
  20 permit any (3 matches)
RT2(config)#

```



## BGP Route Dampening

Route dampening provides a mechanism to control route instability due to route flapping which continuously generates BGP UPDATE and WITHDRAWN messages on the internetwork. The amount of routing updates can consume considerable network bandwidth and router resources. A route flap occurs when it transitions from the up to the down state – when the prefix is withdrawn.

Routes are categorized as either **well-behaved** or **ill-behaved**. A well-behaved route shows a high level of stability over an extended period of time; while an ill-behaved route experiences a high level of instability in a short period of time. An ill-behaved unstable route is penalized and suppressed (not being advertised) until there is some level of confidence that the route has become stable.

**Note:** Route dampening applies upon EBGp routes only.

The recent history of a route is used for estimating future stability. Dampening tracks the number of times a route has flapped over a period of time. A route is assigned a **penalty** upon each time it flaps. When the penalty reaches a predefined threshold – the **suppress limit**, the route is suppressed. A route can continue to accumulate penalties even after it is suppressed. The more frequent a route flaps in a short period of time, the faster the route is suppressed.

An algorithm is implemented to decay or reduce the penalty value exponentially in order to unsuppress a route and start readvertising it again. The algorithm bases its configuration on a user-defined set of parameters as below:

<b>Half-life</b>	A numeric value that describes the amount of time that must elapse to reduce the penalty by one half. A longer half-life might be desirable for a route that has a habit of oscillating frequently. A larger half-life value would cause the penalty to decay more slowly, in which a route is being suppressed longer.
<b>Suppress limit</b>	A numeric value that is compared with the penalty. If the penalty is greater than the suppress limit, the route is suppressed.
<b>Reuse limit</b>	A numeric value that is compared with the penalty. If the penalty is less than the reuse limit, a suppressed route that is up will no longer be suppressed.

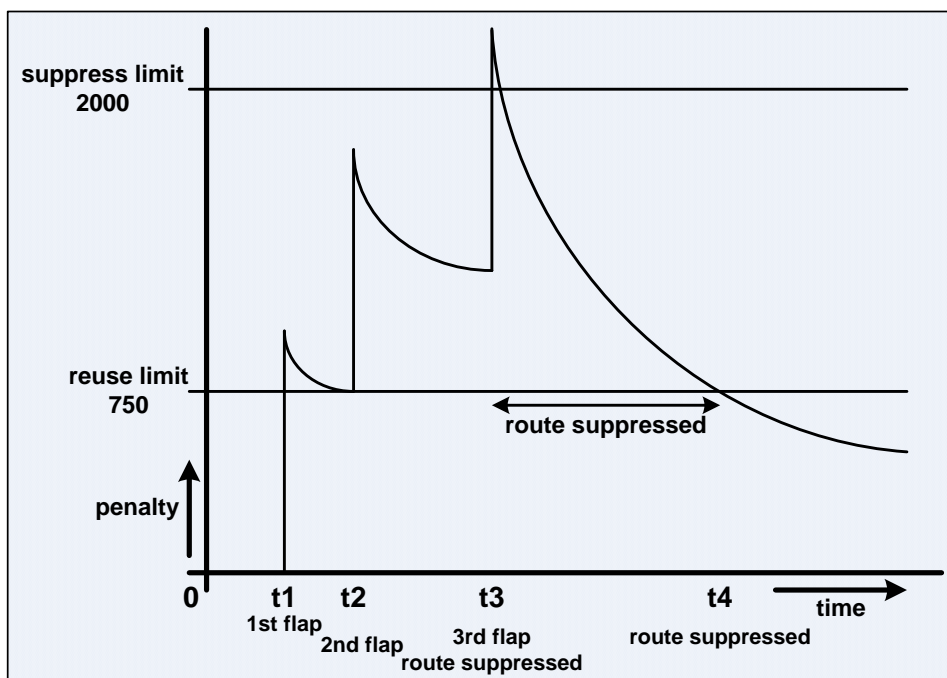


Figure A6-51: BGP Route Dampening Penalty Assessment

When BGP is redistributed or injected into an IGP and/or BGP, it is important to ensure that EBGp instability does not affect internal routing and causes a meltdown inside the AS. Flapping routes will be suppressed and prevented from being injected into the AS until they have a level of stability. **Note:** Route dampening is only applicable upon the routes received from EBGp peers; it has no effect upon the external routes received from IBGP peers.

Route dampening is often implemented in ISP environments to shield the instabilities that occur inside a customer network from burdening the provider network and the outside world – the Internet. This is not an issue when a provider advertises a customer network as part of an aggregate, which is stable and always advertised even if most of its component more-specific routes are not. When a customer network cannot be aggregated due to multi-homing or it is not being part of the address space of the provider, instabilities will be carried to the outside world.

A possible side effect of route dampening in ISP environments is that a customer will experience some short outages even if his routes have become stable. If administrators are unaware that their routes are being dampened and caused some subnets to be unreachable from the outside world, they might try to resolve the problem by troubleshooting the IGP, resetting BGP sessions, etc; and makes their routes flap even more and become more penalized. The better approach is to contact the provider whether he is receiving the routes, and if he is, check why they are not being advertised. Providers have strict policies and might not change the dampening behavior as per customer request. What the provider can do is flush the history info of the dampened routes to advertise the routes.

The **bgp dampening** [*half-life reuse suppress max-suppress-time* | **route-map** *map-name*] BGP router subcommand enables BGP route dampening and/or modifies it parameters. The *max-suppress-time* indicates the maximum of time in minutes a route can be suppressed; when the *max-suppress-time* is configured, the maximum penalty will never be exceeded, regardless of the number of times that the route dampens. The maximum penalty is computed with the following formula.

$$max\ penalty = reuse\ limit \times 2^{\frac{max-suppress-time}{half-life}}$$

The Cisco defaults for the various route dampening variables are as below:

- **Penalty** – 1000 per flap
- **Suppress limit** – 2000
- **Reuse limit** – 750
- **Half-life** – 15 minutes
- **Maximum suppress time** – 60 minutes, or 4 times the half-time (4 x half-time)

```
Router#sh ip bgp dampening parameters
dampening 15 750 2000 60 (DEFAULT)
  Half-life time      : 15 mins      Decay Time           : 2320 secs
  Max suppress penalty: 12000        Max suppress time    : 60 mins
  Suppress penalty    : 2000         Reuse penalty        : 750

Router#
```

The process of reducing the penalty happens every 5 seconds. The process of unsuppressing routes happens every 10 seconds.

A route map can be associated with route dampening to selectively apply the dampening parameters if certain criteria are found, eg: matching upon a specific IP prefix, AS\_PATH, or community.

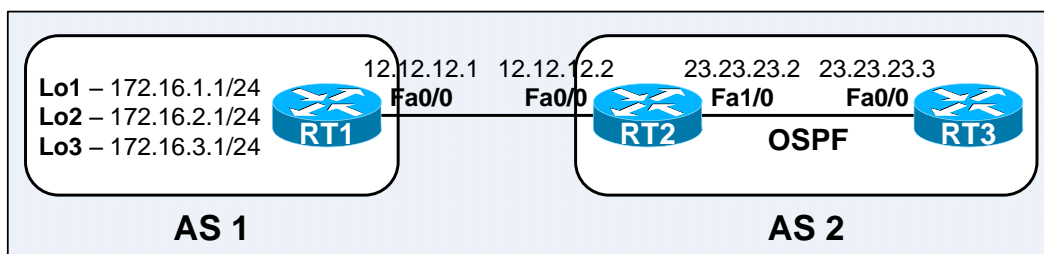


Figure A6-52: Network Setup for BGP Route Dampening

Below implements BGP route dampening with a route map called SELECTIVE\_DAMPENING to apply the dampening upon 172.16.2.0/24 only. All other routes will not be dampened upon flapping.

```

RT2#sh ip bgp
BGP table version is 4, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop        Metric LocPrf Weight Path
*> 172.16.1.0/24    12.12.12.1         0         0 1 i
*> 172.16.2.0/24    12.12.12.1         0         0 1 i
*> 172.16.3.0/24    12.12.12.1         0         0 1 i
RT2#
RT2#debug ip bgp dampening ?
  <1-199>           Access list
  <1300-2699>       Access list (expanded range)
  <cr>

RT2#debug ip bgp dampening 1
BGP dampening debugging is on for access list 1
RT2#
RT2#conf t
Enter configuration commands, one per line. End with CNTL/Z.
RT2(config)#access-list 1 permit 172.16.2.0 0.0.0.255
RT2(config)#
RT2(config)#ip prefix-list dampening-route permit 172.16.2.0/24
RT2(config)#route-map SELECTIVE_DAMPENING permit 10
RT2(config-route-map)#match ip address prefix-list dampening-route
RT2(config-route-map)#set dampening 15 750 2000 60
RT2(config-route-map)#
RT2(config-route-map)#router bgp 2
RT2(config-router)#bgp dampening route-map SELECTIVE_DAMPENING
RT2(config-router)#end
RT2#
00:01:49: BGP(0): Created dampening structures with halflife time 15,
reuse/suppress 750/2000
RT2#

```

**Note:** the route-map SELECTIVE\_DAMPENING permit 20 is not required.

When BGP receives a withdrawn for a prefix, BGP considers the withdrawn prefix as a flap and increases the penalty by 1000; if BGP receives an attribute change, BGP increases the penalty by 500. BGP keeps the withdrawn prefix in the BGP table as a history entry. Below shows the 1st flap.

```
RT2#! RT1 shuts Lo2
00:02:47: EvD: charge penalty 1000, new accum. penalty 1000, flap count 1
00:02:47: BGP(0): charge penalty for 172.16.2.0/24 path 1 with halflife-time 15
reuse/suppress 750/2000
00:02:47: BGP(0): flapped 1 times since 00:00:00. New penalty is 1000
RT2#sh ip bgp 172.16.2.0
BGP routing table entry for 172.16.2.0/24, version 5
Paths: (1 available, no best path)
  Not advertised to any peer
    1 (history entry)
      12.12.12.1 from 12.12.12.1 (1.1.1.1)
        Origin IGP, metric 0, localpref 100, external
        Dampinfo: penalty 1000, flapped 1 times in 00:00:03
RT2#
00:02:49: EvD: accum. penalty decayed to 1000 after 2 second(s)
00:02:50: EvD: accum. penalty decayed to 1000 after 1 second(s)
RT2#
```

Below shows the 2nd flap.

```
RT2#! RT1 no shuts Lo2
00:03:16: EvD: accum. penalty 980, not suppressed
RT2#sh ip bgp 172.16.2.0
BGP routing table entry for 172.16.2.0/24, version 6
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
  Not advertised to any peer
    1
      12.12.12.1 from 12.12.12.1 (1.1.1.1)
        Origin IGP, metric 0, localpref 100, valid, external, best
        Dampinfo: penalty 976, flapped 1 times in 00:00:34
RT2#
00:03:21: EvD: accum. penalty decayed to 976 after 5 second(s)
RT2#! RT1 shuts Lo2
00:03:44: EvD: accum. penalty decayed to 961 after 23 second(s)
00:03:44: EvD: charge penalty 1000, new accum. penalty 1961, flap count 2
00:03:44: BGP(0): charge penalty for 172.16.2.0/24 path 1 with halflife-time 15
reuse/suppress 750/2000
00:03:44: BGP(0): flapped 2 times since 00:00:57. New penalty is 1961
RT2#sh ip bgp 172.16.2.0
BGP routing table entry for 172.16.2.0/24, version 7
Paths: (1 available, no best path)
  Not advertised to any peer
    1 (history entry)
      12.12.12.1 from 12.12.12.1 (1.1.1.1)
        Origin IGP, metric 0, localpref 100, external
        Dampinfo: penalty 1953, flapped 2 times in 00:01:02
RT2#
00:03:49: EvD: accum. penalty decayed to 1953 after 5 second(s)
00:03:49: EvD: accum. penalty decayed to 1953 after 0 second(s)
RT2#
```

Below shows the 3rd flap.

```
RT2#! RT1 no shuts Lo2
00:04:12: EvD: accum. penalty 1923, not suppressed
RT2#sh ip bgp 172.16.2.0
BGP routing table entry for 172.16.2.0/24, version 8
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
    Not advertised to any peer
    1
      12.12.12.1 from 12.12.12.1 (1.1.1.1)
        Origin IGP, metric 0, localpref 100, valid, external, best
        Dampinfo: penalty 1923, flapped 2 times in 00:01:29
RT2#
00:04:16: EvD: accum. penalty decayed to 1923 after 4 second(s)
RT2#! RT1 shuts Lo2
00:04:41: EvD: accum. penalty decayed to 1886 after 25 second(s)
00:04:41: EvD: charge penalty 1000, new accum. penalty 2886, flap count 3
00:04:41: BGP(0): charge penalty for 172.16.2.0/24 path 1 with halflife-time 15
reuse/suppress 750/2000
00:04:41: BGP(0): flapped 3 times since 00:01:54. New penalty is 2886
RT2#sh ip bgp 172.16.2.0
BGP routing table entry for 172.16.2.0/24, version 9
Paths: (1 available, no best path)
Flag: 0x820
    Not advertised to any peer
    1 (history entry)
      12.12.12.1 from 12.12.12.1 (1.1.1.1)
        Origin IGP, metric 0, localpref 100, external
        Dampinfo: penalty 2874, flapped 3 times in 00:01:59
RT2#
00:04:46: EvD: accum. penalty decayed to 2874 after 5 second(s)
00:04:49: EvD: accum. penalty decayed to 2874 after 3 second(s)
RT2#
```

Below shows that 172.16.2.0/24 is dampened after it comes back up again after the 3rd flap. Dampened prefixes are not used in the BGP decision process and not installed into the routing table.

```

RT2#! RT1 no shuts Lo2
00:05:11: BGP(0): suppress 172.16.2.0/24 path 1 for 00:28:40 (penalty 2829)
00:05:11: halflife-time 15, reuse/suppress 750/2000
00:05:11: EvD: accum. penalty 2829, now suppressed with a reuse intervals of 172
RT2#sh ip bgp 172.16.2.0
BGP routing table entry for 172.16.2.0/24, version 9
Paths: (1 available, no best path)
Not advertised to any peer
  1, (suppressed due to dampening)
    12.12.12.1 from 12.12.12.1 (1.1.1.1)
      Origin IGP, metric 0, localpref 100, valid, external
      Dampinfo: penalty 2818, flapped 3 times in 00:02:30, reuse in 00:06:49
RT2#
00:05:17: EvD: accum. penalty decayed to 2818 after 6 second(s)
RT2#sh ip bgp dampening dampened-paths
BGP table version is 9, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          From           Reuse      Path
*d 172.16.2.0/24    12.12.12.1     00:06:39  1 i
RT2#
00:05:27: EvD: accum. penalty decayed to 2796 after 10 second(s)
RT2#sh ip bgp dampening flap-statistics
BGP table version is 9, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          From           Flaps Duration Reuse      Path
*d 172.16.2.0/24    12.12.12.1     3      00:02:48 00:06:29  1
RT2#
00:05:35: EvD: accum. penalty decayed to 2785 after 8 second(s)
RT2#

```

Below shows that the route is unsuppressed around 2 round of half-life (2 x 15 minutes, total 30 minutes) to reduce the penalty from 3000 → 1500, and then from 1500 → 750.

```

RT2#
00:34:01: EvD: accum. penalty decayed to 749 after 98 second(s)
00:34:01: EvD: accum. penalty 749, now unsuppressed
00:34:01: BGP(0): Unsuppressed 172.16.2.0/24, path 1
RT2#sh ip bgp 172.16.2.0
BGP routing table entry for 172.16.2.0/24, version 10
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
Not advertised to any peer
  1
    12.12.12.1 from 12.12.12.1 (1.1.1.1)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Dampinfo: penalty 746, flapped 3 times in 00:31:20
RT2#

```

The **history event** is an entry used to store route flap information that is important for monitoring and calculating the oscillation level of a route. When the route stabilizes, the history event becomes useless and must be flushed from the router using the **clear ip bgp dampening** privileged command.

## BGP Outbound Route Filtering (ORF)

The BGP Outbound Route Filtering capability conserves network bandwidth and router resources upon the BGP route update process by minimizing the number of BGP updates sent to a neighbor. A BGP router will allow its neighbor to send over its inbound prefix filter, which is then installed in addition to any outbound filter configured for the neighbor.

- ❖ The inbound filter list can be sent to the peer and installed as its outbound filter

Below are the main benefits of implementing BGP ORF:

- A BGP router will no longer consume resources generating routing updates that will be filtered by the neighbor.
- A BGP router will not need to consume resources to process unwanted incoming routing updates and filter out prefixes based on the inbound filter implemented for the neighbor. A low-end router with limited memory can be overwhelmed (run out of memory) by large number of BGP updates even if it has the inbound filters configured.
- Reduce the administration overhead on a provider upon handling the constant change requests from customers for modifying the outbound filters to determine the prefixes that the customers would like to and don't want to receive. Likewise from the perspective of the customers, this is the optimal administrative design, as they don't need to send change requests to the provider, and can change their filtering design anytime.

The BGP ORF feature is disabled by default. It is enabled by advertising ORF capability to peers. BGP ORF is asymmetric in nature and can be configured in one direction between 2 routers, with one end configured to send ORF capability and another end configured to receive ORF capability. It can be independently configured with **send**, **receive**, or **both** ORF capabilities. The ORF capability is exchanged during session establishment.

The local peer that advertises the support for BGP ORF in send mode will send its inbound filter only if it receives the receive mode BGP ORF capability from the remote peer. However, the remote peer will not send the first update until it receives a ROUTE-REFRESH Request message with the IMMEDIATE ORF flag along with the ORF prefix list from the local peer. The local peer will continue to apply the locally defined inbound prefix list filter upon the received updates. BGP updates are exchanged between peer routers for each address family depending on the BGP ORF capability advertised.

A sample network with RT1 in AS1 and RT2 in AS2 is used to demonstrate the usage of BGP ORF. The BGP table on RT2 below lists the routes that received from RT1.

```
RT2#sh ip bgp
BGP table version is 6, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network        Next Hop        Metric LocPrf Weight Path
*> 0.0.0.0        12.12.12.1      0           0 1 ?
*> 172.16.1.0/24  12.12.12.1      0           0 1 i
*> 192.168.1.0    12.12.12.1      0           0 1 i
*> 192.168.2.0    12.12.12.1      0           0 1 i
*> 192.168.3.0    12.12.12.1      0           0 1 i
RT2#
```

Note that advertising a default route via BGP requires the following 3 steps:

- Create a static default route using the **ip route 0.0.0.0 0.0.0.0 {ip-addr | out-intf}** command.
- Redistribute static into BGP using the **redistribute static** BGP router subcommand.
- Configure the **default-information originate** BGP router subcommand.

Suppose that AS2 would like to only receive the default route 0.0.0.0/0 and 172.16.1.0/24 from AS1. Below shows the result of implementing traditional filtering using prefix list. The filtering goal is achieved, but with inefficiency; as the unwanted routes actually being sent from AS1 and filtered in AS2, wasting the network bandwidth and router resources for generating and processing them.

```
RT2#conf t
Enter configuration commands, one per line. End with CNTL/Z.
RT2(config)#ip prefix-list AS1-Inbound permit 0.0.0.0/0
RT2(config)#ip prefix-list AS1-Inbound permit 172.16.1.0/24
RT2(config)#router bgp 2
RT2(config-router)#neighbor 12.12.12.1 prefix-list AS1-Inbound in
RT2(config-router)#end
RT2#
RT2#debug ip bgp updates
BGP updates debugging is on
RT2#
00:01:59: BGP(0): no valid path for 0.0.0.0/0
00:01:59: BGP(0): no valid path for 172.16.1.0/24
00:01:59: BGP(0): no valid path for 192.168.1.0/24
00:01:59: BGP(0): no valid path for 192.168.2.0/24
00:01:59: BGP(0): no valid path for 192.168.3.0/24
00:01:59: %BGP-5-ADJCHANGE: neighbor 12.12.12.1 Down User reset
00:01:59: BGP(0): nettable_walker 0.0.0.0/0 no best path
00:01:59: BGP(0): nettable_walker 172.16.1.0/24 no best path
00:01:59: BGP(0): nettable_walker 192.168.1.0/24 no best path
00:01:59: BGP(0): nettable_walker 192.168.2.0/24 no best path
00:01:59: BGP(0): nettable_walker 192.168.3.0/24 no best path
00:02:22: %BGP-5-ADJCHANGE: neighbor 12.12.12.1 Up
00:02:22: BGP(0): 12.12.12.1 computing updates, afi 0, neighbor version 0, table
version 11, starting at 0.0.0.0
00:02:22: BGP(0): 12.12.12.1 update run completed, afi 0, ran for 4ms, neighbor
version 0, start version 11, throttled to 11
00:02:22: BGP(0): 12.12.12.1 initial update completed
00:02:23: BGP(0): 12.12.12.1 rcvd UPDATE w/ attr: nexthop 12.12.12.1, origin ?,
metric 0, path 1
00:02:23: BGP(0): 12.12.12.1 rcvd 0.0.0.0/0
00:02:23: BGP(0): 12.12.12.1 rcvd UPDATE w/ attr: nexthop 12.12.12.1, origin i,
metric 0, path 1
00:02:23: BGP(0): 12.12.12.1 rcvd 172.16.1.0/24
00:02:23: BGP(0): 12.12.12.1 rcvd 192.168.1.0/24 -- DENIED due to:
distribute/prefix-list;
00:02:23: BGP(0): 12.12.12.1 rcvd 192.168.2.0/24 -- DENIED due to:
distribute/prefix-list;
00:02:23: BGP(0): 12.12.12.1 rcvd 192.168.3.0/24 -- DENIED due to:
distribute/prefix-list;
00:02:23: BGP(0): Revise route installing 1 of 1 route for 0.0.0.0/0 ->
12.12.12.1 to main IP table
00:02:23: BGP(0): Revise route installing 1 of 1 route for 172.16.1.0/24 ->
12.12.12.1 to main IP table
RT2#
RT2#sh ip bgp
BGP table version is 13, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 0.0.0.0          12.12.12.1         0           0 1 ?
*> 172.16.1.0/24    12.12.12.1         0           0 1 i
RT2#
```



Without BGP ORF, the upstream router RT1 sent the full BGP table to the downstream router RT2, and filtering was applied inbound on the downstream router. With BGP ORF, the downstream router dynamically informs the upstream router the routes to filter outbound; and that the downstream router will only receive update messages based upon the prefixes that it wants.

Below shows the implementation of the BGP ORF send mode on RT2 and receive mode on RT1.

The **neighbor capability orf prefix-list {receive | send | both}** address family submode command can be configured and take effect under the BGP router configuration mode, but the command will only be displayed under the address family section.

Note that configuring this command will terminate the established BGP session with the neighbor.

The output of the **debug ip bgp** privileged command shows the negotiation of the BGP ORF capability during initial BGP peering session establishment.

```
RT1 (AS1) :
!
router bgp 1
  bgp router-id 1.1.1.1
  bgp log-neighbor-changes
  neighbor 12.12.12.2 remote-as 2
  !
  address-family ipv4
    redistribute static
    neighbor 12.12.12.2 activate
    neighbor 12.12.12.2 capability orf prefix-list receive
    default-information originate
    no auto-summary
    no synchronization
    network 172.16.1.0 mask 255.255.255.0
    network 192.168.1.0
    network 192.168.2.0
    network 192.168.3.0
  exit-address-family
!
=====
RT2 (AS2) :
!
router bgp 2
  bgp router-id 2.2.2.2
  bgp log-neighbor-changes
  neighbor 12.12.12.1 remote-as 1
  !
  address-family ipv4
    neighbor 12.12.12.1 activate
    neighbor 12.12.12.1 capability orf prefix-list send
    neighbor 12.12.12.1 prefix-list AS1-Inbound in
    no auto-summary
    no synchronization
  exit-address-family
!
ip prefix-list AS1-Inbound seq 5 permit 0.0.0.0/0
ip prefix-list AS1-Inbound seq 10 permit 172.16.1.0/24
!
```

BGP ORF is defined as a new capability with the Capability Code of 130, BGP ORF-Type 128, with Receive-mode (value 1), Send-mode (value 2), or Both (value 3) in the BGP OPEN messages. The **send** keyword is supported for individual neighbors, peer group members, or the peer group itself; the **receive** and **both** keywords are not supported for peer group members and peer groups.

Below shows the excerpt of the **show ip bgp neighbors** command on RT1 and RT2 regarding BGP ORF.

```
RT1#sh ip bgp neighbors
--- output omitted ---
For address family: IPv4 Unicast
BGP table version 6, neighbor version 6
Index 1, Offset 0, Mask 0x2
AF-dependant capabilities:
  Outbound Route Filter (ORF) type (128) Prefix-list:
    Send-mode: received
    Receive-mode: advertised
Outbound Route Filter (ORF): received (2 entries)

```

	Sent	Rcvd
Prefix activity:	----	----
Prefixes Current:	2	0
Prefixes Total:	2	0
Implicit Withdraw:	0	0
Explicit Withdraw:	0	0
Used as bestpath:	n/a	0
Used as multipath:	n/a	0

	Outbound	Inbound
Local Policy Denied Prefixes:	-----	-----
Total:	0	0

```
Number of NLRIs in the update sent: max 4, min 0
--- output omitted ---
=====
RT2#sh ip bgp neighbors
--- output omitted ---
For address family: IPv4 Unicast
BGP table version 17, neighbor version 17
Index 1, Offset 0, Mask 0x2
AF-dependant capabilities:
  Outbound Route Filter (ORF) type (128) Prefix-list:
    Send-mode: advertised
    Receive-mode: received
Outbound Route Filter (ORF): sent;
Incoming update prefix filter list is AS1-Inbound

```

	Sent	Rcvd
Prefix activity:	----	----
Prefixes Current:	0	2 (Consumes 96 bytes)
Prefixes Total:	0	2
Implicit Withdraw:	0	0
Explicit Withdraw:	0	0
Used as bestpath:	n/a	2
Used as multipath:	n/a	0

	Outbound	Inbound
Local Policy Denied Prefixes:	-----	-----
Bestpath from this peer:	2	n/a
Total:	2	0

```
Number of NLRIs in the update sent: max 0, min 0
--- output omitted ---
```

BGP ORF support is available for all address families, eg: IPv4 unicast, IPv4 multicast, etc.

Below shows that RT1 received the inbound prefix filter of RT2.  
 Note that the prefix list does not show up locally in the running configuration.  
 RT1 then uses the prefix list as an outbound filter towards RT2 and sends only the interested routes.

```

RT1#sh ip bgp neighbors 12.12.12.2 received prefix-filter
Address family: IPv4 Unicast
ip prefix-list 12.12.12.2: 2 entries
    seq 5 permit 0.0.0.0/0
    seq 10 permit 172.16.1.0/24
RT1#
RT1#sh ip prefix-list

RT1#
RT1#sh ip bgp neighbors 12.12.12.2 advertised-routes
BGP table version is 6, local router ID is 1.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 0.0.0.0          0.0.0.0           0             32768 ?
*> 172.16.1.0/24   0.0.0.0           0             32768 i
RT1#
  
```

Below shows how RT2 dynamically requests RT1 to defer the old ORF prefix list and sends the new ORF prefix list when requesting RT1 to send an additional prefix – 192.168.1.0/24.

```

RT2#conf t
Enter configuration commands, one per line. End with CNTL/Z.
RT2(config)#ip prefix-list AS1-Inbound permit 192.168.1.0/24
RT2(config)#end
RT2#
RT2#clear ip bgp 12.12.12.1 in prefix-filter
RT2#
00:09:25: BGP: 12.12.12.1 sending REFRESH_REQ to remove orftype 128 (defer) for
afi/safi: 1/1
00:09:25: BGP: 12.12.12.1 sending REFRESH_REQ with pfxlist ORF for IPv4 Unicast
00:09:25: BGP(0): 12.12.12.1 rcvd UPDATE w/ attr: nexthop 12.12.12.1, origin ?,
metric 0, path 1
00:09:25: BGP(0): 12.12.12.1 rcvd 0.0.0.0/0...duplicate ignored
00:09:25: BGP(0): 12.12.12.1 rcvd UPDATE w/ attr: nexthop 12.12.12.1, origin i,
metric 0, path 1
00:09:25: BGP(0): 12.12.12.1 rcvd 172.16.1.0/24...duplicate ignored
00:09:25: BGP(0): 12.12.12.1 rcvd 192.168.1.0/24
00:09:25: BGP(0): Revise route installing 1 of 1 route for 192.168.1.0/24 ->
12.12.12.1 to main IP table
00:09:25: BGP(0): 12.12.12.1 computing updates, afi 0, neighbor version 17,
table version 18, starting at 0.0.0.0
00:09:25: BGP(0): 12.12.12.1 update run completed, afi 0, ran for 0ms, neighbor
version 17, start version 18, throttled to 18
RT2#
RT2#sh ip bgp
BGP table version is 18, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*> 0.0.0.0          12.12.12.1        0             0 1 ?
*> 172.16.1.0/24   12.12.12.1        0             0 1 i
*> 192.168.1.0     12.12.12.1        0             0 1 i
RT2#
  
```

When the inbound prefix list changes or being removed, the **clear ip bgp {ip-addr} in prefix-filter** privileged command can be used to send out the prefix list and receive route refresh from the neighbor based on the new prefix list. The **prefix-filter** keyword will be ignored if the receive-orf capability has not been received from the neighbor, or the local router has not sent the send-orf for the neighbor.

Without the **prefix-filter** keyword, the **clear ip bgp {ip-addr} in** would simply perform the normal route refresh with the neighbor – does not send out the inbound prefix filter to the neighbor. This command is useful when using inbound routing policies other than the inbound prefix list filter such as a route map change.

## Multiprotocol BGP (MBGP)

Multiprotocol BGP (MBGP) defines backward-compatible extensions upon BGP-4 that allow it to exchange reachability information for multiple network layer protocols or address families, eg: IPv4, IPv6, and sub-AFs, eg: unicast or multicast. Individual network layer protocols are identified by an Address Family (AF) as defined in RFC 1700 – Assigned Numbers.

The initial Cisco IOS CLI design that focus on accommodating only a single AF – IPv4, and its associated sub-AFs – unicast and multicast, introduced unnecessary management and policy expression complexities to provide support for additional AFs. A new approach that cleanly separates general session-related parameters from AF-specific parameters was then implemented. The new approach provides several advantages beyond the management aspects alone:

- Inbound and outbound policies can be different for each session on specific AF.
- A BGP router can be configured as a route reflector for a single AF or multiple AFs.
- No new configuration is required to support vanilla BGP – IPv4 unicast.
- Prefixes can be sourced within any AF independently via **network** statements, **aggregate-address** statements, and redistribution.
- Peer group functionality will be maintained within the associated AF, as it is relevant for the generation of UPDATE messages.

The new CLI model is much more flexible when using multiple AFs. The BGP router subcommands were divided into 3 command groups:

- **Global BGP commands** – commands that affect the operation of BGP global to the router, eg: **bgp deterministic-med**, **bgp cluster-id**, etc. There is no ambiguity for this group of commands; these commands appear only once in the configuration.
- **Commands to identify the neighbors or peer groups** – commands that define the neighbor or peer group that is accessible from the default routing table by specifying session parameters, eg: **neighbor {ip-addr} remote-as {asn}**, **neighbor {ip-addr} ebgp-multihop {ttl}**, etc. There is no ambiguity for this group of commands; these commands follow just after the global BGP commands. BGP neighbors are defined once in the configuration, with exception to VRF.
- **Commands per-address family**. 2 sets of commands can be per-AF.
  - ❖ **Global to per-AF** – commands that are neighbor-independent and modify the behavior of BGP for a specific AF, eg: **bgp scan-time {interval}**, and the **network**, **aggregate-address**, and **redistribute** commands that source prefixes for a particular AF belong to this category.
  - ❖ **AF-specific per-neighbor / peer group** – commands that configure policy for the neighbor(s) or peer groups with distribute lists, prefix lists, or route maps. The neighbor can also be configured as a route reflector client or peer group member. The neighbor must be explicitly **activated** to enable the exchange of MBGP prefixes. **Ex: neighbor {ip-addr} filter-list in, neighbor {peer-group-name} route-map {map-name} in, neighbor {ip-addr} activate**, etc.

The new **address-family** submode was introduced under **router bgp {asn}** configuration mode for implementing the per-AF commands. Below shows a sample MBGP configuration.

```

router bgp 1
  bgp log-neighbor-changes
  bgp deterministic-med ! Global to BGP
  bgp bestpath med confed ! Global to BGP
  neighbor EBGP-PEERS peer-group ! Peer group definition; Global to BGP
  neighbor 1.1.1.1 remote-as 1 ! Neighbor definition; Global to BGP
  neighbor 2.2.2.2 remote-as 2 ! Neighbor definition; Global to BGP
  neighbor 2.2.2.2 peer-group EBGP-PEERS ! Neighbor membership; Global to BGP
  neighbor 3.3.3.3 remote-as 3 ! Neighbor definition; Global to BGP
  neighbor 3.3.3.3 peer-group EBGP-PEERS ! Neighbor membership; Global to BGP
  !
  address-family ipv4 ! address-family IPv4-unicast sub-mode
    no synchronization ! Global to IPv4-unicast
    bgp scan-time 10 ! Global to IPv4-unicast
    network 10.0.0.0 ! Global to IPv4-unicast
    aggregate-address 20.0.0.0 255.0.0.0 ! Global to IPv4-unicast
    neighbor EBGP-PEERS route-map ucast-out out ! IPv4-unicast peer group policy
    neighbor 1.1.1.1 activate ! Activate neighbor for IPv4-unicast
    neighbor 1.1.1.1 route-reflector-client ! IPv4-unicast route reflector client
    neighbor 2.2.2.2 activate ! Activate neighbor for IPv4-unicast
    neighbor 2.2.2.2 route-map ucast-in in ! IPv4-unicast neighbor policy
    neighbor 3.3.3.3 activate ! Activate neighbor for IPv4-unicast
    no auto-summary ! Disable IPv4-unicast auto summarization
  exit-address-family ! Exit AF sub-mode

```

A BGP router can have different policies applied to a single peer, eg: **route-map** and **prefix-list** statements, one per-AF.

It is possible to configure IPv4-unicast BGP (vanilla BGP) policy for neighbors using the old style CLI. Although the **address-family ipv4 unicast** is implicit, explicit configuration is recommended. The IPv4-unicast global (global to per-AF) and policy (AF-specific per-neighbor / peer group) commands are listed within the **address-family ipv4 unicast** section in the **show running-config**. The global to per-AF group commands appear first, followed by the AF-specific per-neighbor / peer group commands.

The **bgp upgrade-cli** BGP router subcommand provides a smooth upgrade path from old-style to AF-style by parsing the old-style commands. Issue **copy running-config startup-config** to save the new configuration. **Note:** The **bgp upgrade-cli** command is not shown in the configuration.

The **activate** command is new to Cisco IOS BGP configuration as part of the AF CLI enhancements. It is used to enable or activate the support of a specific AF for a neighbor. In the old-style CLI, the neighbor was activated for IPv4 BGP automatically. If MBGP was enabled, the **nlri** keyword in the **neighbor** command was required. If the **nlri** keyword was not specified, the router would exchange IPv4 prefixes only. If the **nlri** keyword was specified with the **multicast** option only, the router would exchange IPv4 multicast NLRI only. This command could active only unicast, only multicast, or both.

```
Router(config-router) #neighbor 1.1.1.1 remote-as 1 nlri unicast multicast
```

The **activate** command in the **address-family** submode enables an AF for a neighbor in the AF-style CLI. The neighbors that are defined under the BGP router configuration mode are automatically activated for IPv4. The neighbors must be explicitly activated for all other AFs.

```
Router(config-router) #address-family ipv4 multicast
```

```
Router(config-router-af) #neighbor 1.1.1.1 activate
```

The old-style **network** command used an **nlri** extension to specify if a network was to be advertised as unicast, multicast, or both. The absence of the **nlri** keyword implied IPv4 unicast only.

```
Router(config)#router bgp 1
Router(config-router)#network 1.1.1.0 mask 255.255.255.0
Router(config-router)#network 2.0.0.0 nlri multicast
Router(config-router)#network 3.0.0.0 nlri unicast multicast
```

The **network** command can now be specified under specific AFs in the AF-style CLI to independently announce a unicast or multicast NLRI under specific AFs.

```
address-family ipv4
  network 1.1.1.0 mask 255.255.255.0
  network 3.0.0.0
exit-address-family
!
address-family ipv4 multicast
  network 2.0.0.0
  network 3.0.0.0
exit-address-family
```

With the old-style CLI, a multicast aggregate was configured the same way as a unicast aggregate via the **aggregate-address** command. The **nlri** keyword in the **aggregate-address** command specifies whether the aggregate address should be applied to unicast and/or multicast.

```
router bgp 1
  aggregate-address 10.0.0.0 255.0.0.0 nlri multicast as-set
```

Configuring aggregation for multiple AFs was complex if varying policies were required. The AF-style CLI eliminates the need of the **nlri** keyword in the **aggregate-address** command. The AF mode under which the aggregate is specified determines the unicast or multicast table that the aggregated prefix should be generated.

```
!
router bgp 1
!
address-family ipv4 multicast
  aggregate-address 10.0.0.0 255.0.0.0 as-set
exit-address-family
!
```

The old-style configuration used the **set nlri** clause in the redistribution route map when importing routes from another routing protocol into BGP. The **match nlri** clause in a redistribution route map was being used in the following ways:

```
set nlri
! redistributes the matching prefixes into the unicast table only
set nlri unicast
! redistributes the matching prefixes into the unicast table only
set nlri multicast
! redistributes the matching prefixes into the multicast table only
set nlri unicast multicast
! redistributes the matching prefixes into both the unicast and multicast tables
```

Below shows an example in which all directly connected prefixes that match access list 1 are redistributed as multicast NLRI.

```
!
router bgp 1
  redistribute connected route-map mcast-map
!
route-map mcast-map permit 10
  match ip address 1
  set nlri multicast
!
```

The AF-style of redistribution determines the unicast or multicast table into which the redistributed prefixes are injected. If the **redistribute** resides under the **address-family ipv4 multicast** mode, the redistributed prefixes are injected as IPv4 multicast NLRI. The **set nlri** keyword is no longer required with the new AF-style CLI.

Below shows the AF-style configuration for the preceding old-style configuration:

```
!  
router bgp 1  
!  
  address-family ipv4 multicast  
    redistribute connected route-map mcast-map  
  exit-address-family  
!  
  route-map mcast-map permit 10  
  match ip address 1  
!
```

A peer group was defined in the BGP router configuration mode in the old-style CLI. The **nlri** keyword was used to specify the peer group to exchange unicast and/or multicast prefixes. The peer group members automatically inherited the unicast and/or multicast capability of the peer group.

```
Router(config-router) #neighbor EBGP-PEERS peer-group nlri unicast multicast  
Router(config-router) #neighbor 2.2.2.2 remote-as 2  
Router(config-router) #neighbor 2.2.2.2 peer-group EBGP-PEERS
```

Peer group can now be configured under specific AFs. Peer groups are automatically activated in the specific AFs when the parameters are configured.

With the old-style CLI, route reflector client properties were specified globally, and the configuration applied to all AFs negotiated with its clients. The route reflector knew that it had to reflect routes to and from clients by specifying **route-reflector-client** for a particular IBGP peer or peer group. Below shows an example in which the IBGP peer 1.1.1.1 is a route reflector client for both IPv4 unicast and multicast prefixes.

```
!  
router bgp 1  
  neighbor 1.1.1.1 remote-as 1 nlri unicast multicast  
  neighbor 1.1.1.1 route-reflector-client  
!
```

The AF-style of configuring a route reflector client is AF-dependent and is configured in the AF mode. A peer that is a route reflector client in IPv4 unicast mode does not automatically make it a route reflector client in IPv4 multicast mode. Route reflector configuration is now on a per-AF basis, and the route reflector topologies for different AFs can vary.

Below shows the AF-style configuration for the preceding old-style configuration:

```
!  
router bgp 1  
  neighbor 1.1.1.1 remote-as 1  
  !  
  address-family ipv4  
    neighbor 1.1.1.1 activate  
    neighbor 1.1.1.1 route-reflector-client  
  exit-address-family  
  !  
  address-family ipv4 multicast  
    neighbor 1.1.1.1 activate  
    neighbor 1.1.1.1 route-reflector-client  
  exit-address-family  
!
```

A single route map was used to specify the policies for all AFs in the old-style configuration. The route map was then applied as either inbound or outbound for a peer or peer group. Routing policies relating to the IPv4 unicast and IPv4 multicast address families that could be carried in a BGP session were represented by specifying the **match nlri** clause in the route map sequence. The **match nlri** clause in a route map was being used in the following ways:

```

match nlri                                ! matches only IPv4 unicast
match nlri unicast                        ! matches only IPv4 unicast
match nlri multicast                      ! matches only IPv4 multicast
match nlri unicast multicast             ! matches both IPv4 unicast and multicast

```

The following example shows how to configure BGP to accept multicast routes that match access list 1 and received from neighbor 2.2.2.2:

```

!
router bgp 1
  neighbor 2.2.2.2 remote-as 2 nlri unicast multicast
  neighbor 2.2.2.2 route-map mcast-filter in
!
route-map mcast-filter permit 10
  match nlri multicast
  match ip address 1
!

```

A single route map for expressing all policies was seen as complex, unmanageable, and non-scalable when complicated and differing policies were used for different AFs. The introduction of new configuration mode for each AF facilitated the configuration of policies on a per-AF basis – a separate route map for each AF associated with a neighbor. Not only can route maps be specified on a per-AF basis, but they can also provide AF-specific filtering rules, eg: prefix lists, distribute lists, AS\_PATH access lists, etc. The **match nlri** keyword is no longer required with the new AF-style CLI. The policy can then be expressed in AF-style as follows:

```

!
router bgp 1
  bgp log-neighbor-changes
  neighbor 2.2.2.2 remote-as 2
!
address-family ipv4
  no synchronization
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 route-map ucast-filter in
  no auto-summary
exit-address-family
!
address-family ipv4 multicast
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 route-map mcast-filter in
  no auto-summary
exit-address-family
!
route-map mcast-filter permit 10
  match ip address 2
!
route-map ucast-filter permit 10
  match ip address 1
!

```



Below lists the BGP commands and the category for each command or subcommand:

<b>Command / Subcommand</b>	<b>Category</b>
<b>aggregate-address</b>	Per-AF
<b>auto-summary</b>	Per-AF
<b>bgp always-compare-med</b>	Global to BGP
<b>bgp bestpath</b>	Global to BGP
<b>bgp client-to-client reflection</b>	Global to BGP
<b>bgp cluster-id</b>	Global to BGP
<b>bgp confederation</b>	Global to BGP
<b>bgp dampening</b>	Per-AF
<b>bgp default-metric</b>	Per-AF
<b>bgp deterministic-med</b>	Global to BGP
<b>bgp fast-external-fallover</b>	Global to BGP
<b>bgp log-neighbor-changes</b>	Global to BGP
<b>bgp redistribute-internal</b>	Per-AF
<b>bgp router-id</b>	Global to BGP
<b>bgp scan-time</b>	Per-AF
<b>default</b>	Global to BGP
<b>distance bgp</b>	Per-AF
<b>maximum-paths</b>	Per-AF
<b>neighbor activate</b>	Per-AF
<b>neighbor advertisement-interval</b>	Global to the neighbor (session)
<b>neighbor default-originate</b>	Per-AF (policy)
<b>neighbor description</b>	Global to the neighbor
<b>neighbor distribute-list</b>	Per-AF (policy)
<b>neighbor ebgp-multihop</b>	Global to the neighbor (session)
<b>neighbor filter-list</b>	Per-AF (policy)
<b>neighbor local-as</b>	Global to the neighbor (session)
<b>neighbor maximum-prefix</b>	Per-AF (policy)
<b>neighbor next-hop-self</b>	Global to the neighbor (session)
<b>neighbor password</b>	Global to the neighbor (session)
<b>neighbor peer-group</b>	Per-AF (policy)
<b>neighbor prefix-list</b>	Per-AF (policy)
<b>neighbor remote-as</b>	Global to the neighbor (session)
<b>neighbor remove-private-as</b>	Global to the neighbor (session)
<b>neighbor route-map</b>	Per-AF (policy)
<b>neighbor route-reflector-client</b>	Per-AF
<b>neighbor send-community</b>	Per-AF (policy)
<b>neighbor shutdown</b>	Global to the neighbor (session)
<b>neighbor soft-reconfiguration</b>	Per-AF (policy)
<b>neighbor timers</b>	Global to the neighbor (session)
<b>neighbor update-source</b>	Global to the neighbor (session)
<b>neighbor version</b>	Global to the neighbor (session)
<b>neighbor weight</b>	Per-AF (policy)
<b>network</b>	Per-AF
<b>redistribute</b>	Per-AF
<b>synchronization</b>	Per-AF
<b>table-map</b>	Per-AF
<b>timers bgp</b>	Global to BGP

## GRE Tunnel Recursive Routing

This section illustrates an interesting problem when configuring GRE tunnels and routing protocols.

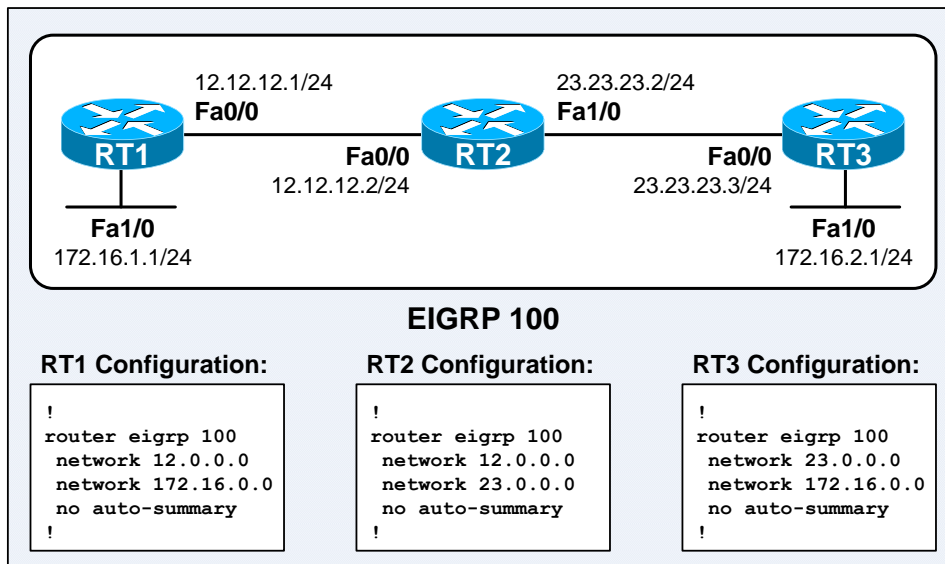


Figure A6-53: A Simple Network

The figure above shows a simple network prior to implementing a GRE tunnel.

Below shows the routing tables on RT1 and RT3:

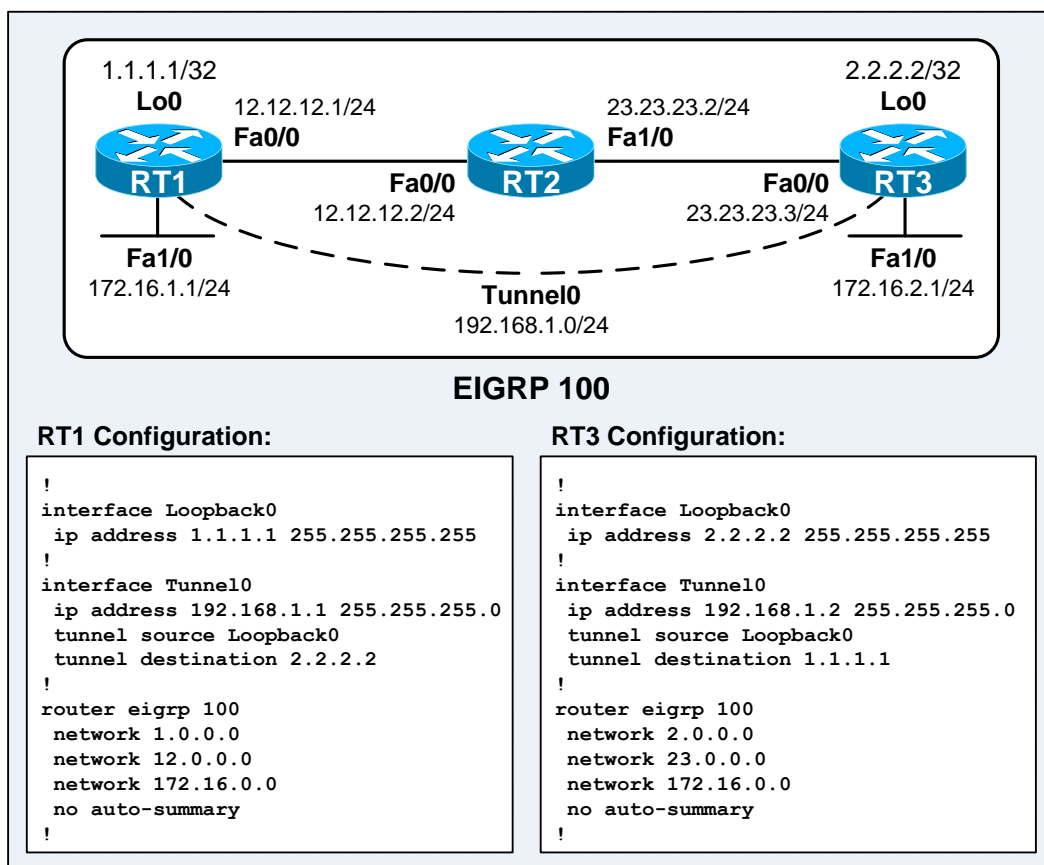
```
RT1#sh ip route

Gateway of last resort is not set

    23.0.0.0/24 is subnetted, 1 subnets
D       23.23.23.0 [90/30720] via 12.12.12.2, 00:00:26, FastEthernet0/0
    172.16.0.0/24 is subnetted, 2 subnets
C       172.16.1.0 is directly connected, FastEthernet1/0
D       172.16.2.0 [90/33280] via 12.12.12.2, 00:00:14, FastEthernet0/0
    12.0.0.0/24 is subnetted, 1 subnets
C       12.12.12.0 is directly connected, FastEthernet0/0
RT1#
=====
RT3#sh ip route

Gateway of last resort is not set

    23.0.0.0/24 is subnetted, 1 subnets
C       23.23.23.0 is directly connected, FastEthernet0/0
    172.16.0.0/24 is subnetted, 2 subnets
D       172.16.1.0 [90/33280] via 23.23.23.2, 00:00:17, FastEthernet0/0
C       172.16.2.0 is directly connected, FastEthernet1/0
    12.0.0.0/24 is subnetted, 1 subnets
D       12.12.12.0 [90/30720] via 23.23.23.2, 00:00:17, FastEthernet0/0
RT3#
```



**Figure A6-54: A Simple GRE Tunnel Network**

The figure above shows how a GRE tunnel between RT1 and RT3 is implemented.

Loopback interfaces are often being preferred to establish GRE tunnels.

Below verifies that the GRE tunnel is operational on RT1:

```

RT1#sh ip route

Gateway of last resort is not set

 1.0.0.0/32 is subnetted, 1 subnets
C    1.1.1.1 is directly connected, Loopback0
 2.0.0.0/32 is subnetted, 1 subnets
D    2.2.2.2 [90/158720] via 12.12.12.2, 00:00:35, FastEthernet0/0
 23.0.0.0/24 is subnetted, 1 subnets
D    23.23.23.0 [90/30720] via 12.12.12.2, 00:15:42, FastEthernet0/0
 172.16.0.0/24 is subnetted, 2 subnets
C    172.16.1.0 is directly connected, FastEthernet1/0
D    172.16.2.0 [90/33280] via 12.12.12.2, 00:15:30, FastEthernet0/0
 12.0.0.0/24 is subnetted, 1 subnets
C    12.12.12.0 is directly connected, FastEthernet0/0
C    192.168.1.0/24 is directly connected, Tunnel0
RT1#
RT1#ping 192.168.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.2, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/43/80 ms
RT1#

```

```

RT1#sh int tu0
Tunnel0 is up, line protocol is up
  Hardware is Tunnel
  Internet address is 192.168.1.1/24
  MTU 1514 bytes, BW 9 Kbit, DLY 500000 usec,
    reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel source 1.1.1.1 (Loopback0), destination 2.2.2.2
  Tunnel protocol/transport GRE/IP, key disabled, sequencing disabled
  Tunnel TTL 255
  Checksumming of packets disabled, fast tunneling enabled
  Last input 00:02:40, output 00:02:40, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/0 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
    17 packets input, 2540 bytes, 0 no buffer
    Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
    0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
    17 packets output, 2108 bytes, 0 underruns
    0 output errors, 0 collisions, 0 interface resets
    0 output buffer failures, 0 output buffers swapped out
RT1#

```

Below shows that RT1 and RT3 became EIGRP neighbors through the directly connected tunnel interface after implemented the **network 192.168.1.0** EIGRP router subcommand on both of them. However, the Ethernet networks 172.16.1.0/24 and 172.16.2.0/24 which were known through RT2, are still known through RT2; the path through the GRE tunnel is not being preferred. This is because tunnel interfaces have extremely low speed (bandwidth) and high delay by default. Tunnels are logical software constructs in which the processing delay is variable compared to physical interfaces; and hence should not be the most desirable paths by default.

```

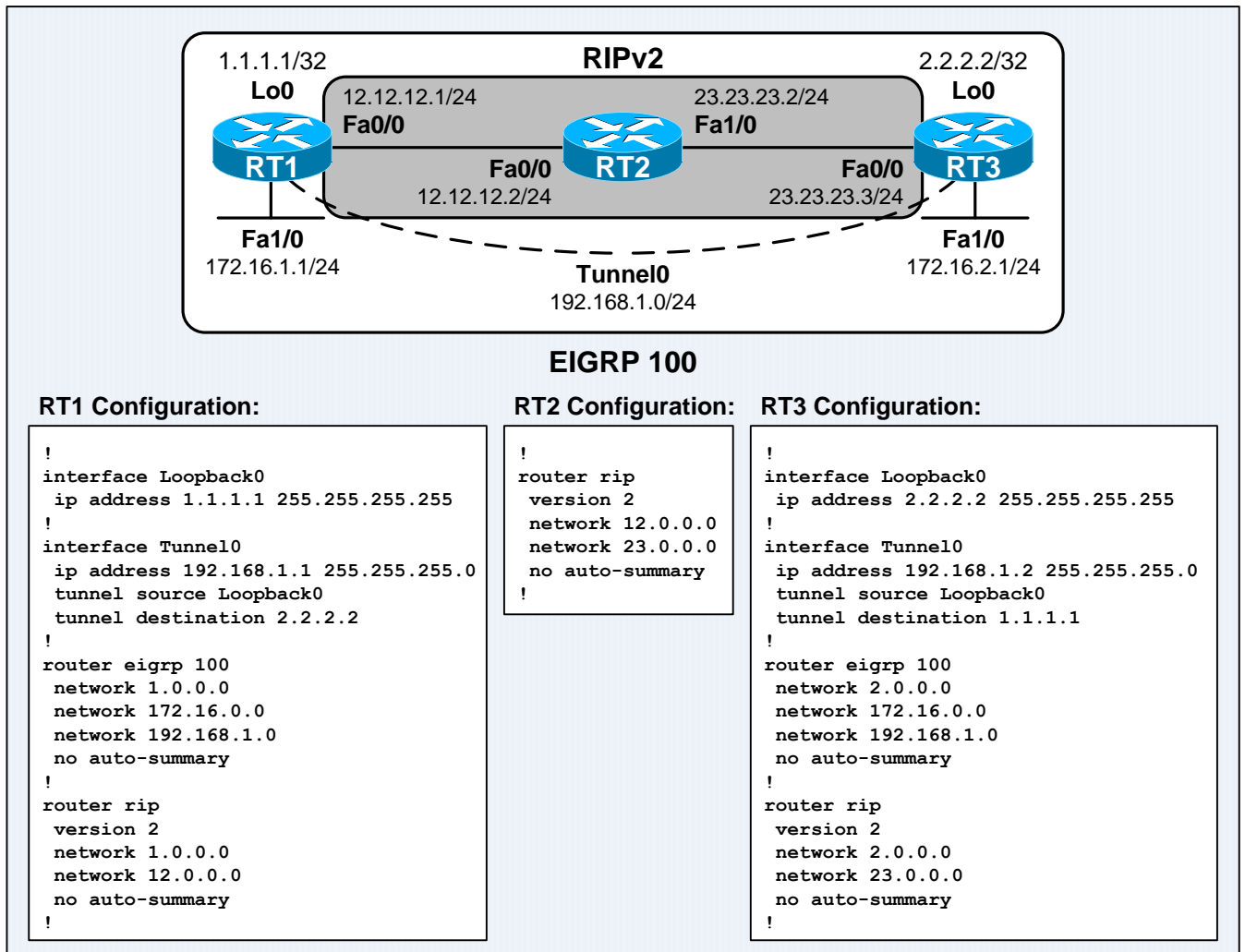
RT1#sh ip eigrp neighbors
IP-EIGRP neighbors for process 100
H   Address                Interface           Hold Uptime    SRTT   RTO   Q   Seq
                               (sec)          (ms)          Cnt Num
1  192.168.1.2              Tu0               13 00:00:06    1 3000  0  7
0  12.12.12.2               Fa0/0             12 00:25:49    402  2412  0  14
RT1#
RT1#sh ip eigrp topology
IP-EIGRP Topology Table for AS(100)/ID(172.16.1.1)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

--- output omitted ---
P 172.16.2.0/24, 1 successors, FD is 33280
  via 12.12.12.2 (33280/30720), FastEthernet0/0
  via 192.168.1.2 (297246976/28160), Tunnel0
RT1#

```

The introduction of a routing protocol across a GRE tunnel can cause an interesting problem known as **recursive routing**.



**Figure A6-55: GRE Tunnel Recursive Routing**

The network topology and configuration have been modified as shown in the figure above.

The routers are exchanging routes through RIPv2.

Note that RIPv2 exchange the routes for the loopback interfaces used for establishing the GRE tunnel. This may look a bit odd at the first glance. Consider the possibility that RT2 is owned and operated by a service provider, in which we cannot control them, and it only run RIPv2.

EIGRP is running on RT1 and RT3 to exchange routes for the Ethernet networks 172.16.1.0/24 and 172.16.2.0/24 between them.

The GRE tunnel comes up shortly after RT1 and RT3 learnt the RIPv2 routes for the loopback interfaces used for establishing the GRE tunnel.

But shortly after the tunnel came up, an error message is received upon the console.

```
RT1#sh ip route
```

```
Gateway of last resort is not set
```

```
    1.0.0.0/32 is subnetted, 1 subnets
C       1.1.1.1 is directly connected, Loopback0
    2.0.0.0/32 is subnetted, 1 subnets
R       2.2.2.2 [120/2] via 12.12.12.2, 00:00:19, FastEthernet0/0
    23.0.0.0/24 is subnetted, 1 subnets
R       23.23.23.0 [120/1] via 12.12.12.2, 00:00:19, FastEthernet0/0
    172.16.0.0/24 is subnetted, 1 subnets
C       172.16.1.0 is directly connected, FastEthernet1/0
    12.0.0.0/24 is subnetted, 1 subnets
C       12.12.12.0 is directly connected, FastEthernet0/0
C       192.168.1.0/24 is directly connected, Tunnel0
```

```
RT1#
```

```
00:03:08: %LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel0, changed state to up
```

```
00:03:12: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 192.168.1.2 (Tunnel0) is up: new adjacency
```

```
RT1#sh ip route
```

```
Gateway of last resort is not set
```

```
    1.0.0.0/32 is subnetted, 1 subnets
C       1.1.1.1 is directly connected, Loopback0
    2.0.0.0/32 is subnetted, 1 subnets
D       2.2.2.2 [90/297372416] via 192.168.1.2, 00:00:02, Tunnel0
    23.0.0.0/24 is subnetted, 1 subnets
R       23.23.23.0 [120/1] via 12.12.12.2, 00:00:25, FastEthernet0/0
    172.16.0.0/24 is subnetted, 2 subnets
C       172.16.1.0 is directly connected, FastEthernet1/0
D       172.16.2.0 [90/297246976] via 192.168.1.2, 00:00:02, Tunnel0
    12.0.0.0/24 is subnetted, 1 subnets
C       12.12.12.0 is directly connected, FastEthernet0/0
C       192.168.1.0/24 is directly connected, Tunnel0
```

```
RT1#
```

```
00:03:17: %TUN-5-RECURDOWN: Tunnel0 temporarily disabled due to recursive routing
```

```
00:03:18: %LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel0, changed state to down
```

```
00:03:18: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 192.168.1.2 (Tunnel0) is down: interface down
```

```
RT1#
```

```
RT1#sh ip route
```

```
Gateway of last resort is not set
```

```
    1.0.0.0/32 is subnetted, 1 subnets
C       1.1.1.1 is directly connected, Loopback0
    23.0.0.0/24 is subnetted, 1 subnets
R       23.23.23.0 [120/1] via 12.12.12.2, 00:00:04, FastEthernet0/0
    172.16.0.0/24 is subnetted, 1 subnets
C       172.16.1.0 is directly connected, FastEthernet1/0
    12.0.0.0/24 is subnetted, 1 subnets
C       12.12.12.0 is directly connected, FastEthernet0/0
```

```
RT1#
```

The %TUN-5-RECURDOWN: Tunnel0 temporarily disabled due to recursive routing error message is a result of the tunnel destination address is being learned through the tunnel itself. The main underlying reason of this problem is the administrative distances of the routing protocols. When the routing protocol with a better AD learns a route, it is being placed into the routing table. The tunnel relies upon the RIPv2 routes to the remote loopback interfaces, but the tunnel is also learning routes to the remote loopback interfaces – the next-hop of the tunnel is inside the tunnel! The router that first recognizes this routing loop problem would shut down the tunnel immediately. The EIGRP route is then lost, and the RIPv2 route returns; the tunnel is then being brought up again; the cycle continues indefinitely until a configuration change is made.

Below are the possible solutions for this problem:

- i) Stopping RT1 and RT3 from advertising the EIGRP routes to their loopback interfaces using the **no network 1.0.0.0** and **no network 2.0.0.0** EIGRP router subcommands on both of them.
- ii) Implement outbound EIGRP distribute lists on RT1 and RT3 to prevent them from advertising the EIGRP routes to their loopback interfaces.

Finally, the routes to the remote loopback interfaces are learnt through RIPv2 only; while the routes to the remote Ethernet networks 172.16.1.0/24 and 172.16.2.0/24 are learnt through EIGRP only.

```

RT1#sh ip route

Gateway of last resort is not set

    1.0.0.0/32 is subnetted, 1 subnets
C       1.1.1.1 is directly connected, Loopback0
    2.0.0.0/32 is subnetted, 1 subnets
R       2.2.2.2 [120/2] via 12.12.12.2, 00:00:05, FastEthernet0/0
    23.0.0.0/24 is subnetted, 1 subnets
R       23.23.23.0 [120/1] via 12.12.12.2, 00:00:05, FastEthernet0/0
    172.16.0.0/24 is subnetted, 2 subnets
C       172.16.1.0 is directly connected, FastEthernet1/0
D       172.16.2.0 [90/297246976] via 192.168.1.2, 00:00:25, Tunnel0
    12.0.0.0/24 is subnetted, 1 subnets
C       12.12.12.0 is directly connected, FastEthernet0/0
C       192.168.1.0/24 is directly connected, Tunnel0
RT1#
=====
RT3#sh ip route

Gateway of last resort is not set

    1.0.0.0/32 is subnetted, 1 subnets
R       1.1.1.1 [120/2] via 23.23.23.2, 00:00:03, FastEthernet0/0
    2.0.0.0/32 is subnetted, 1 subnets
C       2.2.2.2 is directly connected, Loopback0
    23.0.0.0/24 is subnetted, 1 subnets
C       23.23.23.0 is directly connected, FastEthernet0/0
    172.16.0.0/24 is subnetted, 2 subnets
D       172.16.1.0 [90/297246976] via 192.168.1.1, 00:00:35, Tunnel0
C       172.16.2.0 is directly connected, FastEthernet1/0
    12.0.0.0/24 is subnetted, 1 subnets
R       12.12.12.0 [120/1] via 23.23.23.2, 00:00:03, FastEthernet0/0
C       192.168.1.0/24 is directly connected, Tunnel0
RT3#

```

