Network Coding - an Introduction

Ralf Koetter and Muriel Medard
University of Illinois, Urbana-Champaign
Massachusetts Institute of Technology

Goals of Class

- To provide a general introduction to the new field of network coding
- To provide sufficient tools to enable the participants to apply and develop network coding methods in diverse applications
- To place network coding in the context of traditional network operation

Outline

- Basics of networks, routing and network coding:
  - Introduction to routing in traditional networks
    - routing along shortest paths
    - routing for recovery
  - Introduction to concepts of network coding
- Algebraic foundations:
  - Formal setup of linear network coding
  - Algebraic formulation
  - Algebraic min cut max flow condition
  - The basic multicast theorem
  - Other scenarios solvable with algebraic framework
  - Delays in networks

Outline (contd)

- More multicast - constructing codes
  - Coding gain is unbounded
  - Construction based on algebraic system
  - Construction based on flows
  - Undirected networks

Outline (contd)

- Decentralized code construction and network coding for multicast with a cost criterion
  - Randomized construction and its error behavior
  - Performance of distributed randomized construction - case studies
  - Robustness of randomized methods
  - Traditional methods based on flows - a review
  - Trees for multicasting - a review
  - Network coding with a cost criterion - flow-based methods for multicasting through linear programming
  - Distributed operation - one approach
  - A special case - wireless networks
  - Sample ISPs

Outline(contd)

- Non-multicast:
  - The algebraic difficulty
  - Vector solutions vs. instantaneous
  - Issue of linearity
  - Is the non-multicast case interesting?

Outline(contd)

- Network coding for multicast - relation to compression and generalization of Slepian-Wolf
  - Review of Slepian-Wolf
  - Distributed network compression
  - Error exponents
  - Source-channel separation issues
  - Code construction for finite field multiple access networks
- Network coding for security and robustness
  - Network coding for detecting attacks
  - Network management requirements for robustness
  - Centralized versus distributed network management
- New directions

Main topics

- Routing in networks operates in a manner akin to a transportation problem in which we seek to transport goods (data) in a cost-efficient fashion (multicast is a notable exception)
- Data is compressed and recovered at the edges
- Cost is defined according to a given cost of routes or by adjusting to the flows
- Current approaches do not generally make use of the fact that data (bits) are being transmitted

Shortest Paths

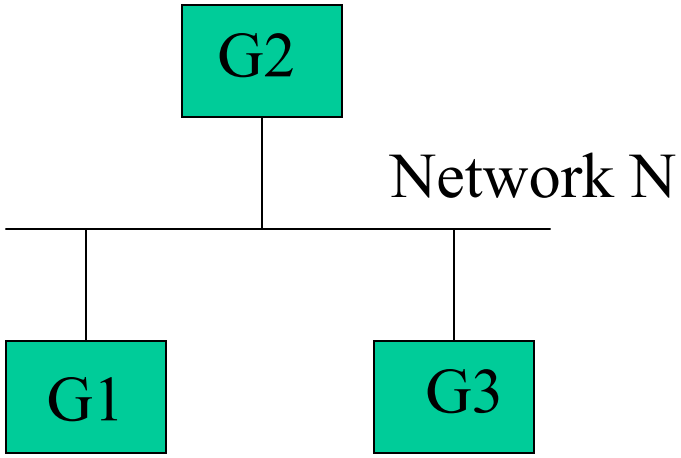
- Interior gateway protocol
- Option 1 (routing information protocol (RIP)):
  - vector distance protocol: each gateway propagates a list of the networks it can reach and the distance to each network
  - gateways use the list to compute new routes, then propagate their list of reachable networks
- Option 2 (open shortest path first (OSPF)):
  - link-state protocol: each gateway propagates status of its individual connections to networks
  - protocol delivers each link state message to all other participating gateways
  - if new link state information arrives, then gateway recomputes next-hop along shortest path to each destination

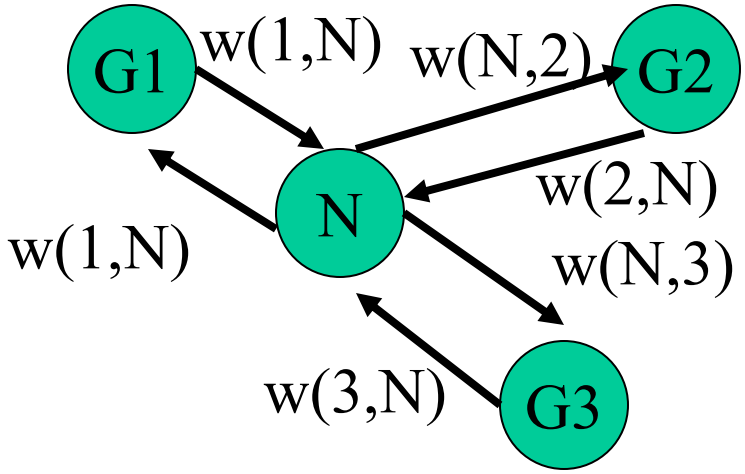
OSPF

- OSPF has each gateway maintain a topology graph
- Each node is either a gateway or a network
- If a physical connection exists between two objects in an internet, the OSPF graph contains a pair of directed edges between the nodes representing the objects
- Note: gateways engage in active propagation of routing information while hosts acquire routing information passively and never propagate it

OSPF

- Weights can be asymmetric: w(i,j) need not be equal to w(j,i)
- All weights are positive
- Weights are assigned by the network manager





Shortest Path Algorithms

- Shortest path between two nodes: length = weight
- Directed graphs (digraphs) (recall that MSTs were on undirected graphs), edges are called arcs and have a direction $(i,j) \neq (j,i)$
- Shortest path problem: a directed path from A to B is a sequence of distinct nodes A, n_1, n_2, …, n_k, B, where (A, n_1), (n_1, n_2), …, (n_k, B) are directed arcs - find the shortest such path
- Variants of the problem: find shortest path from an origin to all nodes or from all nodes to an origin
- Assumption: all cycles have non-negative length
- Three main algorithms:
  - Dijsktra
  - Bellman-Ford
  - Floyd-Warshall

Bellman-Ford

- Allows negative lengths, but not negative cycles
- B-F works at looking at negative lengths from every node to node 1
- If arc (i,j) does not exist, we set d(i,j) to
- We look at walks: consider the shortest walk from node i to 1 after at most h arcs
- Algorithm:
  - $D^{h+1}(i) = \min_{\text{over all } j}[d(i,j) + D^h(i)]$ for all i other than 1
  - we terminate when $D^{h+1}(i) = D^h(i)$
- The $D^{h+1}(i)$ are the lengths of the shortest path from i to 1 with no more than h arcs in it

Bellman-Ford

- Let us show this by induction
  - $D^1(i) = d(i,1)$ for every i other than 1, since one hop corresponds to having a single arc
  - now suppose this holds for some h, let us show it for h+1: we assume that for all $k \leq h$, $D^k(i)$ is the length of the shortest walk from i to 1 with k arcs or fewer
  - $\min_{\text{over all } j}[d(i,j) + D^h(i)]$ allows up to h+1 arcs, but $D^h(i)$ would have fewer than h arcs, so $\min[D^h(i), \min_{\text{over all } j}[d(i,j) + D^h(i)]] = D^{h+1}(i)$
- Time complexity: A, where A is the number of arcs, for at most N-1 nodes (note: A can be up to $(N-1)^2$)
- In practice, B-F still often performs better than Dijkstra ($O(N^2)$)

Distributed Asynchronous B-F

- The algorithms we investigated work well when we have a single centralized entity doing all the computation - what happens when we have a network that is operating in a distributed and asynchronous fashion?
- Let us call N(i) the set of nodes that are neighbors of node i
- At every time t, every node i other than 1 has available :
  - $D^i_j(t)$: estimate of shortest distance of each neighbor node j in N(i) which was last communicated to node i
  - $D^i(t)$: estimate of the shortest distance of node i which was last computed at node i using B-F

Distributed Asynchronous B-F

- $D^1(t) = 0$ at all times
- Each node i has available link lengths d(i,j) for all j in N(i)
- Distance estimates change only at time $t_0, t_1, .., t_m$, where t_m becomes infinitely large at m becomes infinitely large
- At these times:
  - $D^i(t) = \min_{j \text{ in } N(i)}[d(i,j) + D^i_j(t)]$, but leaves estimate $D^i_j(t)$ for all j in N(i) unchanged

OR

  - node i receives from one or more neighbors their D^j, which becomes D^i_j (all other D^i_j are unchanged)

OR

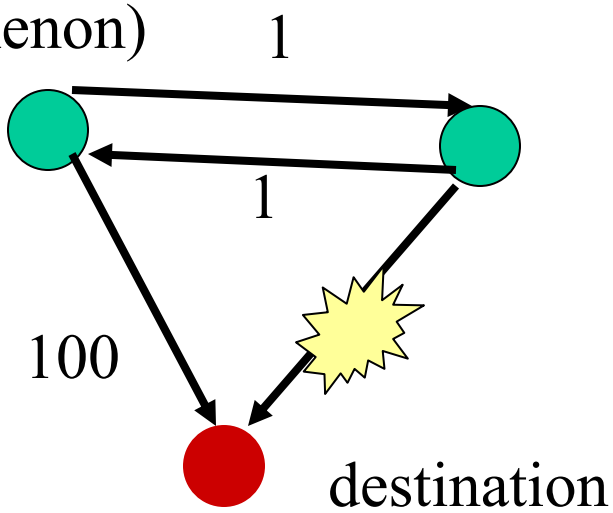
  - node i is idle

Distributed Asynchronous B-F

- Assumptions:
  - if there is a link (i,j), there is also a link (j,i)
  - no negative length cycles
  - nodes never stop updating estimates and receiving updated estimates
  - old distance information is eventually purged
  - distances are fixed
- Under those conditions:for any initial $D^i_j(t0)$, $D^i(t)$, for some t_m, eventually all values $D^i(t) = D^i$ for all t greater than t_m

Failure recovery

- Often asynchronous distributed Bellman-Ford works even when there are changes, including failures
- However, the algorithm may take a long time to recover from a failure that is located on a shortest path, particularly if the alternate path is much longer than the original path (bad news phenomenon)



Rerouting

- We have considered how to route when we have a static network, but we must also consider how to react when we have changes, in particular when we need to avoid a location because of failures or because of congestion

- Preplanned:
  - fast (ms to ns)
  - typically a large portion of the whole network is involved in re-routing
  - traditionally combines self-healing rings (SHRs) and diversity protection (DP) => constrains topology
  - hard-wired
  - all excess capacity is preplanned

- Dynamic:
  - slow (s to mn)
  - typically localized and distributed
  - well-suited to mesh networks => more flexibility in topology
  - software approach
  - uses real-time availability of spare capacity

Example of rerouting in the IP world

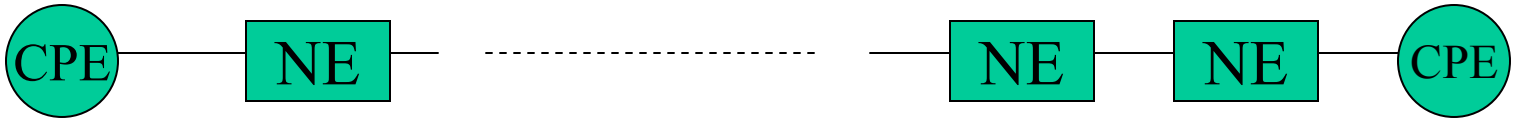
- Internet control message protocol (ICMP)
- Gateway generates ICMP error message, for instance for congestion
- ICMP redirect: “ipdirect” specifies a pointer to a buffer in which there is a packet, an interface number, pointer to a new route
- How do we get new route?
  - First: check the interface is other than the one over which the packet arrives
  - Second: run “rtget” (route get) to compute route to machine that sent datagram, returns a pointer to a structure describing the route
- If the failure or congestion is temporary, we may use flow control instead of a new route

Rerouting for ATM

- ATM is part datagram, part circuit oriented, so recovery methods span many different types
- Dynamic methods release connections and then seek ways of re-establishing them: not necessarily per VP or VC approach
  - private network to network interface (PNNI) crankback
  - distributed restoration algorithms (DRAs)
- Circuit-oriented methods often have preplanned component and work on a per VC, VP basis
  - dedicated shared VPs, VCs or soft VPs, VCs

PNNI self-healing

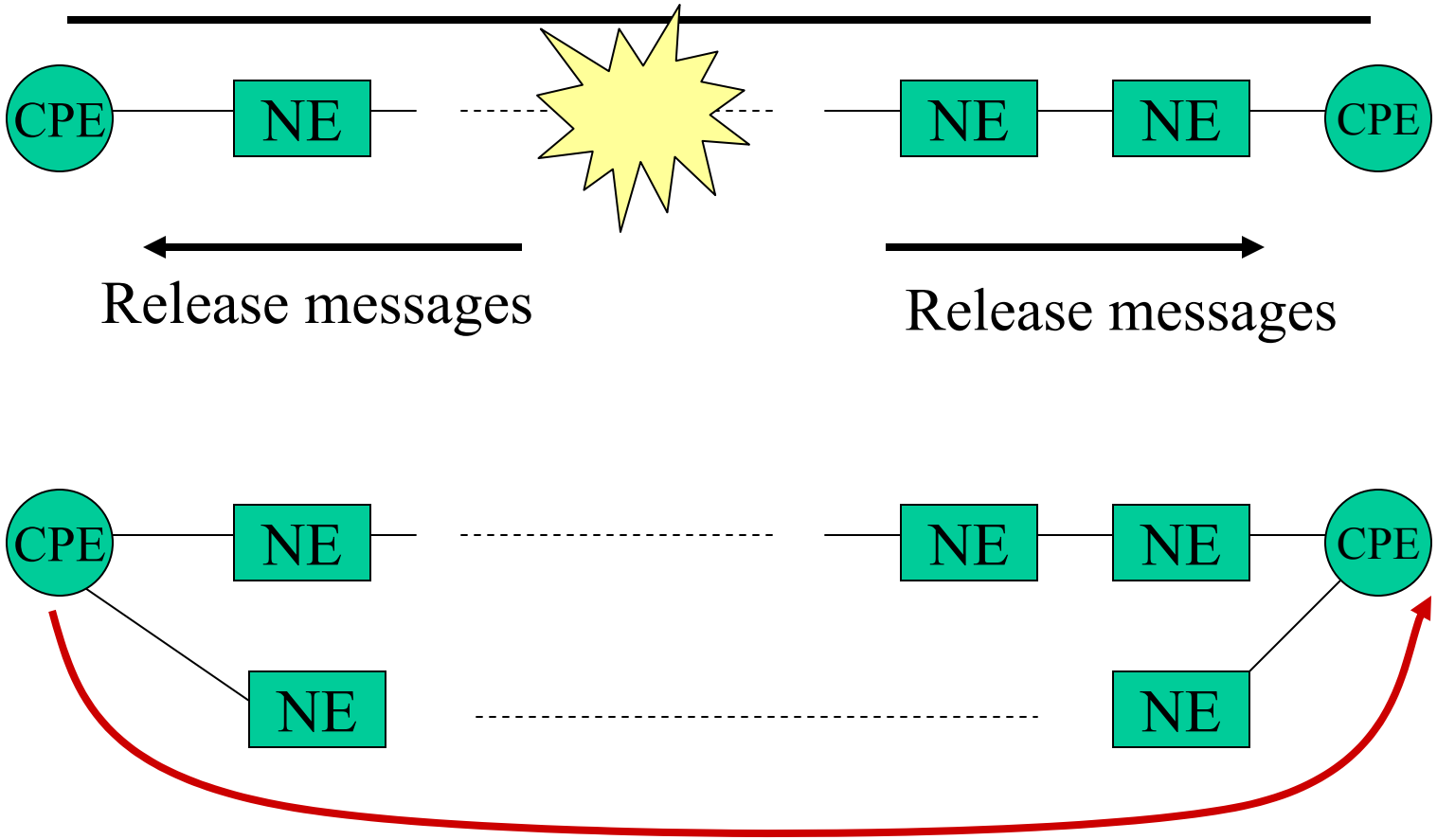
- PNNI is how ATM switches talk to each other
- Around failure or congestion area, initiate crankback
- End equipment (CPE: customer premise equipment) initiates a new connection
- In phase 2 PNNI, automatic call rerouting, freeing up CPEs from having to instigate new calls, the ATM setup message includes a request for a fault-tolerant connection



Network element

Before failure

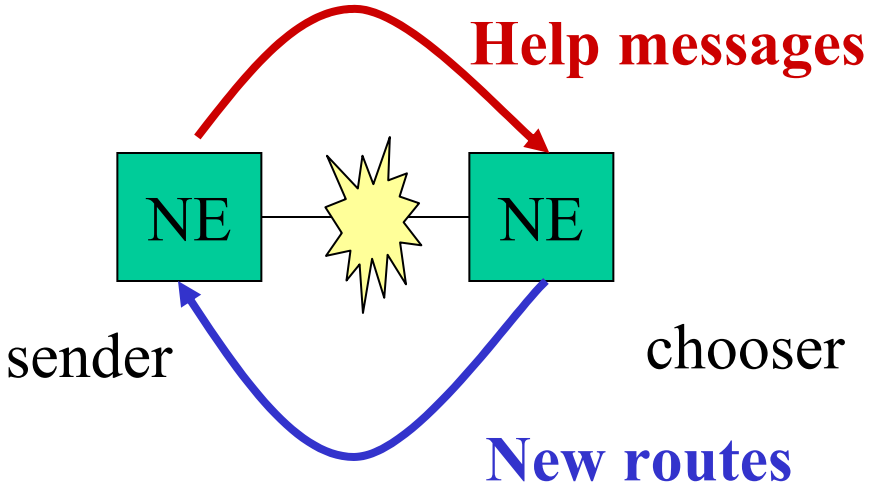
Connection re-establishment

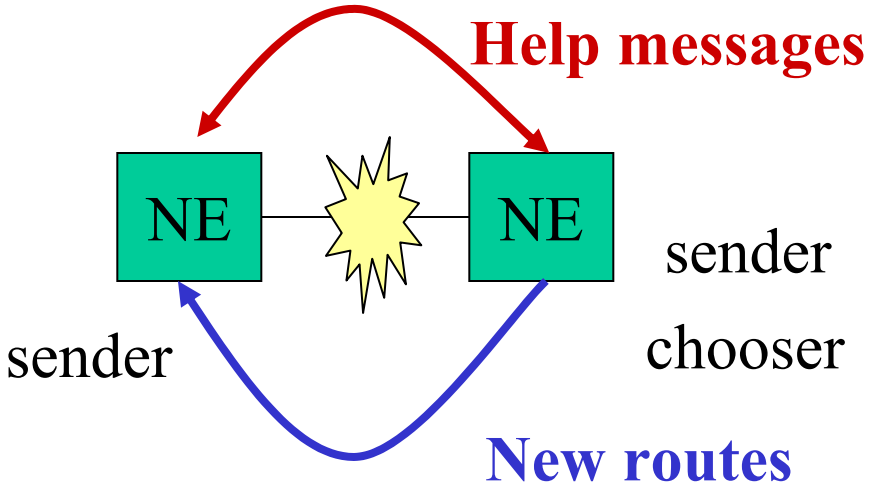


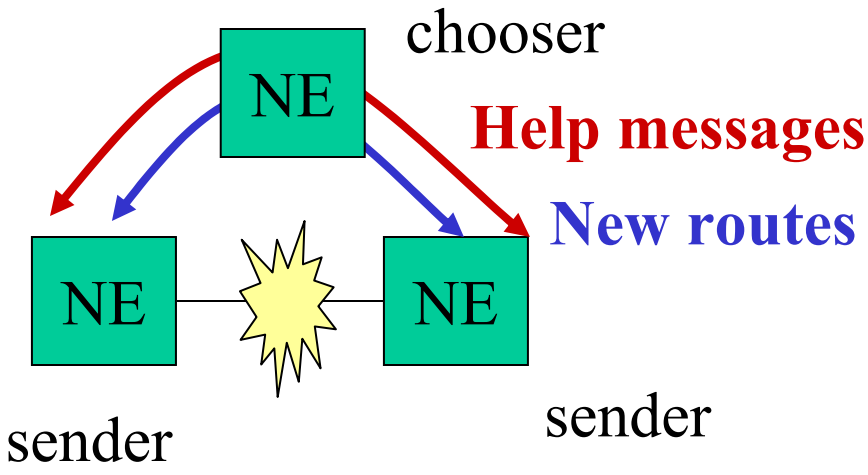
New connection is established

Issue: the congestion may cascade, giving unstable conditions, which cause an ATM storm

DRAs







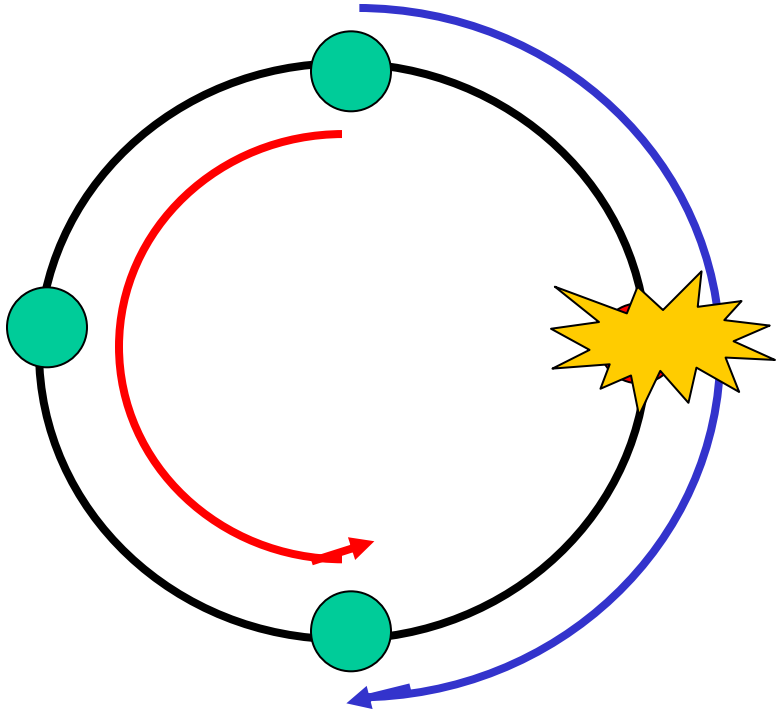
The DRAs have at least one end node transmit help messages to some nodes around them, usually within a certain hop radius, and new routes, possible splitting flows, are selected and used

Circuit-oriented methods

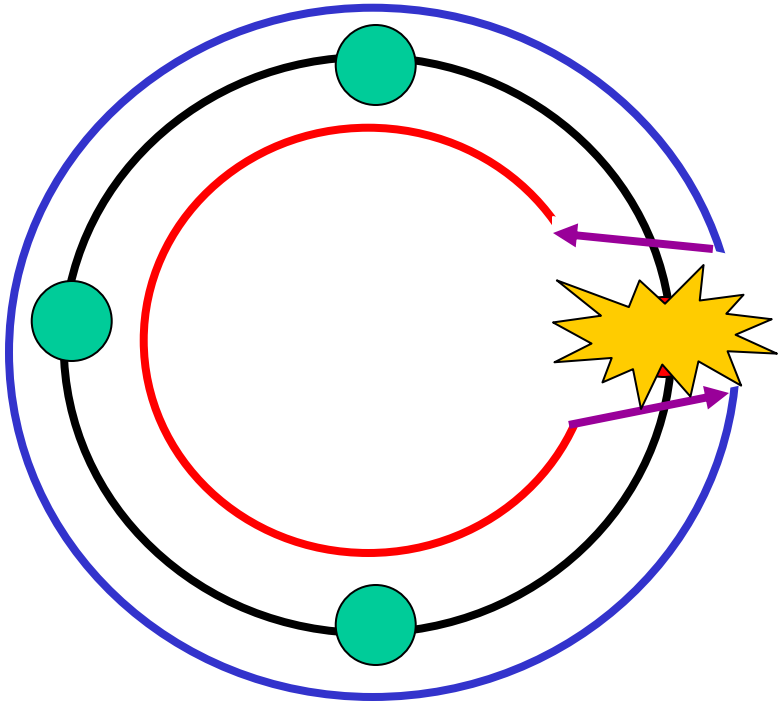
- Circuit-oriented methods seek to replace a route with another one, whether end-to-end or over some portion that is affected by a failure
- Several issues arise:
  - How do we perform recovery in a bandwidth-efficient manner
  - How does recovery interface with network management
  - What sort of granularity do we need
  - What happens when a node rather than a link fails

Rings: Path and Link/Node Rerouting

UPSR: automatic path switching on Unidirectional Path Switched Ring

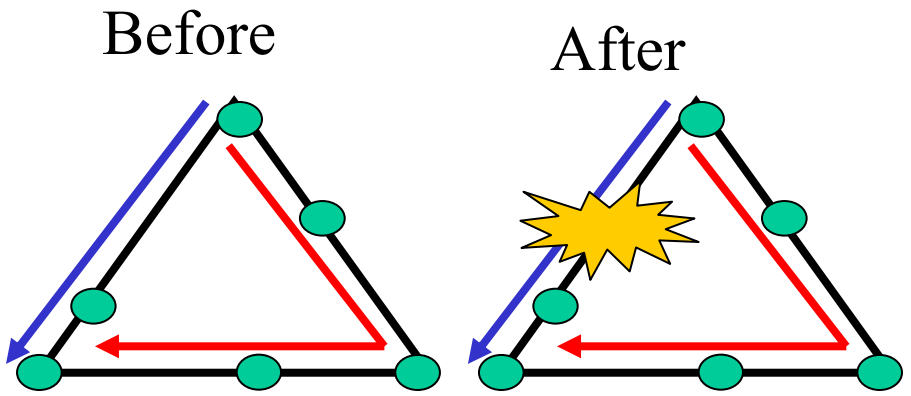


BLSR: link/node rerouting on Bidirectional Line Switched Ring

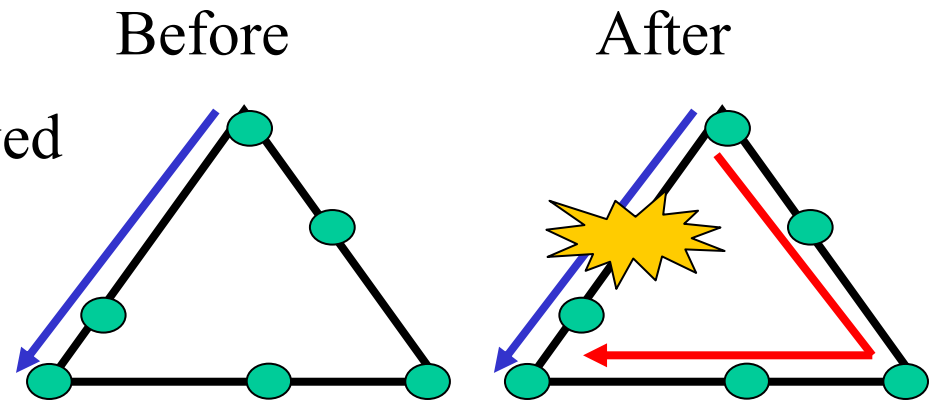


Path-based methods

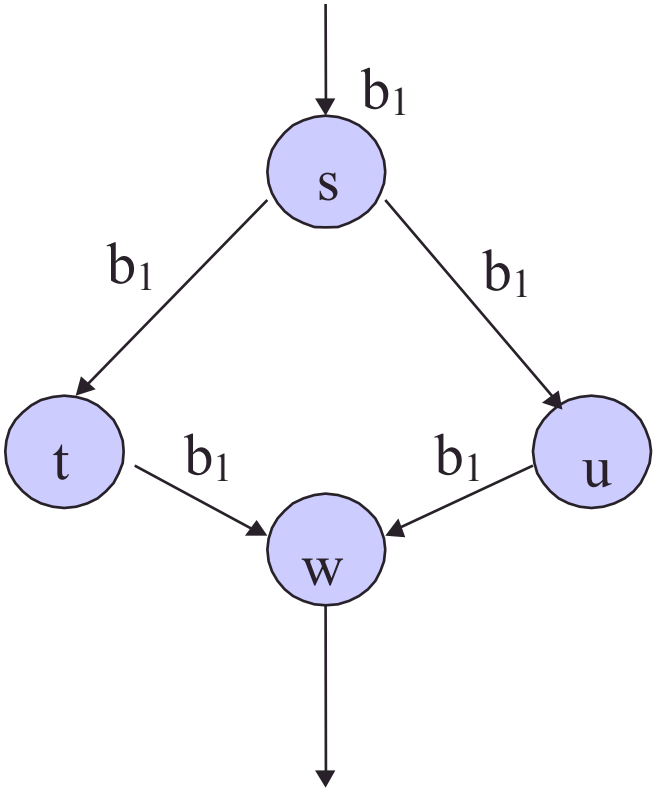
- Live back-up
  - backup bandwidth is dedicated
  - only receiver is involved
  - fast but bandwidth inefficient



- Failure triggered back-up
  - backup bandwidth is shared
  - sender and receiver are involved
  - slow but bandwidth efficient

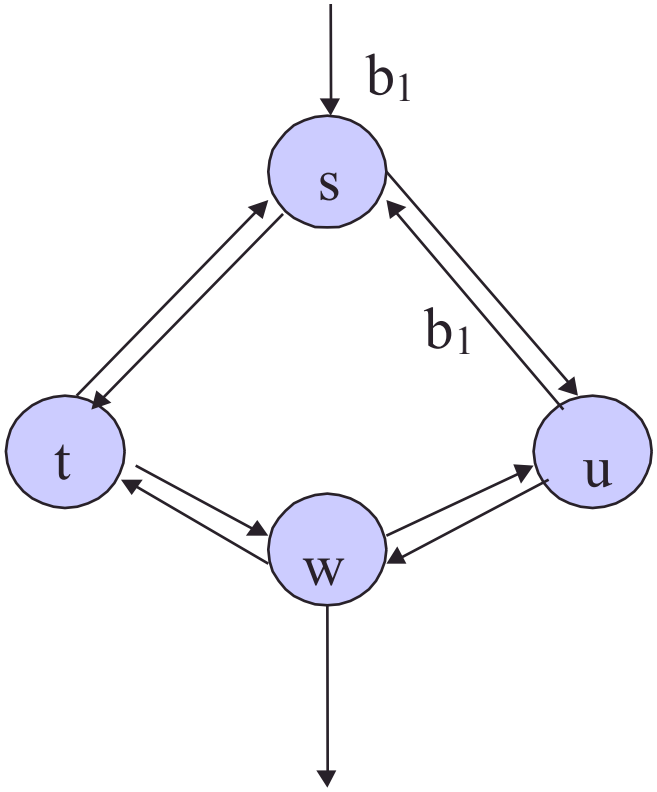


Rerouting as a code



$\overline{s} \cdot d_{t,w} + s \cdot d_{u,w} = b_1$

a. Live path protection

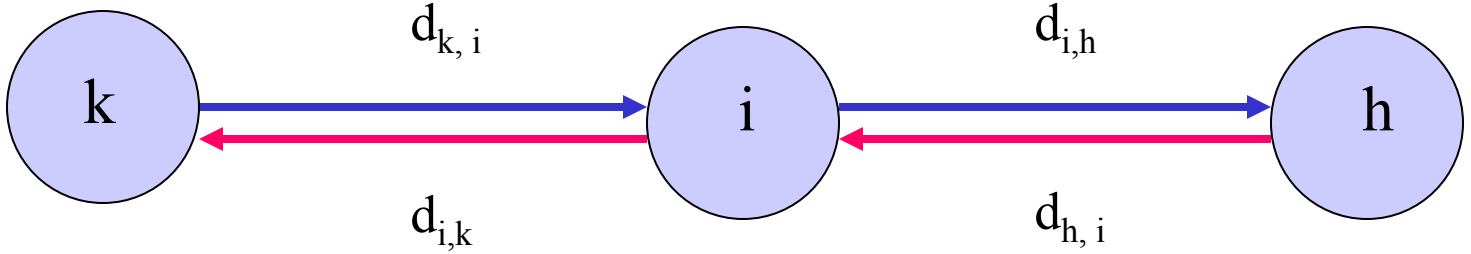


$d_{t,w} + d_{u,w} = b_1$

b. Link recovery

Rerouting as a code

- Live path protection: we have an extra supervisory signal s =1 when the primary path is live, s =0 otherwise
- Failure-triggered path protection: the backup signal is multiplied by $\bar{s}$
- Link recovery:
  - $d_{i,h} = d_{k,i} + d_{h,i}$ for the primary link (i, h) emanating from i, where (k, i) is the primary link into i and (h, i) is the secondary link into i
  - for secondary link emanating from i, the code is $d_{i,k} = d_{i,h} \cdot s_{i,h} + d_{i,h}$

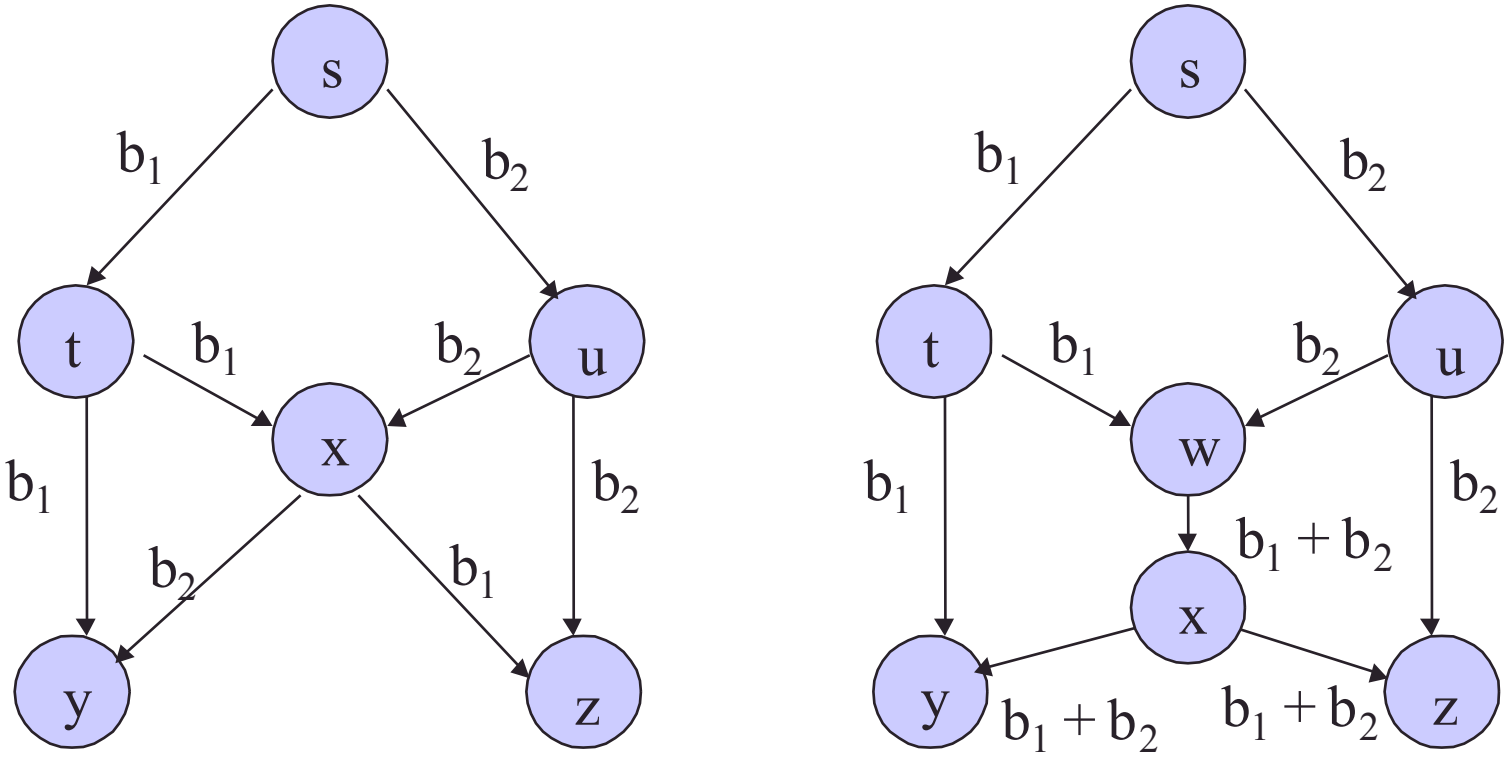


Codes and routes

- In effect, every routing and rerouting scheme can be mapped to some type of code, which may involve the presence of a network management component
- Thus, removing the restrictions of routing can only improve performance - can we actively make use of this generality?

Network coding

- The canonical example



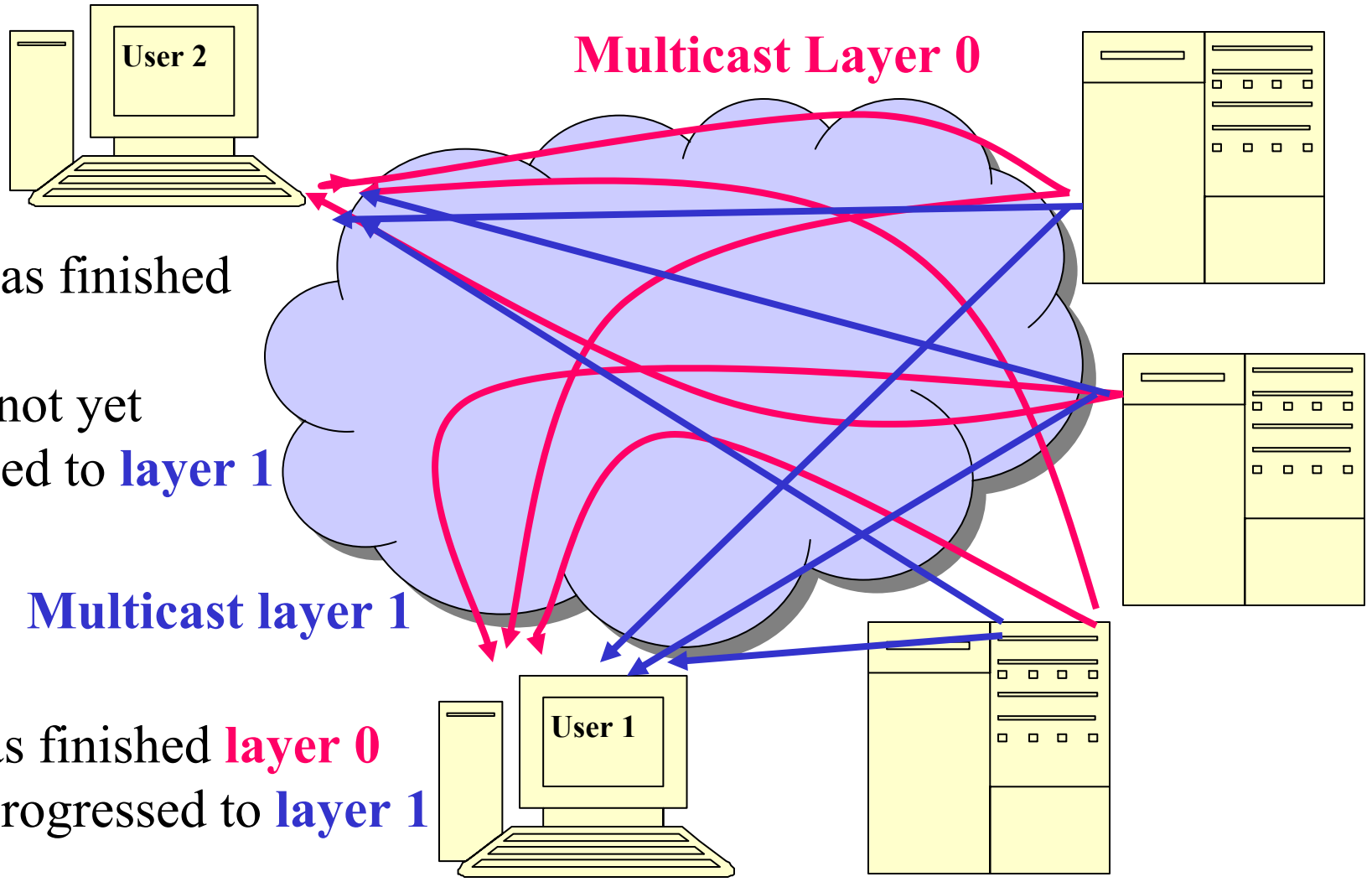
Coding across the network - have I seen this before?

- Several source-based systems exist or have been proposed
- Routing diversity to average out the loss of packets over the network
- Access several mirror sites rather than single one
- The data is then coded across packets in order to withstand the loss of packets without incurring the loss of all packets
- Rather than select the “best” route, routes are diverse enough that congestion in one location will not bring down a whole stream
- This may be done with traditional Reed-Solomon erasure codes or with Tornado codes

The Digital Fountain approach

- Idea: have users tune in whenever they want, and receive data according to the bandwidth that is available at their location in the network – “fountain” because the data stream is always on
- Create multicast layers: each layer has twice the bandwidth of the lower layer (think of progressively better resolution on images, for instance), except for the first two layers
- If receiver stays at same layer throughout, and packet loss rate is low enough, then receiver can reconstruct source data before receiving any duplicate packets : "One-level property"
- Receivers can only subscribe to higher layer after seeing a*synchronization point* (SP) in their own layer
- The frequency of SPs is inversely proportional to layer bandwidth

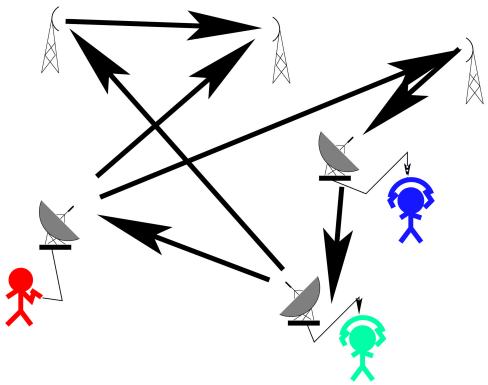
Digital fountain



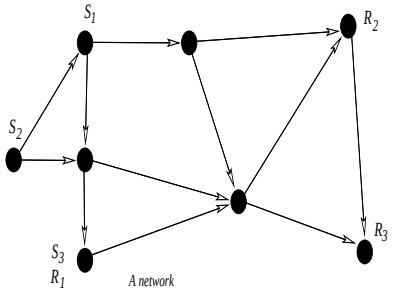
Network coding vs. Coding for networks

- The source-based approaches consider the networks as in effect channels with ergodic erasures or errors, and code over them, attempting to reduce excessive redundancy
- The data is expanded, not combined to adapt to topology and capacity
- Underlying coding for networks, traditional routing problems remain, which yield the virtual channel over which coding takes place
- Network coding subsumes all functions of routing - algebraic data manipulation and forwarding are fused

II — Algebraic Foundations of Network Coding



$\Rightarrow$

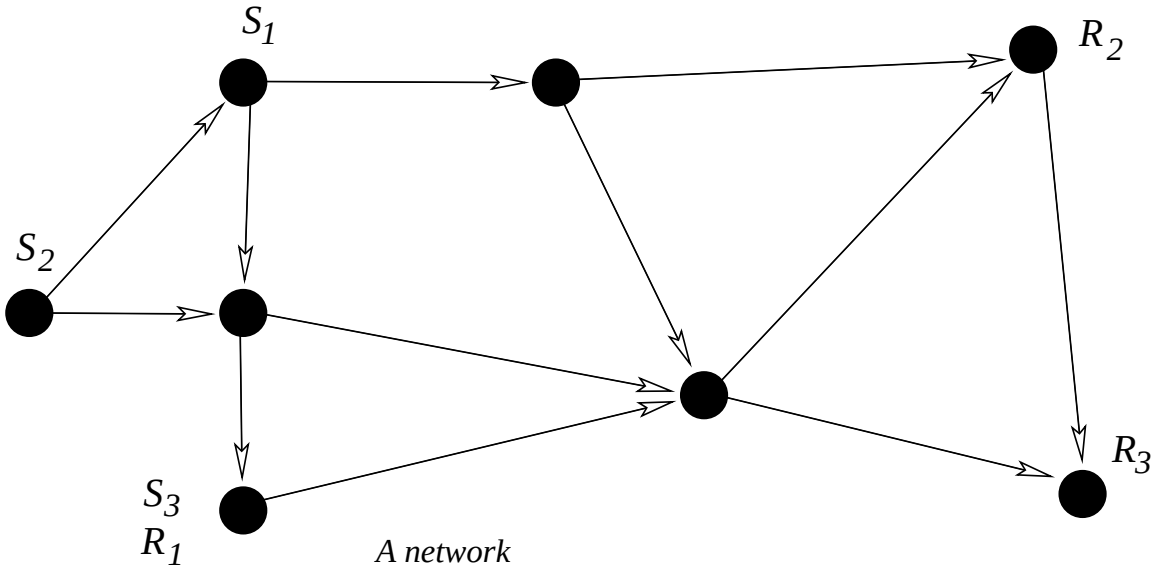


$\Rightarrow \quad A(I-F)^{-1}B^T = I$

Why an "algebraic" characterization?

- Graph-theoretic proofs are cumbersome
- Generalizations are possible
- Equations are easier managed than graphs
- Powerful tools available

Problem Description



A network

Vertices: V
Edges: $E \subseteq V \times V$, $e = (v, u) \in E$
Edge capacity: $C(e)$
Network: $\mathcal{G} = (V, E)$
Source nodes: $\{v_1, v_2, \ldots, v_N\} \subseteq V$
Sink nodes: $\{u_1, u_2, \ldots, u_K\} \subseteq V$

μ input random processes at v:
$\mathscr{X}(v) = \{X(v,1), X(v,2), \ldots, X(v,\mu(v)\}$

ν Output random processes at u:
$\mathscr{Z}(u) = \{Z(u,1), Z(u,2), \ldots, Z(u,\nu(u))\}$

Random processes on edges: $Y(e)$

A connection:
$c = (v, u, \mathscr{X}(v,u))$, $\mathscr{X}(v,u) \subseteq \mathscr{X}(v)$

A connection is established if $\mathscr{Z}(u) \supset \mathscr{X}(v,u)$

Set of connections: $\mathscr{C}$

The pair $(\mathcal{G}, \mathscr{C})$ defines a network coding problem .

Is the problem $(\mathcal{G}, \mathcal{C})$ solvable?

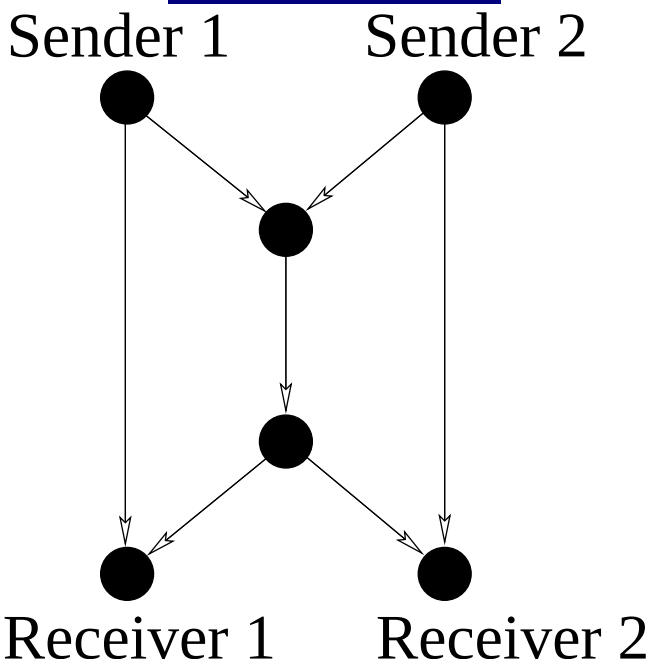
How do we find a solution?

Is the problem $(\mathcal{G}, \mathcal{C})$ solvable?

How do we find a solution?

This is fairly idealized (synchronization, protocol, dynamic behaviour, error free operation,...) but gives insights into possible limits and opportunities.

An Example



[1] Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow", IEEE-IT, vol. 46, pp. 1204-1216, 2000

[2] S.-Y. R. Li, R. W. Yeung, and N. Cai "Linear Network Coding", preprint, 2000

More Simplifications — Linear Network Codes

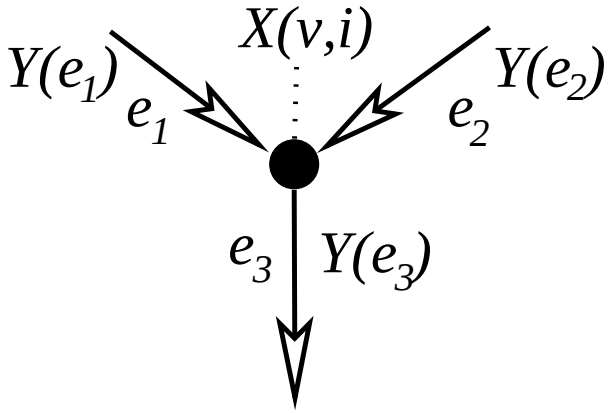
$C(e) = 1$ (links have the same capacity)
$H(X(v,i)) = 1$ (sources have the same rate)
The $X(v,i)$ are mutually independent.
Vector symbols of length m elements in $\mathbb{F}_{2^m}$.

($\mathbb{F}_{2^m}$ is the finite field with m elements we can add, subtract, divide and multiply elements in $\mathbb{F}_{2^m}$ without going crazy!)

This is necessary to define linear operations.

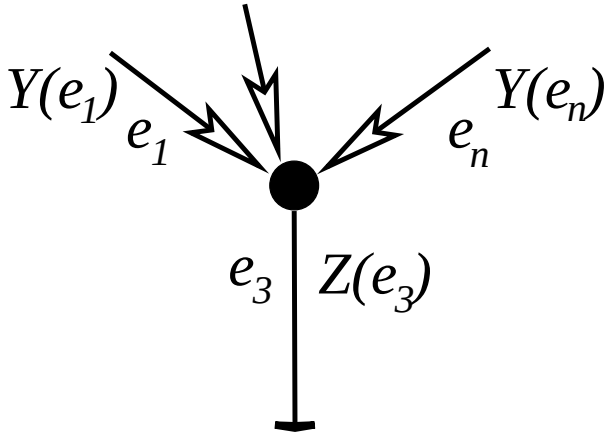
More Simplifications — Linear Network Codes

All operations at network nodes are linear!



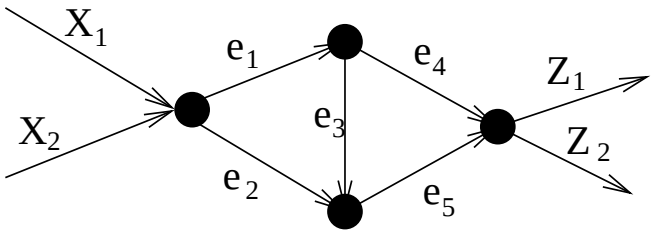
$$Y(e_3) = \sum_i \alpha_i X(v,i) + \sum_{j=1,2} \beta_j Y(e_j)$$

At a receiver (terminal) node:

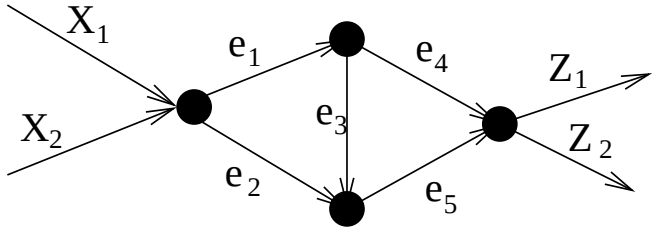


$$Z(v,j) = \sum_{j=1}^{n} \varepsilon_j Y(e_j).$$

A simple example



$$
\begin{aligned}
Y(e_1) &= \alpha_{1,e_1} X_1 + \alpha_{2,e_1} X_2 \\
Y(e_2) &= \alpha_{1,e_2} X_1 + \alpha_{2,e_2} X_2 \\
Y(e_3) &= \beta_{e_1,e_3} Y(e_1) \\
Y(e_4) &= \beta_{e_1,e_4} Y(e_1) \\
Y(e_5) &= \beta_{e_2,e_5} Y(e_2) + \beta_{e_3,e_5} Y(e_3) \\
Z_1 &= \varepsilon_{e_4,1} Y(e_4) + \varepsilon_{e_5,1} Y(e_5) \\
Z_2 &= \varepsilon_{e_4,2} Y(e_4) + \varepsilon_{e_5,2} Y(e_5)
\end{aligned}
$$



In matrix form (after solving the linear system)

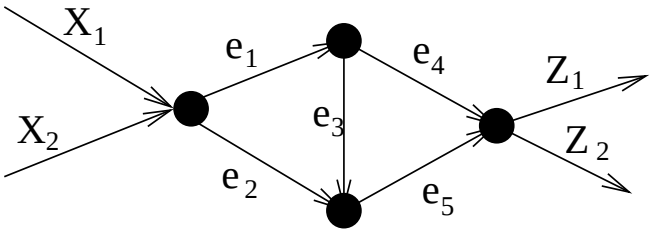
$$\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} = \underbrace{\begin{pmatrix} \varepsilon_{e_4,1} & \varepsilon_{e_5,1} \\ \varepsilon_{e_4,2} & \varepsilon_{e_5,2} \end{pmatrix}}_{B} \underbrace{\begin{pmatrix} \beta_{e_1,e_4} & 0 \\ \beta_{e_1,e_3}\beta_{e_3,e_5} & \beta_{e_2,e_5} \end{pmatrix}}_{G} \underbrace{\begin{pmatrix} \alpha_{1,e_1} & \alpha_{1,e_2} \\ \alpha_{2,e_1} & \alpha_{2,e_2} \end{pmatrix}}_{A} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$$

We define three matrices A, G, B

The main question becomes: Is G invertible?

The transfer matrix

Let a matrix F be defined as an $|E| \times |E|$ matrix where $f_{i,j}$ is defined as β_{e_i,e_j}, i.e. the coefficient with which $Y(e_i)$ is mixed into Y_{e_j}.



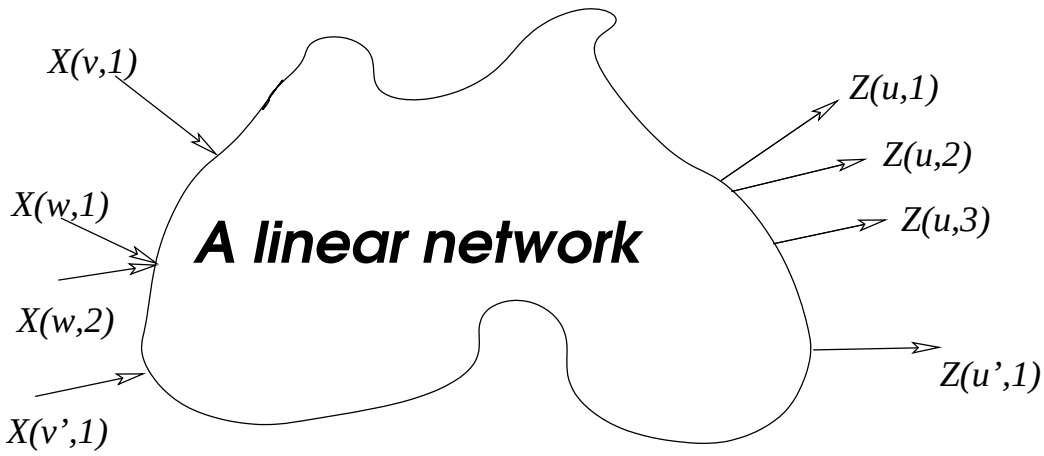
$$F = \begin{pmatrix} 0 & 0 & \beta_{e_1,e_3} & \beta_{e_1,e_4} & 0 \\ 0 & 0 & 0 & 0 & \beta_{e_2,e_5} \\ 0 & 0 & 0 & 0 & \beta_{e_3,e_5} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Summing the "path gains":

$$P = I + F + F^2 + \ldots = (I - F)^{-1} = \begin{pmatrix} 0 & 0 & \beta_{e_1,e_3} & \beta_{e_1,e_4} & \beta_{e_1,e_3}\beta_{e_3,e_5} \\ 0 & 0 & 0 & 0 & \beta_{e_2,e_5} \\ 0 & 0 & 0 & 0 & \beta_{e_3,e_5} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Observe that $G = (I - F)^{-1}$ is polynomial

A linear system



Input vector: $\underline{x}^T = (X(v,1), X(v,2), \ldots, X(v', \mu(v')))$

Output vector: $\underline{z}^T = (Z(u,1), Z(u,2), \ldots, Z(u', \nu(u')))$

Transfer matrix: M, $\underline{z} = M\underline{x} = B \cdot G \cdot A\ \underline{x}$

$\underline{\xi} = (\xi_1, \xi_2, \ldots,) = (\ldots, \alpha_{e,l}, \ldots, \beta_{e',e}, \ldots, \varepsilon_{e',j}, \ldots)$

$$\underline{z} = M\underline{x} = B \cdot \underbrace{(I - F^T)^{-1}}_{G^T} \cdot A\ \underline{x}$$

$$\underline{\xi} = (\xi_1, \xi_2, \ldots,) = (\ldots, \alpha_{e,l}, \ldots, \beta_{e',e}, \ldots, \varepsilon_{e',j}, \ldots)$$

For acyclic networks the elements of G (and hence M) are polynomial functions in variables $\underline{\xi} = (\xi_1, \xi_2, \ldots,)$

$\Rightarrow$ an algebraic characterization of flows....

An algebraic Min-Cut Max-Flow condition

Let network be given with a source v and a sink v'. The following three statements are equivalent:

1. A point-to-point connection $c = (v, v', \mathscr{X}(v, v'))$ is possible.

2. The Min-Cut Max-Flow bound is satisfied for a rate $R(c) = |\mathscr{X}(v, v')|$.

3. The determinant of the $R(c) \times R(c)$ transfer matrix M is nonzero over the ring of polynomials $\mathbb{F}_2[\underline{\xi}]$

3. $\Rightarrow$ We have to study the solution sets of polynomial equations.

An innocent looking Lemma

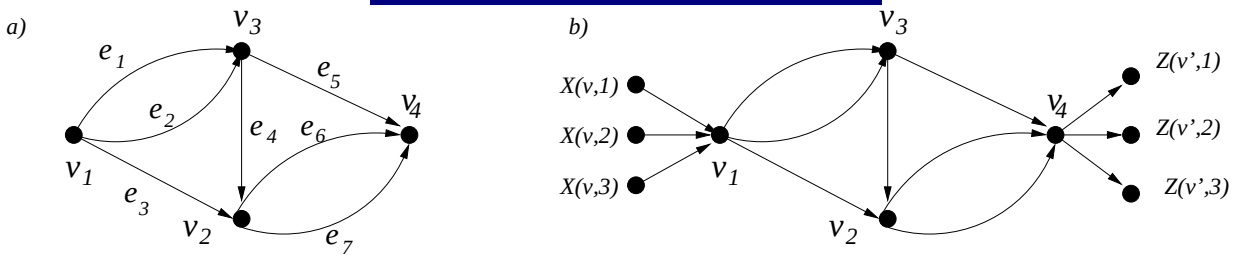
Let $\mathbb{F}[X_1, X_2, \ldots, X_n]$ be the ring of polynomials over an infinite field $\mathbb{F}$ in variables $X_1, X_2, \ldots, X_n$. For any non-zero element $f \in \mathbb{F}[X_1, X_2, \ldots, X_n]$ there exists an infinite set of n-tuples $(x_1, x_2, \ldots, x_n) \in \mathbb{F}^n$ such that $f(x_1, x_2, \ldots, x_n) \neq 0$.

An innocent looking Lemma

Let $\mathbb{F}[X_1, X_2, \ldots, X_n]$ be the ring of polynomials over an infinite field $\mathbb{F}$ in variables $X_1, X_2, \ldots, X_n$. For any non-zero element $f \in \mathbb{F}[X_1, X_2, \ldots, X_n]$ there exists an infinite set of n-tuples $(x_1, x_2, \ldots, x_n) \in \mathbb{F}^n$ such that $f(x_1, x_2, \ldots, x_n) \neq 0$.

$(x^6 - x^4 - x^2 + x)$ does not have have a non-solution in $\mathbb{F}_2$, $\mathbb{F}_3$, $\mathbb{F}_4$ but in $\mathbb{F}_5$ we have $2^6 - 2^4 - 2^2 + 2 = 46 \equiv 1 (mod\ 5)$.

Another Example:



$$\mathscr{C} = (v_1, v_4, \{X(v_1,1), X(v,2), X(v_1,3)\})$$

$$A = \begin{pmatrix} \alpha_{e_1,1} & \alpha_{e_2,1} & \alpha_{e_3,1} \\ \alpha_{e_1,2} & \alpha_{e_2,2} & \alpha_{e_3,2} \\ \alpha_{e_1,3} & \alpha_{e_2,3} & \alpha_{e_3,3} \end{pmatrix}, \quad B = \begin{pmatrix} \varepsilon_{e_5,1} & \varepsilon_{e_5,2} & \varepsilon_{e_5,3} \\ \varepsilon_{e_6,1} & \varepsilon_{e_6,2} & \varepsilon_{e_6,3} \\ \varepsilon_{e_7,1} & \varepsilon_{e_7,2} & \varepsilon_{e_7,3} \end{pmatrix}.$$

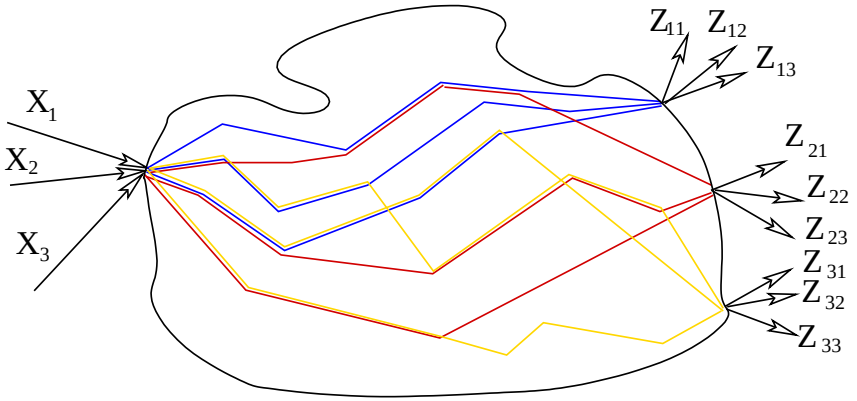
$$M = A \begin{pmatrix} \beta_{e_1,e_5} & \beta_{e_1,e_4}\beta_{e_4,e_6} & \beta_{e_1,e_4}\beta_{e_4,e_7} \\ \beta_{e_2,e_5} & \beta_{e_2,e_4}\beta_{e_4,e_6} & \beta_{e_2,e_4}\beta_{e_4,e_7} \\ 0 & \beta_{e_3,e_6} & \beta_{e_3,e_6} \end{pmatrix} B^T.$$

$$\begin{aligned} \det(M) &= \det(A)\det(B) \\ &\quad (\beta_{e_1,e_5}\beta_{e_2,e_4} - \beta_{e_2,e_5}\beta_{e_1,e_5})(\beta_{e_4,e_6}\beta_{e_3,e_7} - \beta_{e_4,e_7}\beta_{e_3,e_6}) \end{aligned}$$

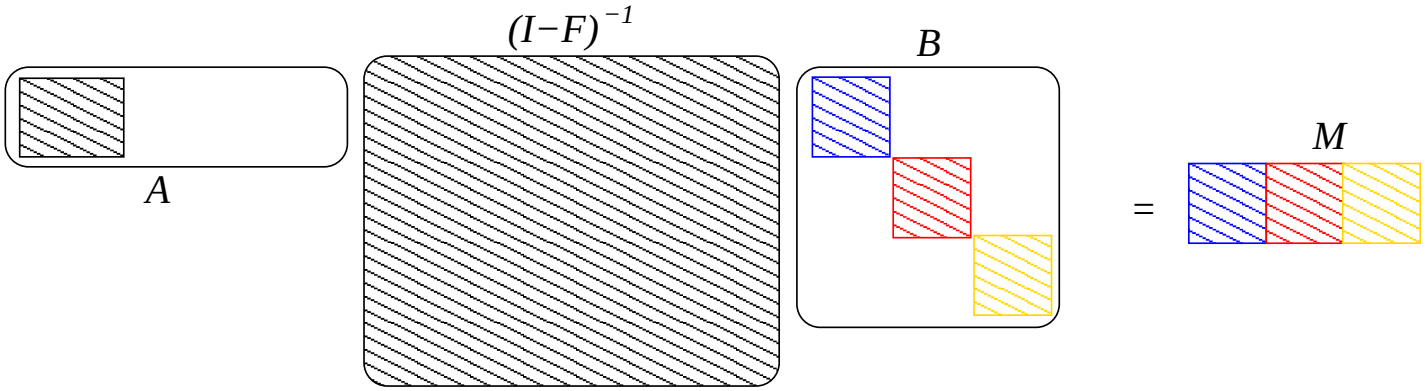
Choose the coefficients so that $\det(M) \neq 0$!

Multicast:

$\mathscr{C} = \{(v, u_1, \mathscr{X}(v)), (v, u_2, \mathscr{X}(v)), \ldots, (v, u_K, \mathscr{X}(v))\}$

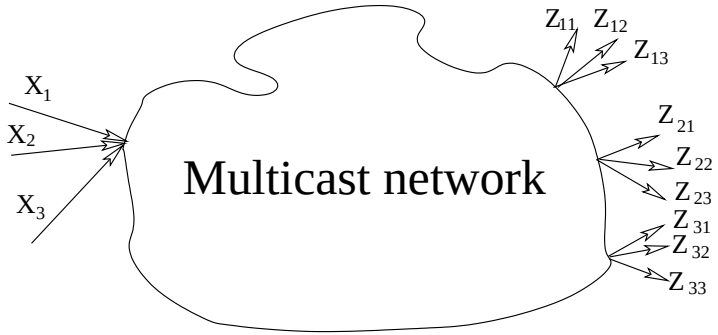


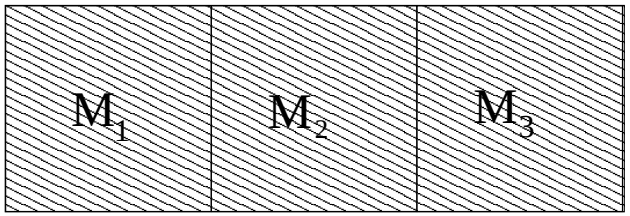
Multicast network



M is a $|\mathscr{X}(v)| \times K|\mathscr{X}(v)|$ matrix.

Multicast:





System Transfer matrix

$\mathscr{C} = \{(v, u_1, \mathscr{X}(v)), (v, u_2, \mathscr{X}(v)), \ldots, (v, u_K, \mathscr{X}(v))\}$

M is a $|\mathscr{X}(v)| \times K|\mathscr{X}(v)|$ matrix.

$m_i(\underline{\xi}) = \det(M_i(\underline{\xi}))$

Choose the coefficients in $\bar{\mathbb{F}}$ so that all $m_i(\underline{\xi})$ are unequal to zero.

Find a solution of $\prod_i m_i(\underline{\xi}) \neq 0$

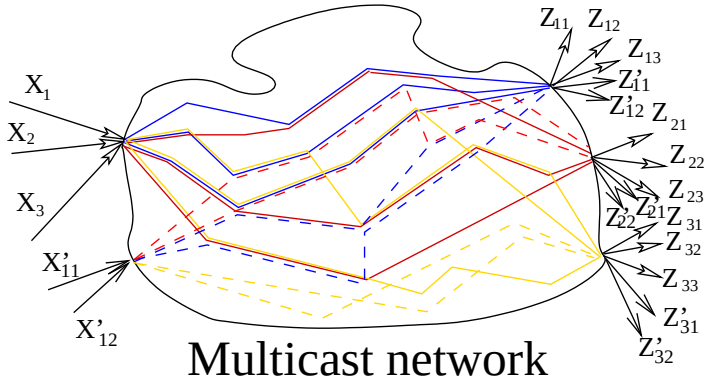
The main Multicast Theorem:

Theorem Let $(\mathcal{G}, \mathscr{C})$ be a multicast network coding problem. There exists a linear network coding solution for $(\mathcal{G}, \mathscr{C})$ over a finite field $\mathbb{F}_{2^m}$ for some large enough m if and only if there exists a flow of sufficient capacity between the source and each sink individually.

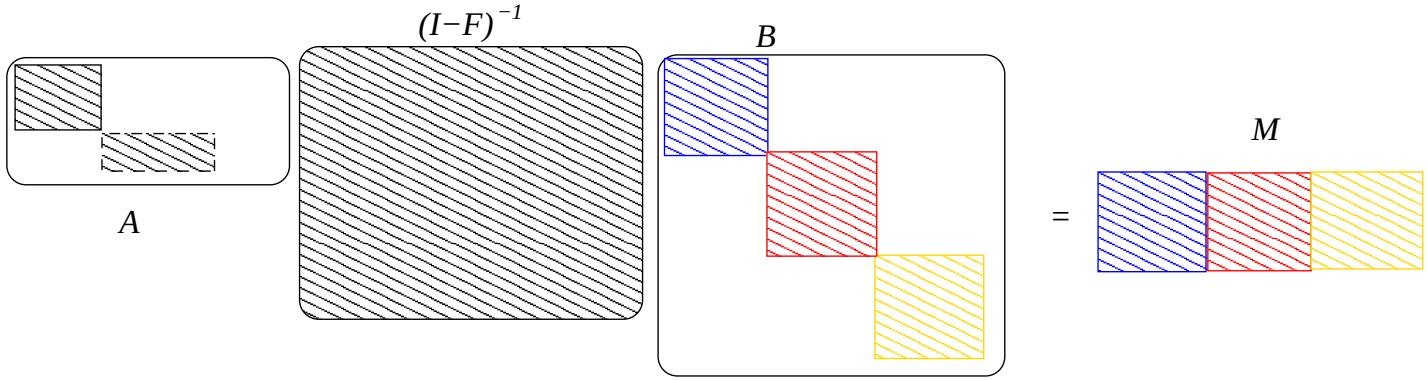
(We will see later how large m will have to be — it's not too bad)

Other (derived) problems: Multisource — Multicast

$\mathscr{C} = \{(v, u_1, \mathscr{X}(v)), (v, u_2, \mathscr{X}(v)), \ldots, (v, u_K, \mathscr{X}(v))\}$



Multicast network

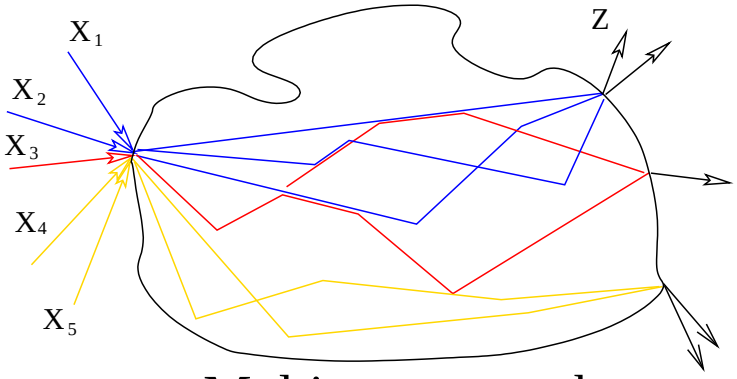


Other (derived) problems: Multisource — Multicast

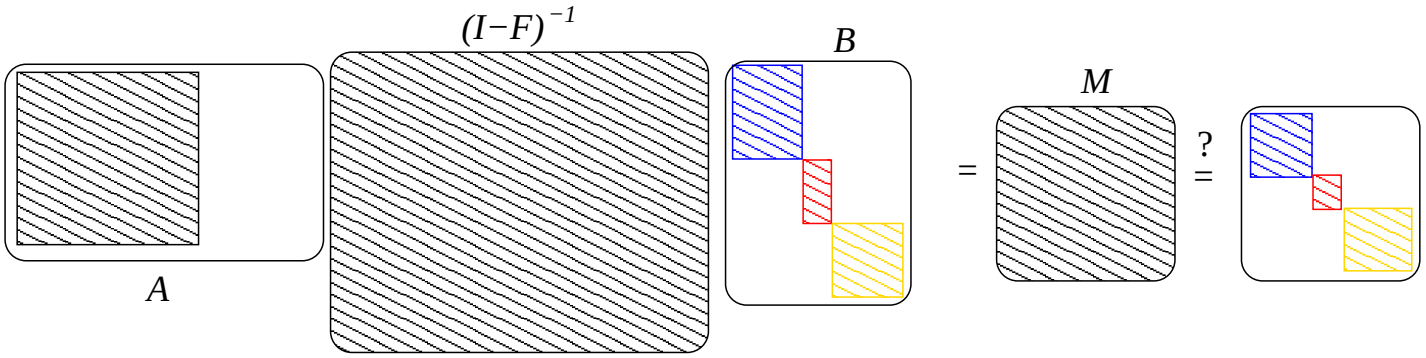
Theorem Let a linear, acyclic, delay-free network $\mathcal{G}$ be given with a set of desired connections $\mathscr{C} = \{(v_i, u_j, \mathscr{X}(v_i)) : i = 0, 1, \ldots N, j = 1, 2, \ldots K\}$. The network problem $(\mathcal{G}, \mathscr{C})$ is solvable if and only if the Min-Cut Max-Flow bound is satisfied for any cut between all source nodes $\{v_i : i = 0, 1, \ldots N\}$ and any sink node u_j.

Other (derived) problems: One source — Disjoint Muticasts

$\mathscr{C} = \{(v, u_j, \mathscr{X}(v, u_j)) : j = 1, 2, \ldots K\}$, $\mathscr{X}(v, u_j) \cap \mathscr{X}(v, u_i) = \emptyset$

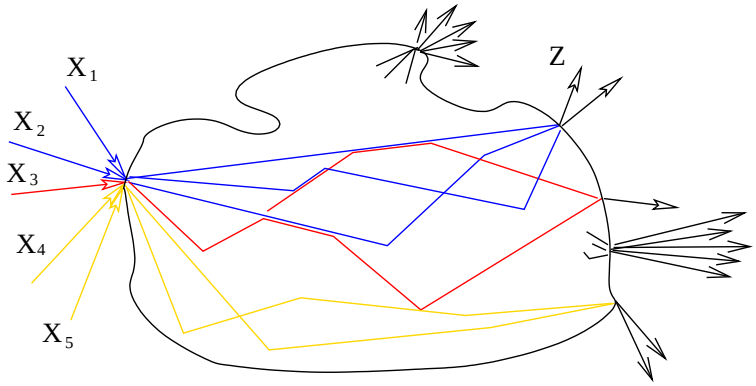


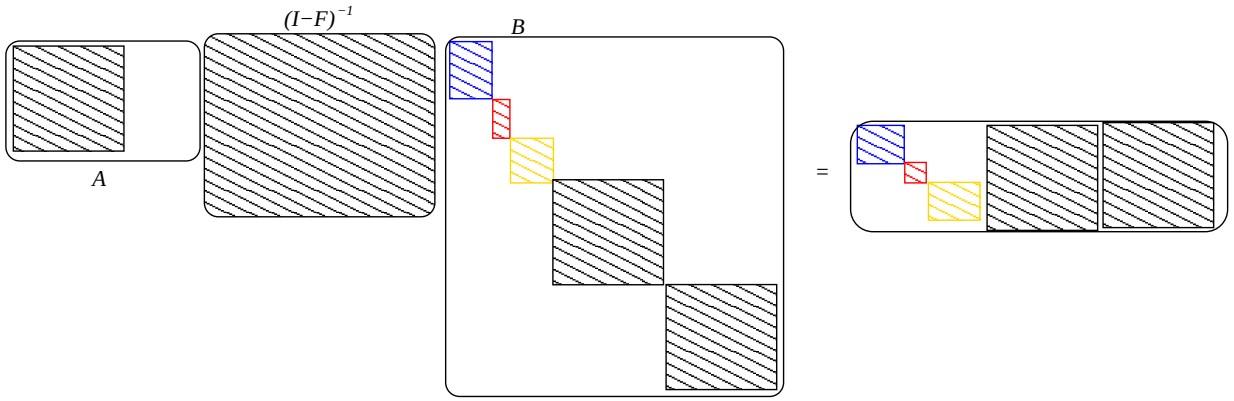
Multicast network



One source — Disjoint Muticasts + Multicasts

$\mathscr{C} = \{(v, u_j, \mathscr{X}(v, u_j)) : j = 1, 2, \ldots K\} \cup \{(v, u_\ell, \mathscr{X}(v)) : j = K+1, K+2, \ldots K+N\}$, $\mathscr{X}(v, u_j) \cap \mathscr{X}(v, u_i) = \emptyset$



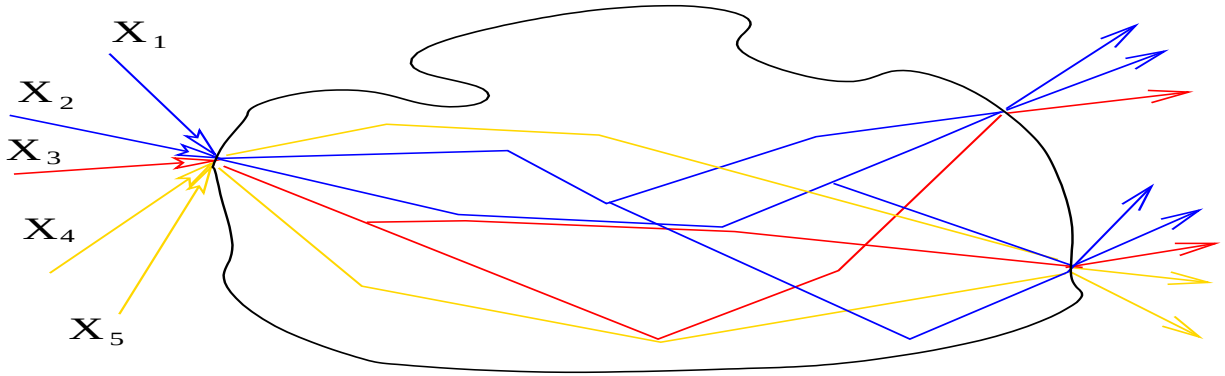


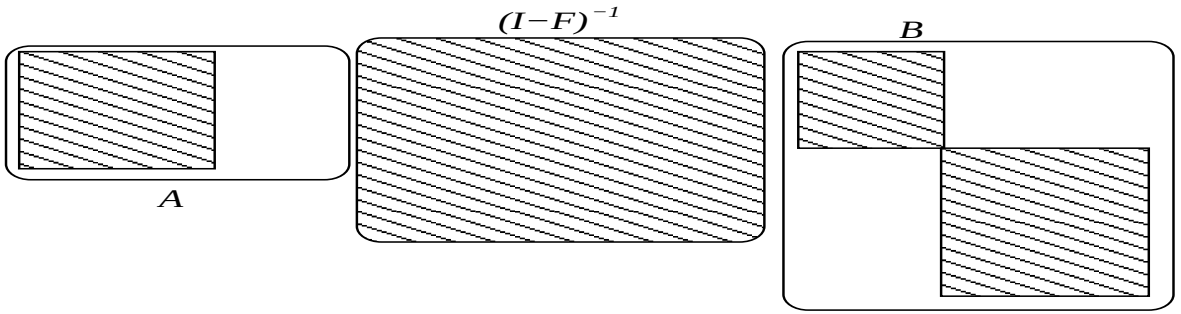
Multisource — Disjoint Muticasts + Multicast

Theorem Let a linear, acyclic, delay-free network $\mathcal{G}$ be given with a set of desired connections $\mathscr{C} = \{(v, u_j, \mathscr{X}(v, u_j)) : j = 1, 2, \ldots K\} \cup \{(v, u_\ell, \mathscr{X}(v)) : j = K+1, K+2, \ldots K+N\}$ such that collection of random processes $\mathscr{X}(v, u_j)$, $\mathscr{X}(v, u_j)$ are mutually disjoint for $i, j < K$, i.e. $\mathscr{X}(v, u_j) \cap \mathscr{X}(v, u_i) = \emptyset$ for $i \neq j$, $i, j \leq K$. The network problem is solvable if and only if the Min-Cut Max-Flow bound is satisfied between v and the set of sink nodes $\{u_1, u_2, \ldots, u_K\}$ at a rate $|\mathscr{X}(v)|$ and between v and u_ℓ, $\ell > K$ also at a rate $|\mathscr{X}(v)|$.

Other (derived) problems: Two level Multicasts

$\mathscr{C} = \{(v, u_1, \mathscr{X}(v, u_1))\} \cup \{(v, u_2, \mathscr{X}(v)\}$





Other (derived) problems: Two Level Multicast

Theorem("Two-level multicast") Let an acyclic network $\mathcal{G}$ be given with a set of desired connections

$$\mathscr{C} = \{(v, u_1, \mathscr{X}(v, u_1)), (v, u_2, \mathscr{X}(v))$$

The network problem is solvable if and only if the Min-Cut Max-Flow bound is satisfied between v and u_1 at a rate $|\mathscr{X}(v, u_1)|$ and between v and u_2 at a rate $|\mathscr{X}(v)|$.

So far so good!

What about networks with cycles?

What about networks with delays?

What about robustness?

Do we really need network coding for multicast?

So far so good!

What about networks with cycles?

What about networks with delays?

What about robustness?

Do we really need network coding for multicast? YES

Robust multicast:

Links in the network may fail. (non-ergodic). Set of failure patterns:$\mathscr{F}$

A network solution is static w.r.t. $\mathscr{F}$ if the operations in the network interior are oblivious to the particular failure in $\mathscr{F}$.

Theorem Let $(\mathcal{G}, \mathscr{C})$ be a multicast network coding problem and let $\mathscr{F}$ be the set of failure patterns such that the problem is solvable. There exists a common static solution to all failure patterns in $\mathscr{F}$.

Proof sketch: All we have to do is to guarantee that the product of all determinants of all scenarios in $\mathscr{F}$ evaluates to a non zero value.

Theorem Let $(\mathcal{G}, \mathscr{C})$ be a multicast network coding problem and let $\mathscr{F}$ be the set of failure patterns such that the problem is solvable.. There exists a solution for $(\mathcal{G}, \mathscr{C})$ over a finite field $\mathbb{F}_{2^m}$ with $m \leq \lceil \log_2(|\mathscr{F}|NR + 1) \rceil$.

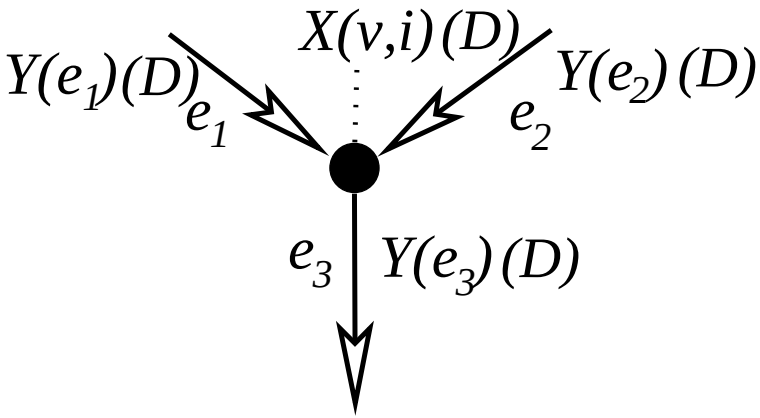
$\vdots$

Linear Networks with Delays

We transmit random processes in a delay variable D on links, i.e.

$$\begin{aligned} X(v,j)(D) &= \sum_{\ell=0}^{\infty} X_\ell(v,j) D^\ell, \\ Z(v,j)(D) &= \sum_{\ell=0}^{\infty} Z_\ell(v,j) D^\ell, \\ Y(e)(D) &= \sum_{\ell=0}^{\infty} Y_\ell(e) D^\ell. \end{aligned}$$

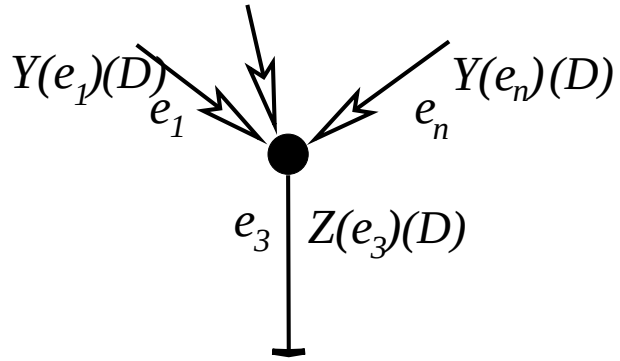
Conceptually, we consider an entire sequence in D as one symbol and work over the field of formal power series.



$$Y(e_3)(D) = \sum_i \alpha_i DX(v,i)(D) + \sum_{j=1,2} \beta_j DY(e_j)(D)$$

(other functions with memory are possible but not necessary)

At a receiver (terminal) node we have to allow for "rational" functions:

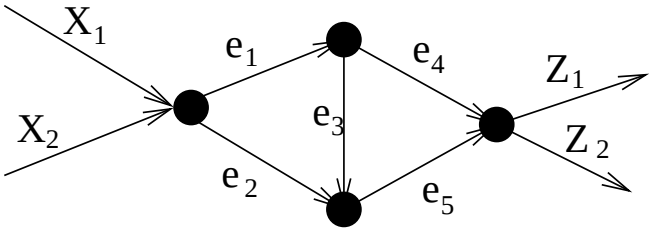


$$Y(e)(D) = \sum_{\ell=0}^{\infty} Y_\ell(e) D^\ell, \quad Z(v,j)(D) = \sum_{\ell=0}^{\infty} Z_\ell(v,j) D^\ell$$

$$Z_\ell(v,j) = \sum_{j=1}^{n} \sum_{k=0}^{\mu} \varepsilon_{j,k} Y_{\ell-k}(e_j) + \sum_{k=1}^{\mu} \lambda_k Z_{\ell-k}(v,j)$$

or $$Z(v,j)(D) = \sum_{j=1}^{n} \frac{\varepsilon_{j,k}(D)}{\lambda(D)} Y(e_j)(D)$$

The transfer matrix with delays



$$F = \begin{pmatrix} 0 & 0 & D\beta_{e_1,e_3} & D\beta_{e_1,e_4} & 0 \\ 0 & 0 & 0 & 0 & D\beta_{e_2,e_5} \\ 0 & 0 & 0 & 0 & D\beta_{e_3,e_5} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Summing the "path gains":

$$P = I + DF + D^2F^2 + \ldots = (I - DF)^{-1} = \begin{pmatrix} 0 & 0 & D\beta_{e_1,e_3} & D\beta_{e_1,e_4} & D^2\beta_{e_1,e_3}\beta_{e_3,e_5} \\ 0 & 0 & 0 & 0 & D\beta_{e_2,e_5} \\ 0 & 0 & 0 & 0 & D\beta_{e_3,e_5} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Observe that $G = (I - DF)^{-1}$ is polynomial over $\mathbb{F}_2(D)$.

An algebraic Min-Cut Max-Flow condition with delays

Let network be given with a source v and a sink v'. The following three statements are equivalent:

1. A point-to-point connection $c = (v, v', \mathscr{X}(v, v'))$ is possible.

2. The Min-Cut Max-Flow bound is satisfied for a rate $R(c) = |\mathscr{X}(v, v')|$.

3. The determinant of the $R(c) \times R(c)$ transfer matrix M is nonzero over the ring of polynomials $\mathbb{F}_2(D)[\underline{\xi}]$ with coefficients from the field of rational functions.

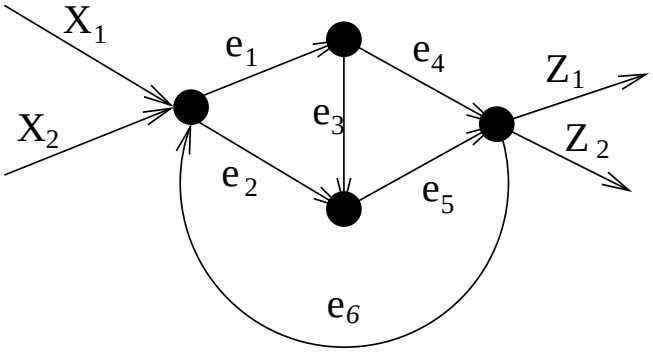
It is only that....

We have to study the solution sets of polynomial equations over $\mathbb{F}_2(D)$.

At receiver nodes we have to allow for memory and the possibility of implementing rational functions!

This is neccessary since now we have to invert a transfer matrix which has as elements polynomials over $\mathbb{F}_2(D)$.

The transfer matrix with delays and cycles



$$F = \begin{pmatrix} 0 & 0 & D\beta_{e_1,e_3} & D\beta_{e_1,e_4} & 0 & \\ 0 & 0 & 0 & 0 & D\beta_{e_2,e_5} & \\ 0 & 0 & 0 & 0 & D\beta_{e_3,e_5} & \\ 0 & 0 & 0 & 0 & 0 & D\beta \\ 0 & 0 & 0 & 0 & 0 & D\beta \\ D\beta_{e_6,e_1} & D\beta_{e_6,e_2} & 0 & 0 & 0 & \end{pmatrix}$$

Summing the "path gains":

$P = I + DF + D^2F^2 + \ldots = (I - DF)^{-1} =$ (6 × 6 matrix with rational coefficients)

Now $G = (I - DF)^{-1}$ is rational over $\mathbb{F}_2(D)$.

Delays and cycle - or really nothing has happened....

Let network be given with a source v and a sink v'. The following three statements are equivalent:

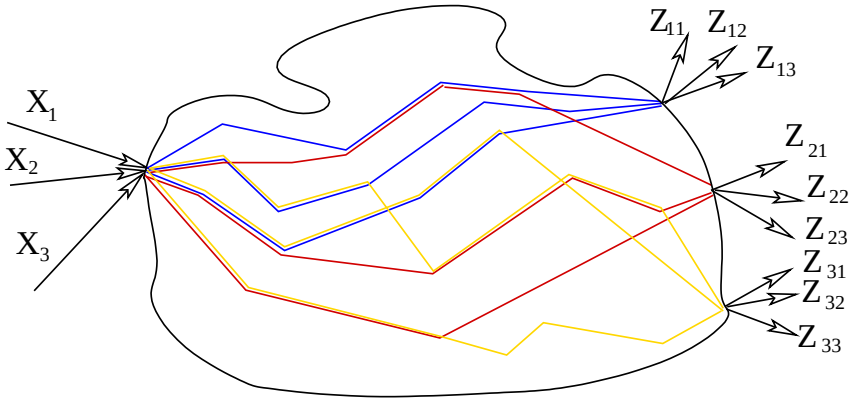
1. A point-to-point connection $c = (v, v', \mathscr{X}(v, v'))$ is possible.

2. The Min-Cut Max-Flow bound is satisfied for a rate $R(c) = |\mathscr{X}(v, v')|$.

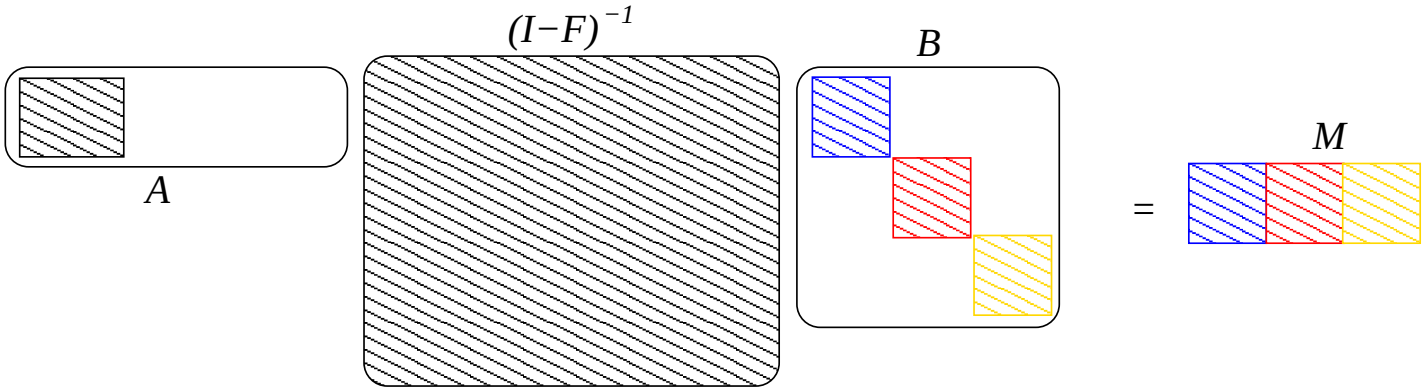
3. The determinant of the $R(c) \times R(c)$ transfer matrix M is nonzero over the ring of polynomials $\mathbb{F}_2(D)[\underline{\xi}]$ with coefficients from the field of rational functions.

Multicast:

$\mathscr{C} = \{(v, u_1, \mathscr{X}(v)), (v, u_2, \mathscr{X}(v)), \ldots, (v, u_K, \mathscr{X}(v))\}$

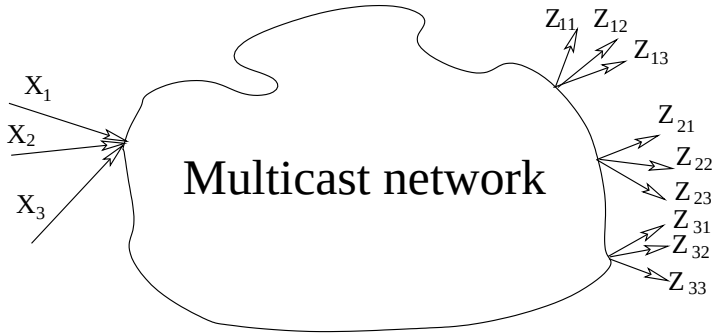


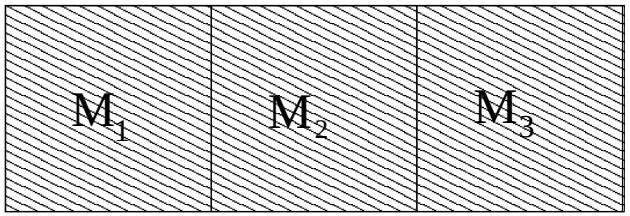
Multicast network



M is a $|\mathscr{X}(v)| \times K|\mathscr{X}(v)|$ matrix.

Multicast:





System Transfer matrix

$\mathscr{C} = \{(v, u_1, \mathscr{X}(v)), (v, u_2, \mathscr{X}(v)), \ldots, (v, u_K, \mathscr{X}(v))\}$

M is a $|\mathscr{X}(v)| \times K|\mathscr{X}(v)|$ matrix.

$m_i(\underline{\xi}) = \det(M_i(\underline{\xi}))$

Choose the coefficients in $\bar{\mathbb{F}}$ so that all $m_i(\underline{\xi})$ are unequal to zero.

Find a solution of $\prod_i m_i(\underline{\xi}) \neq 0$

The main Multicast Theorem:

Theorem Let $(\mathcal{G}, \mathscr{C})$ be a multicast network coding problem on a graph which may have a cyclic structure. There exists a linear network coding solution for $(\mathcal{G}, \mathscr{C})$ over a finite field $\mathbb{F}_{2^m}$ for some large enough m if and only if there exists a flow of sufficient capacity between the source and each sink individually.

Theorems, Theorems.....

Summary

- Connecting network information flow problems to algebraic equations yields powerful tools for analysis of networks.

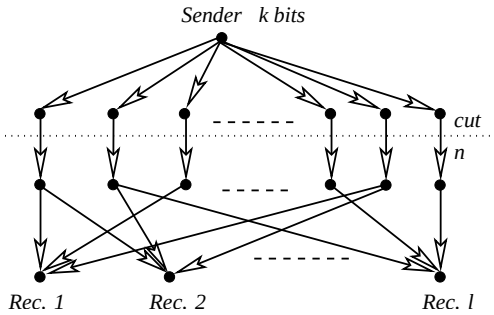
- Multicast especially well suited for the approach since we have to find "non solutions" to equations, which can easily be accomplished in large fields.

- Many network scenarios can be derived from the multicast setup.

- The general non multicast setup will be treated later (much less is known).

- Field size?
- How do we find solutions?
- Is network coding really helpful or just a singular occurrence?

III — More about the multicast



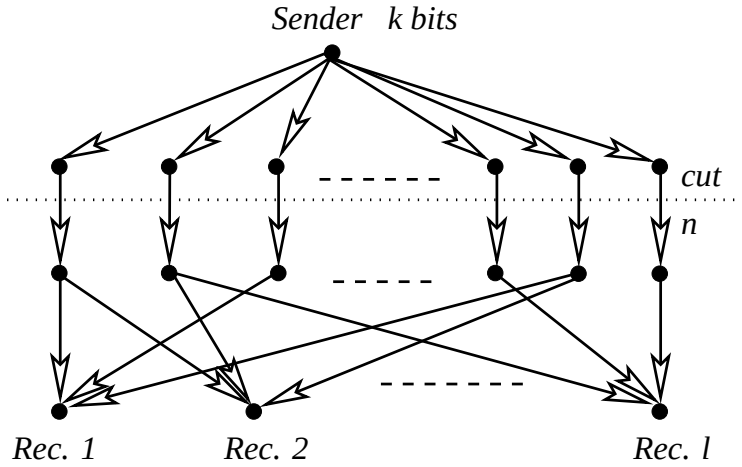
Questions

- Do we really need codes?
- What alphabet sizes do we need?
- How do we find solutions?
- Bidirectional links?

Do we really need codes?

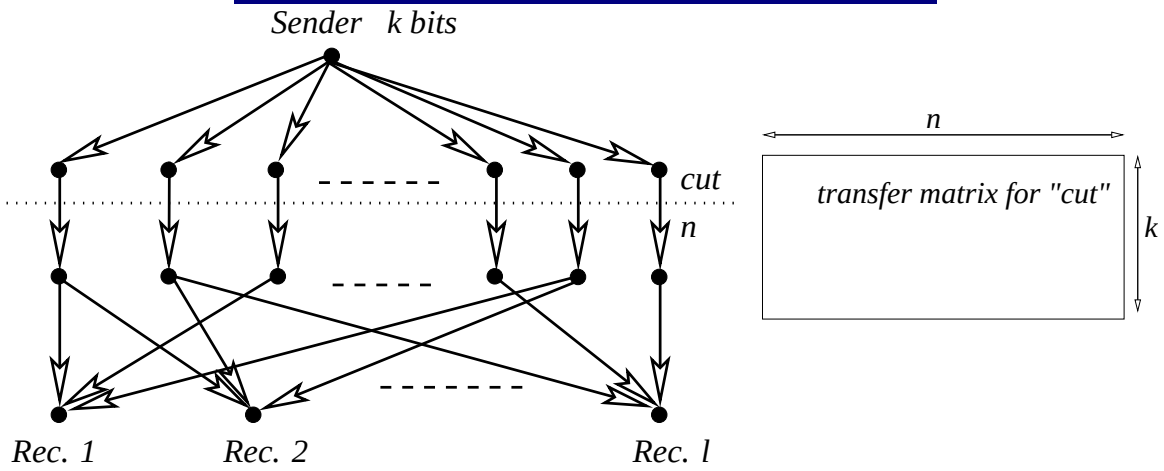
Is the performance gap between the network coding solution and routing, i.e. packing directed Steiner trees bounded ?

A simple example:



Each receiver picks out one of $\binom{n}{k}$ possible middle layer links

Do we really need codes?



The transfer matrix from the transmitter to the "cut" has to satisfy that $k \times k$ submatrix has full rank!

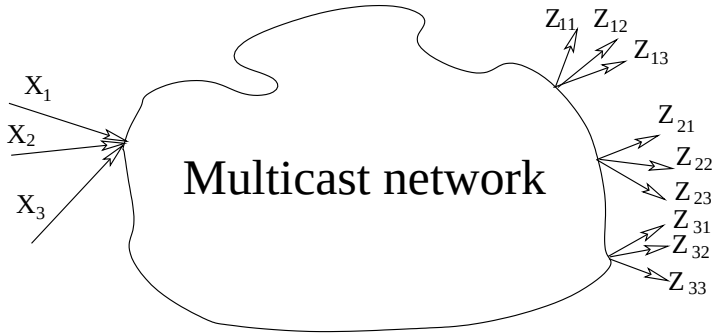
$\Rightarrow$ The field size is at least in the same order as n
(the MDS conjecture)

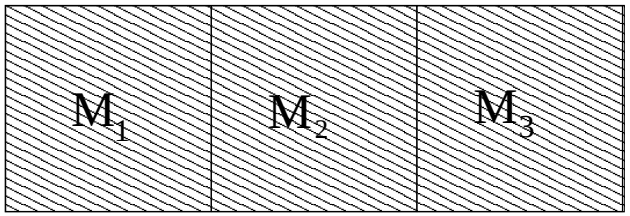
A lower bound

Theorem There exist multicast problems with T receivers such that the minimum field size required for a solution grows as $O(\sqrt{T})$.

Theorem There exist multicast problems such that the gap between routing and network coded strategies is arbitrarily large.

How do we find solutions for the Multicast?





System Transfer matrix

$\mathscr{C} = \{(v, u_1, \mathscr{X}(v)), (v, u_2, \mathscr{X}(v)), \ldots, (v, u_K, \mathscr{X}(v))\}$

M is a $|\mathscr{X}(v)| \times K|\mathscr{X}(v)|$ matrix.

$m_i(\underline{\xi}) = \det(M_i(\underline{\xi}))$

Choose the coefficients in $\bar{\mathbb{F}}$ so that all $m_i(\underline{\xi})$ are unequal to zero.

The degree of each $m_i(\underline{\xi})$ is at most $|\mathscr{X}(v)|$

Multicast:

An algorithm to find a vector $\underline{a}$ such that $F(\underline{a}) \neq 0$ for a polynomial F.

Input: A polynomial F in indeterminates $\xi_1, \xi_2, \ldots, \xi_n$, integers: $i = 1, t = 1$

Iteration:

1. Find the maximal degree δ of F in any variable ξ_j and let i be the smallest number such that $2^i > \delta$.

2. Find an element a_t in $\mathbb{F}_{2^i}$ such that $F(\underline{\xi})|_{\xi_t=a_t} \neq 0$ and let $F \leftarrow F(\underline{\xi})|_{\xi_t=a_t}$.

3. If $t = n$ then halt, else $t \leftarrow t + 1$, goto 2).

Output: $(a_1, a_2, \ldots, a_n)$.

The crucial step is 2) which is successful if the fieldsize is larger than the degree of F.

Multicast:

Let $(\mathcal{G}, \mathscr{C})$ be a multicast network coding problem with T receivers and R symbols transmitted per time unit. There exists a solution for $(\mathcal{G}, \mathscr{C})$ over a finite field $\mathbb{F}_{2^m}$ with

$m \leq \lceil \log_2(TR + 1) \rceil$.

(A more careful analysis shows that a field $\mathbb{F}_{2^m}$ with $m \leq \lceil \log_2(T) \rceil$ or $\mathbb{F} \geq T$)

Multicast:

For any multicast networking problem with T receivers there always exists a solution over an alphabet which is at least as large as T.

Conversely:

There exist multicast networking problems with T receivers such that the minimum alphabet size is bounded below by $\sqrt{T} - o(1)$.

(In practice - just try the random approach...)

A different approach...

S.-Y. R. Li, R. W. Yeung, and N. Cai. "Linear network coding". IEEE Transactions on Information Theory , Februray, 2003

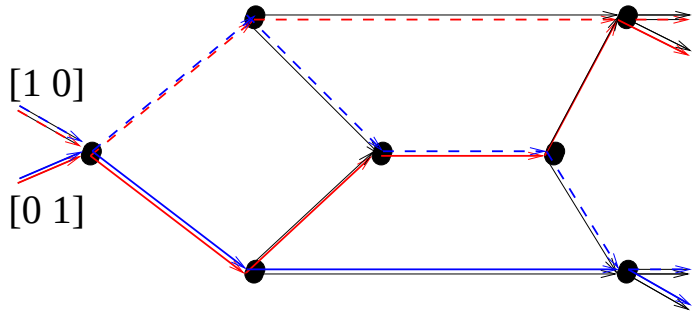
S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction,"IEEE Transactions on Information Theory. Submitted July 2003.

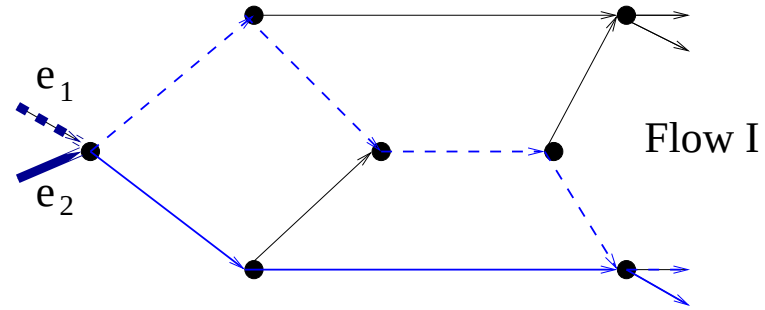
A flow based approach that carefully constructs a solution in polynomial time.

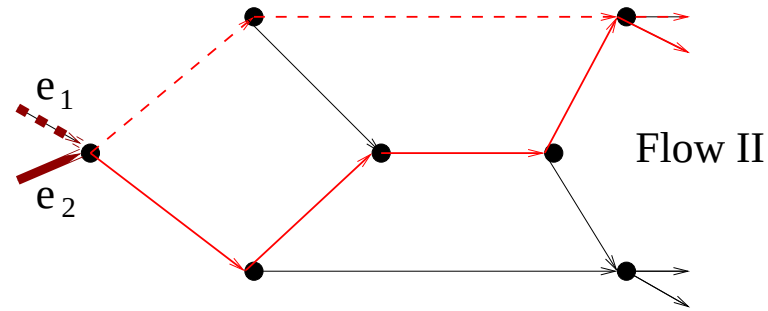
A different approach...

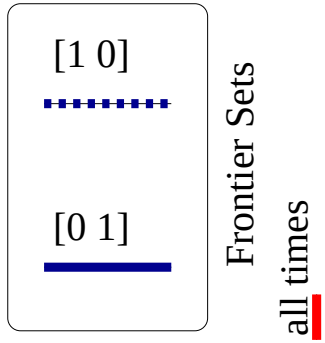
A solution for acyclic networks is constructed "one link at a time" starting at the source.

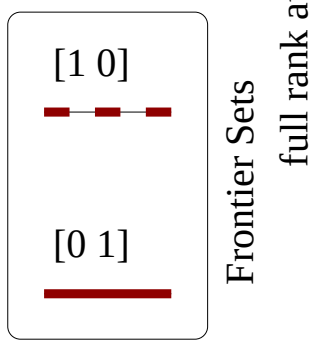
Each flow to a receiver is being treated as a set of disjoint paths with the set of edges that was processed last (the frontier set) having to form a full rank matrix



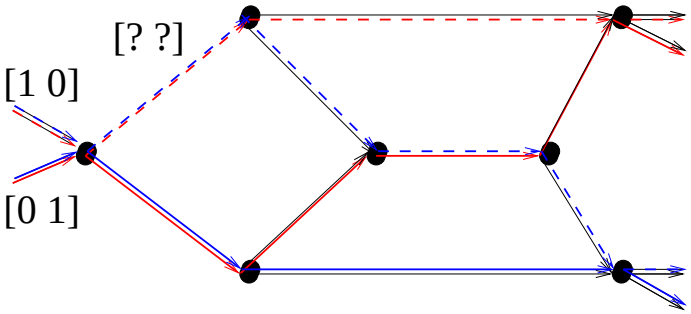


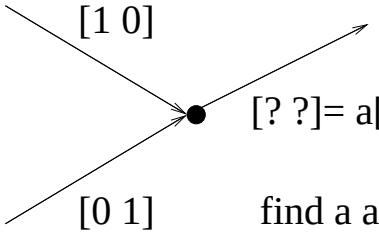






full rank at all times

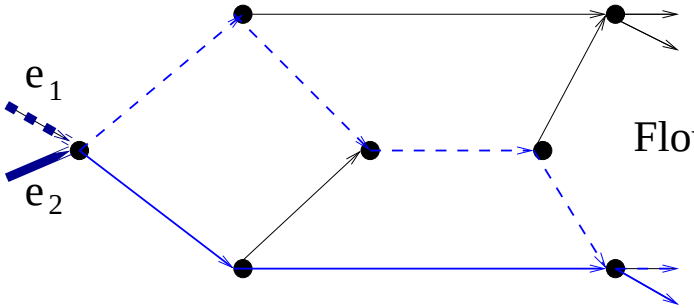


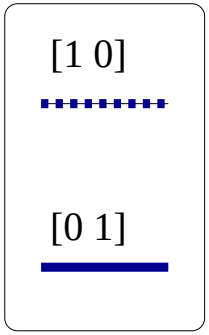


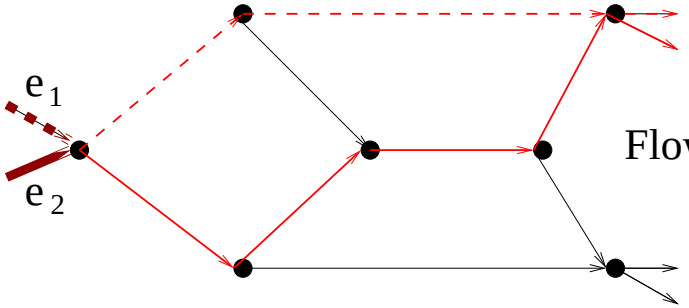
[? ?]= a[1 0] + b[0 1]

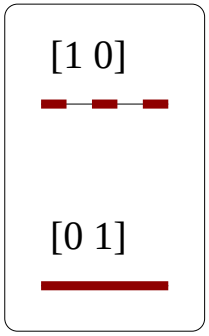
find a and b such that we have

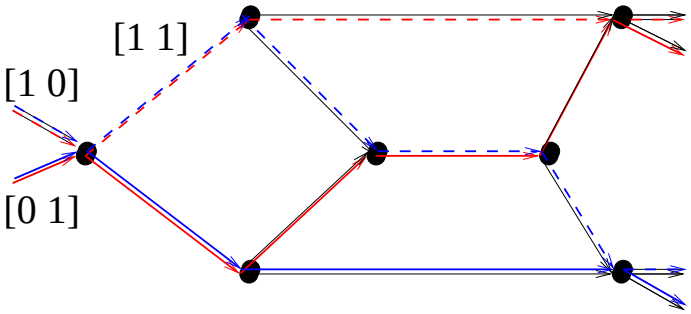
full rank at all times

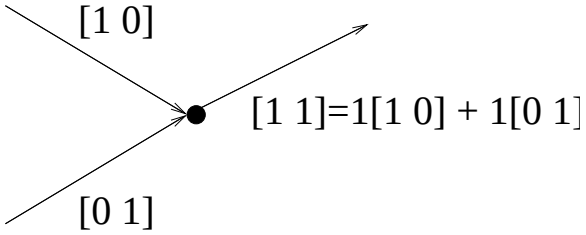


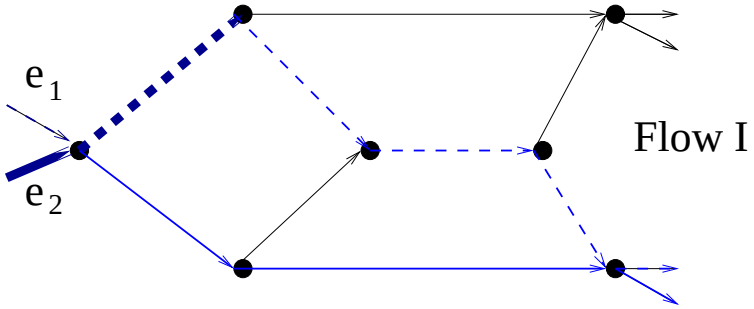


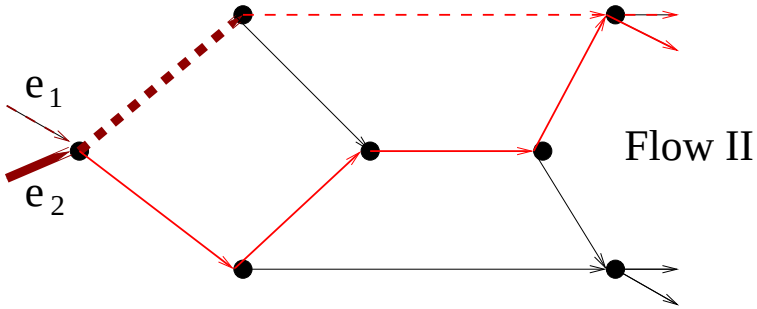


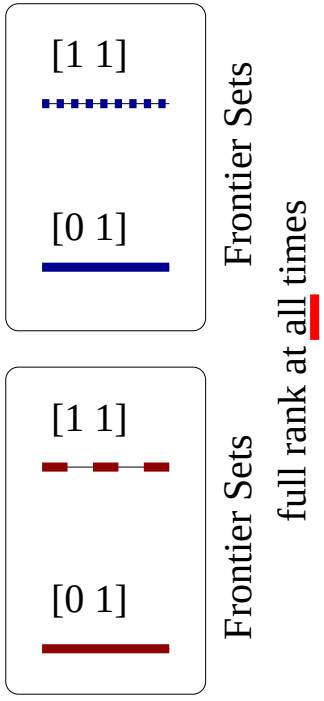


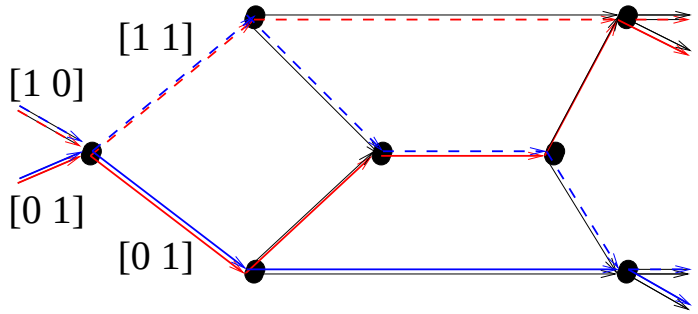

[1 1]
[1 0]
[0 1]

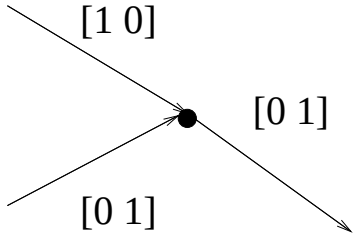

[1 0]
[1 1]=1[1 0] + 1[0 1]
[0 1]

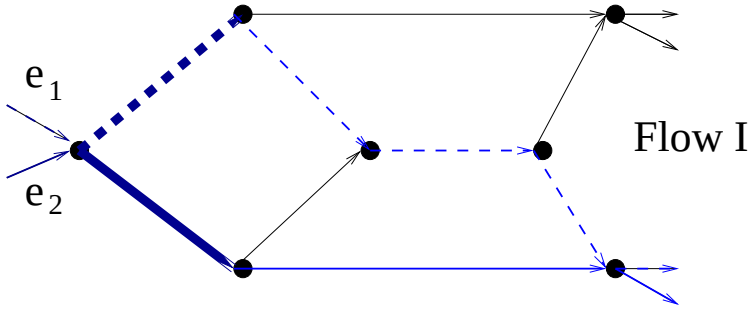

e 1
e 2
Flow I

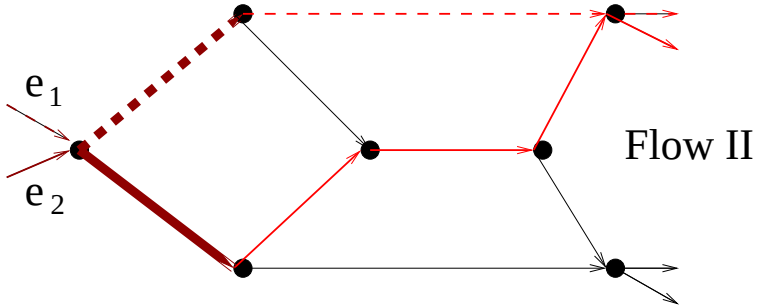

e 1
e 2
Flow II

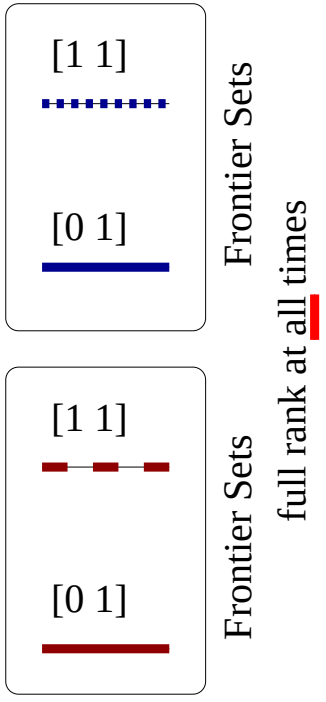

[1 1]
[0 1]
Frontier Sets
[1 1]
[0 1]
Frontier Sets
full rank at all times

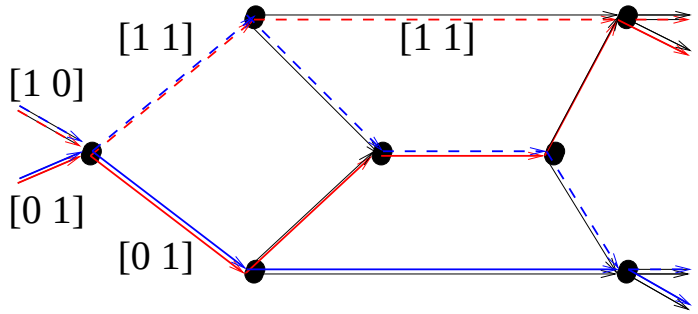

[1 1]
[1 0]
[0 1]
[0 1]

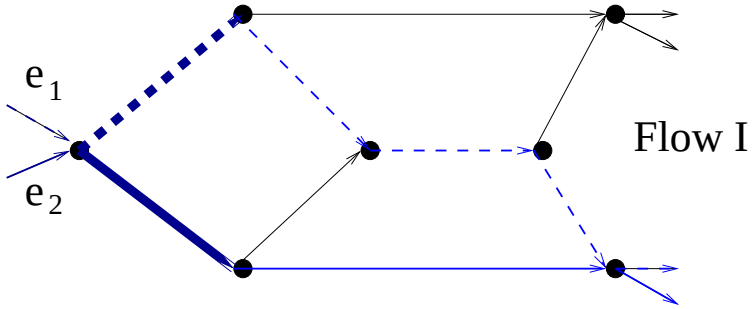

[1 0]
[0 1]
[0 1]

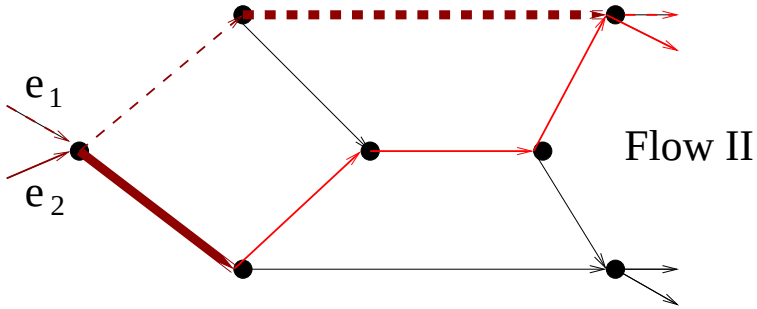

e 1
e 2
Flow I

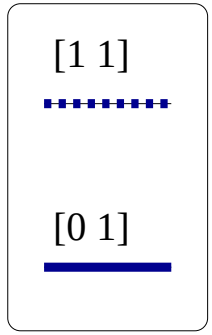

e 1
e 2
Flow II


[1 1]
[0 1]
Frontier Sets
[1 1]
[0 1]
Frontier Sets
full rank at all times

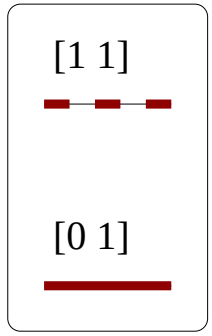






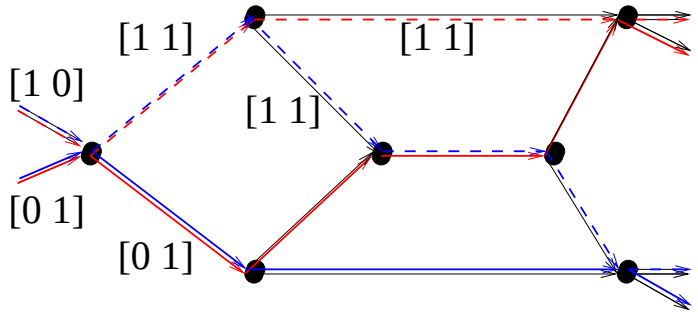


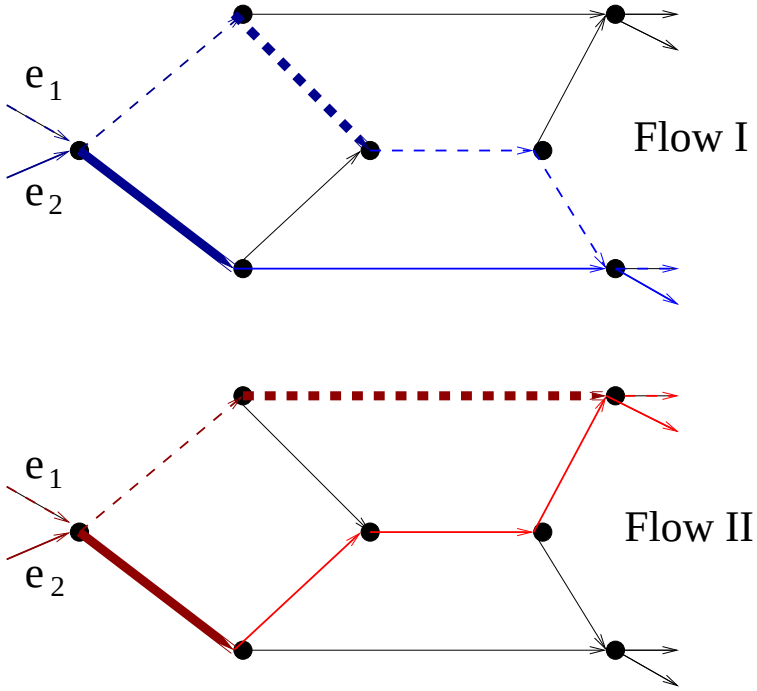
Frontier Sets

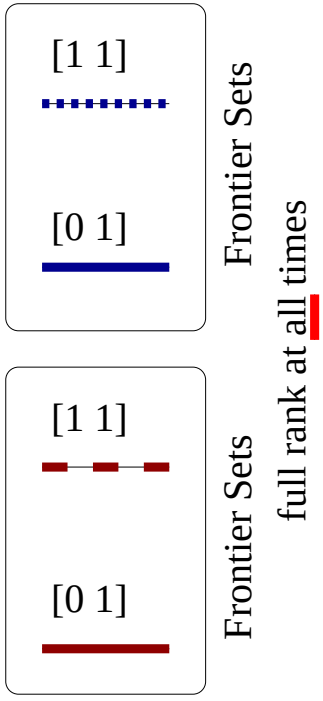


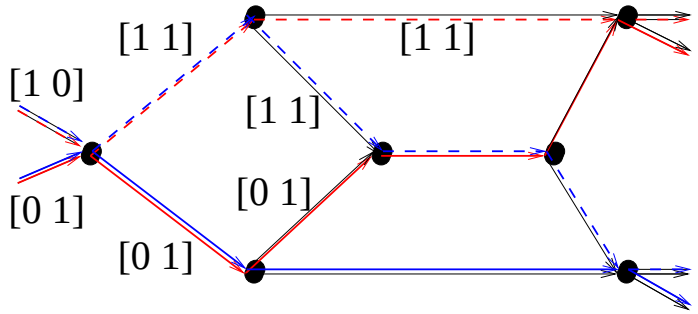
Frontier Sets

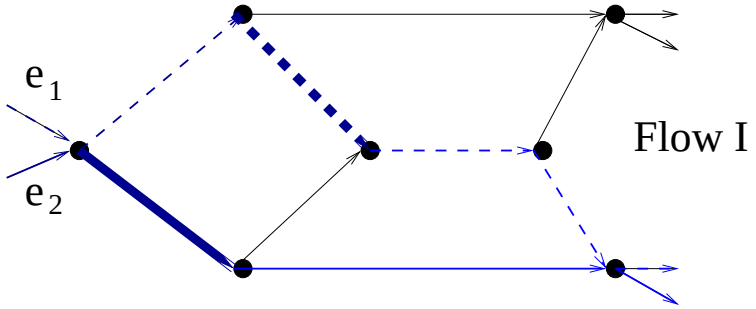
full rank at all times

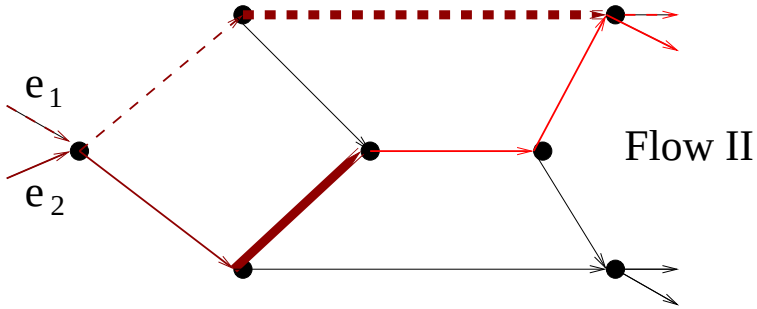

[1 1]
[1 1]
[1 0]
[1 1]
[0 1]
[0 1]

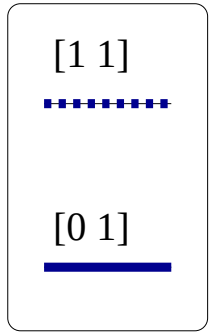

e 1
e 2
Flow I
e 1
e 2
Flow II


[1 1]
[0 1]
Frontier Sets
[1 1]
[0 1]
Frontier Sets
full rank at all times

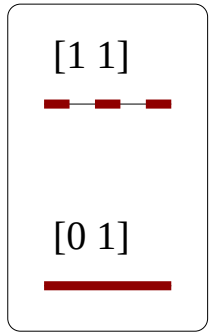






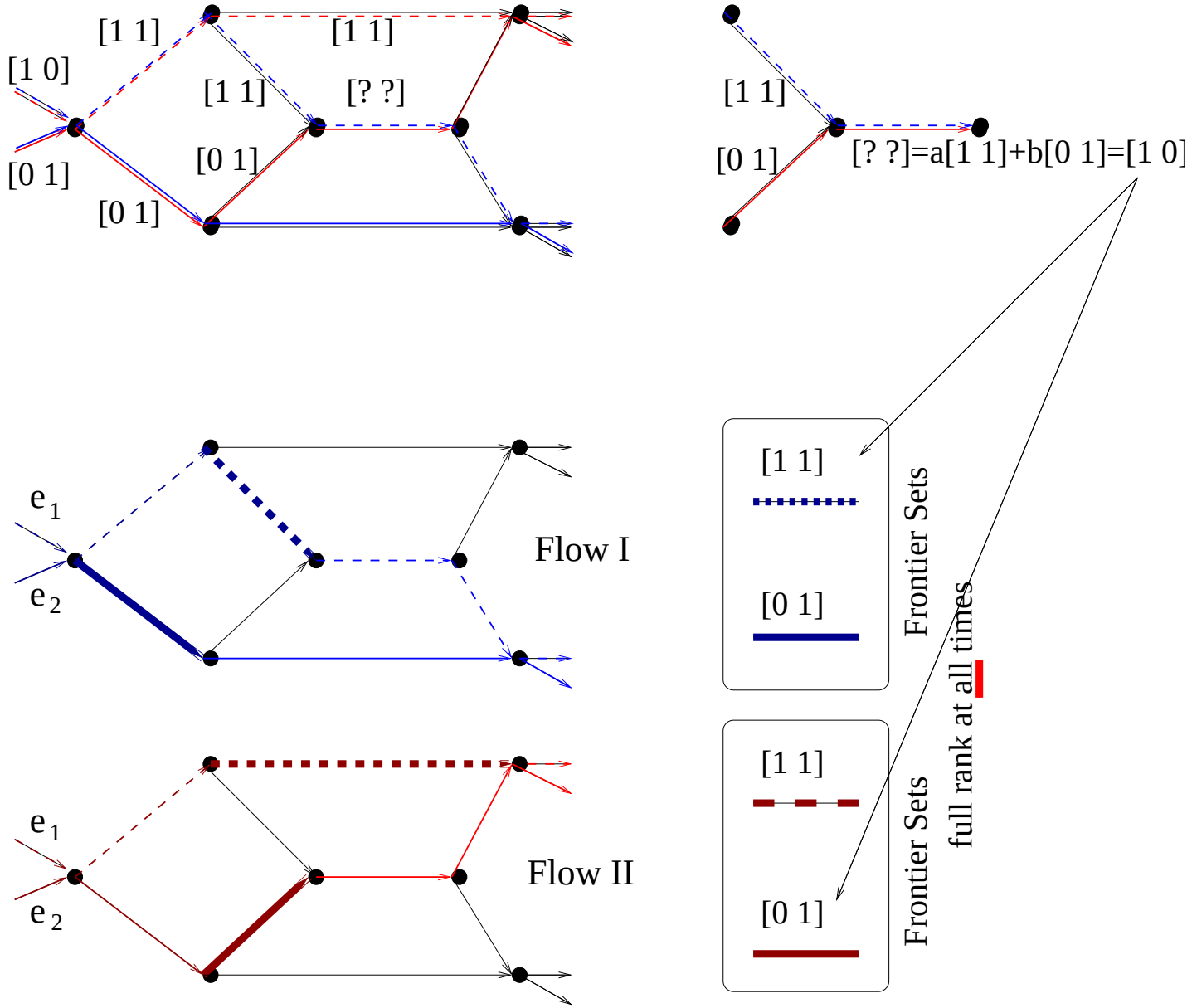


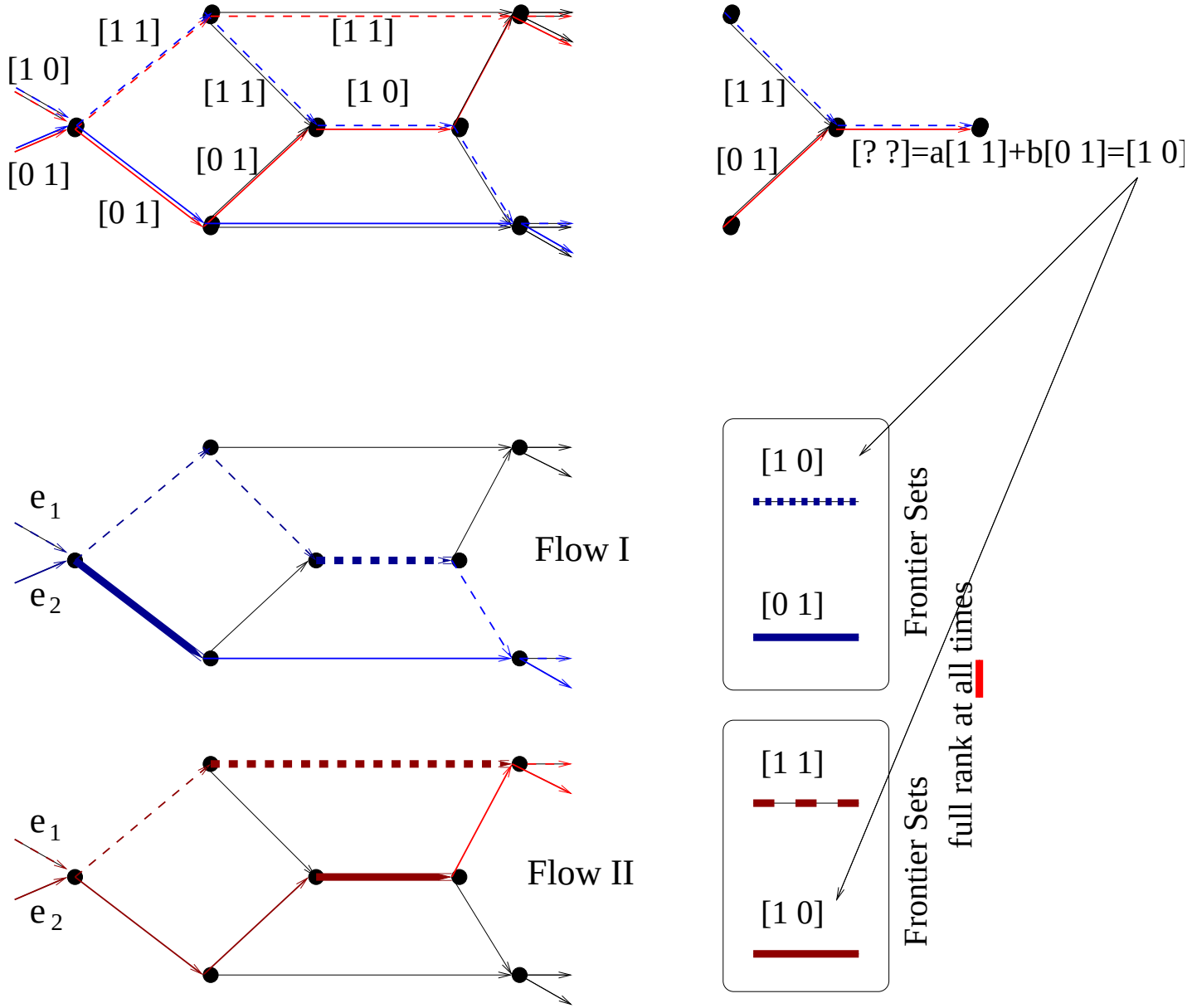
Frontier Sets

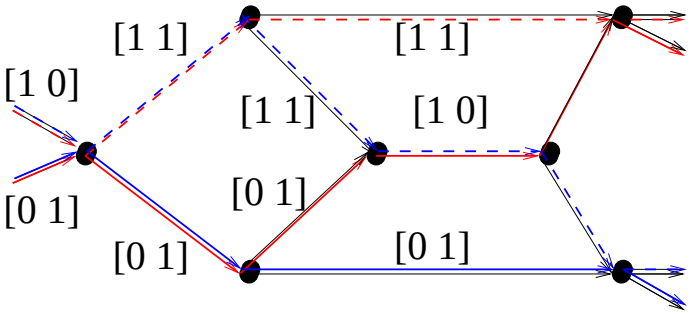


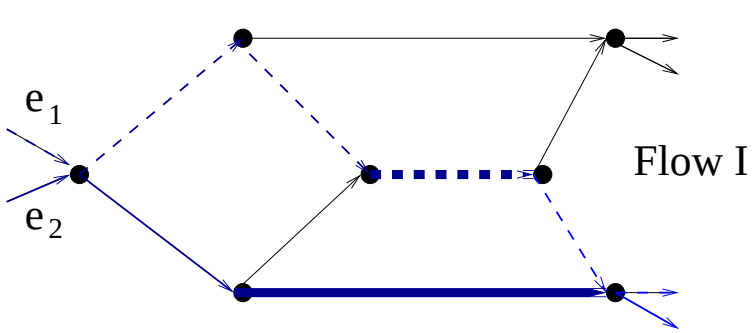
Frontier Sets

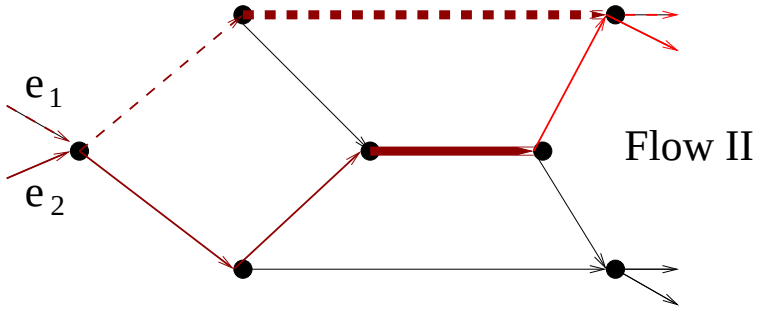
full rank at all times

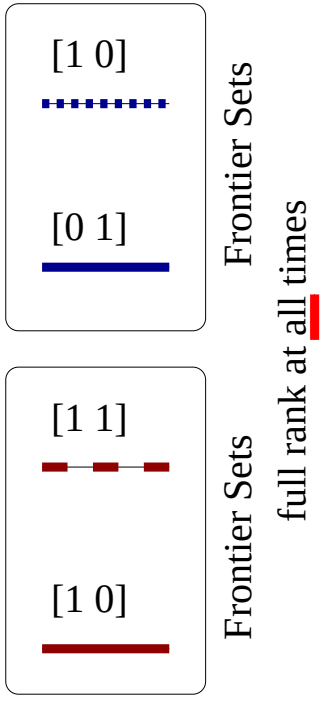

[1 1]
[1 1]
[1 0]
[1 1]
[? ?]
[0 1]
[0 1]
[0 1]
[1 1]
[0 1]
[? ?]=a[1 1]+b[0 1]=[1 0]
e 1
e 2
Flow I
[1 1]
[0 1]
Frontier Sets
full rank at all times
e 1
e 2
Flow II
[1 1]
[0 1]
Frontier Sets

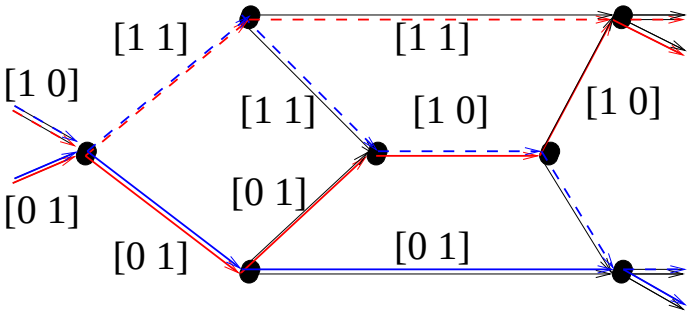

[1 1]
[1 1]
[1 0]
[1 1]
[1 0]
[0 1]
[0 1]
[0 1]
[1 1]
[0 1]
[? ?]=a[1 1]+b[0 1]=[1 0]
e 1
e 2
Flow I
[1 0]
[0 1]
Frontier Sets
full rank at all times
e 1
e 2
Flow II
[1 1]
[1 0]
Frontier Sets

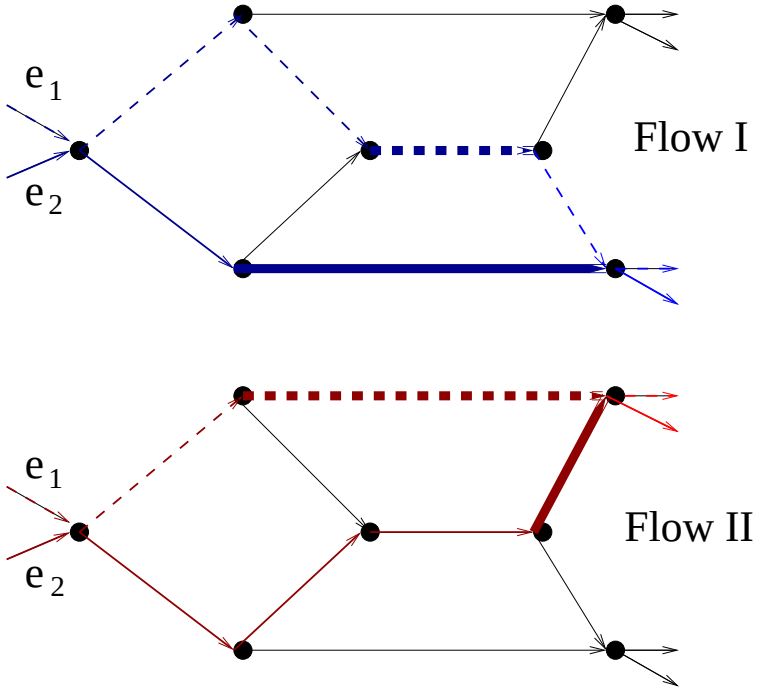

[1 1]
[1 1]
[1 0]
[1 1]
[1 0]
[0 1]
[0 1]
[0 1]
[0 1]

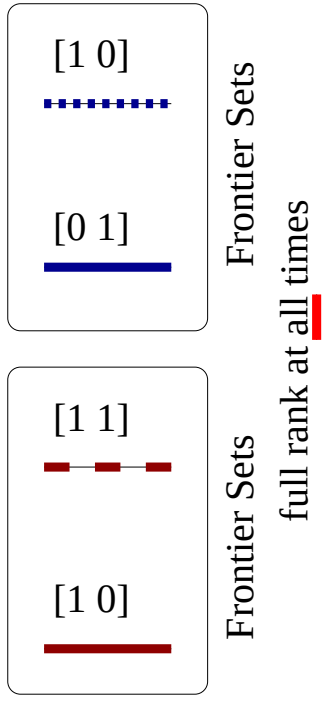

e 1
e 2
Flow I

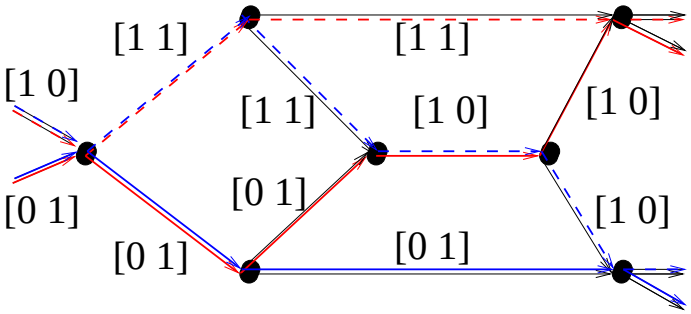

e 1
e 2
Flow II

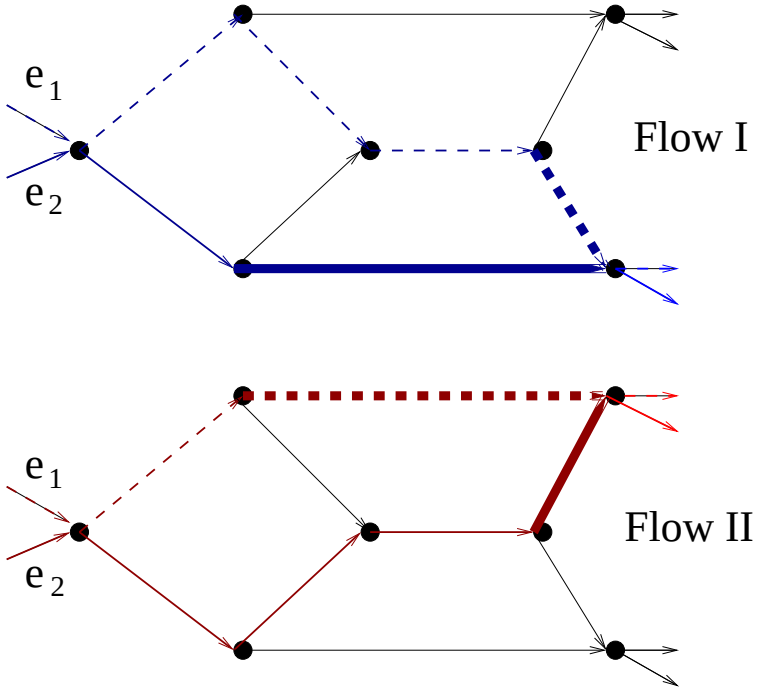

[1 0]
[0 1]
Frontier Sets
[1 1]
[1 0]
Frontier Sets
full rank at all times

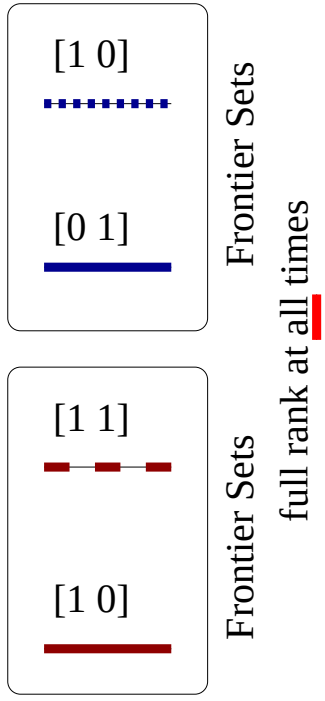

[1 1]
[1 1]
[1 0]
[1 1]
[1 0]
[1 0]
[0 1]
[0 1]
[0 1]
[0 1]


e 1
e 2
Flow I
e 1
e 2
Flow II

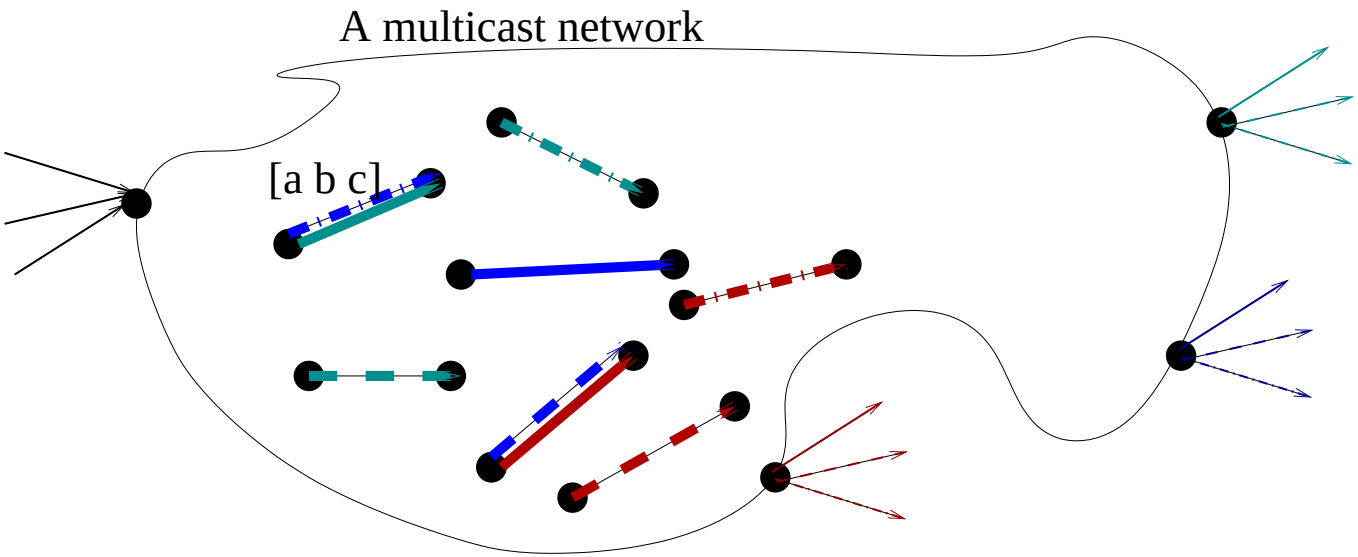

[1 0]
[0 1]
Frontier Sets
[1 1]
[1 0]
Frontier Sets
full rank at all times


[1 1]
[1 1]
[1 0]
[1 1]
[1 0]
[1 0]
[0 1]
[0 1]
[1 0]
[0 1]
[0 1]


e 1
e 2
Flow I
e 1
e 2
Flow II


[1 0]
[0 1]
Frontier Sets
[1 1]
[1 0]
Frontier Sets
full rank at all times

The algorithm of Jaggi, Sanders et al.



The frontier sets of a multicast to three receivers

[a b c]

[a’ b’ c’]

[a’’ b’’ c’’]

[a b c]
[a’ b’ c’]
[a’’ b’’ c’’]

has full rank
for all colors

The algorithm of Jaggi, Sanders et al.

Theorem [Jaggi, Sanders, et al] Let $(\mathcal{G}, \mathscr{C})$ be a multicast network coding problem with E edges, R symbols to be transmitted simultaneously and T receivers. There exists a linear network coding solution for $(\mathcal{G}, \mathscr{C})$ over a finite field $\mathbb{F}$ if $|\mathbb{F}| > T$. Moreover this solution can be found in time $O(E \cdot T \cdot R(R+T))$.

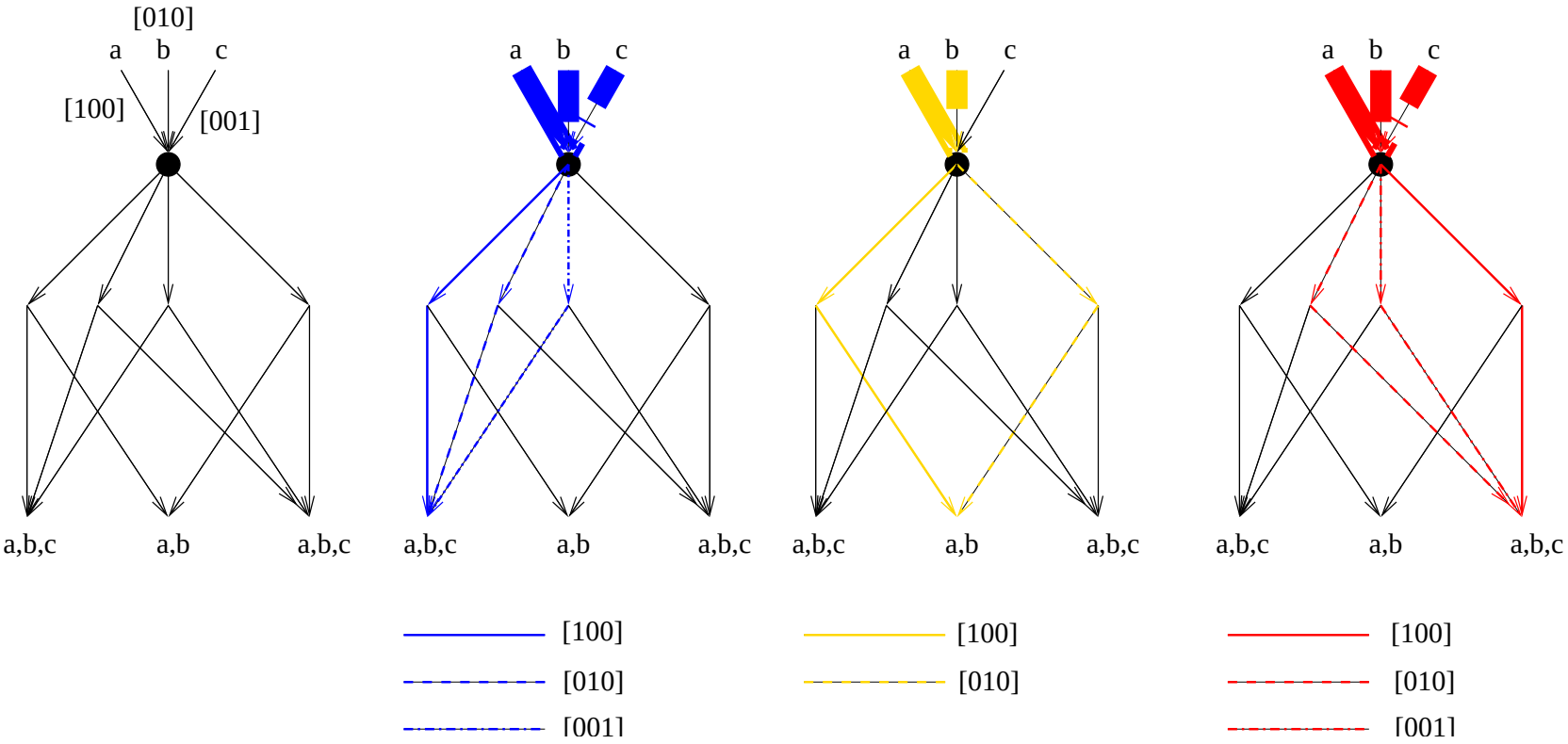
The algorithm of Jaggi, Sanders et al.

Theorem [Jaggi, Sanders, et al] Let $(\mathcal{G}, \mathscr{C})$ be a multicast network coding problem with E edges, R symbols to be transmitted simultaneously and T receivers. There exists a linear network coding solution for $(\mathcal{G}, \mathscr{C})$ over a finite field $\mathbb{F}$ if $|\mathbb{F}| > T$. Moreover this solution can be found in time $O(E \cdot T \cdot R(R+T))$.

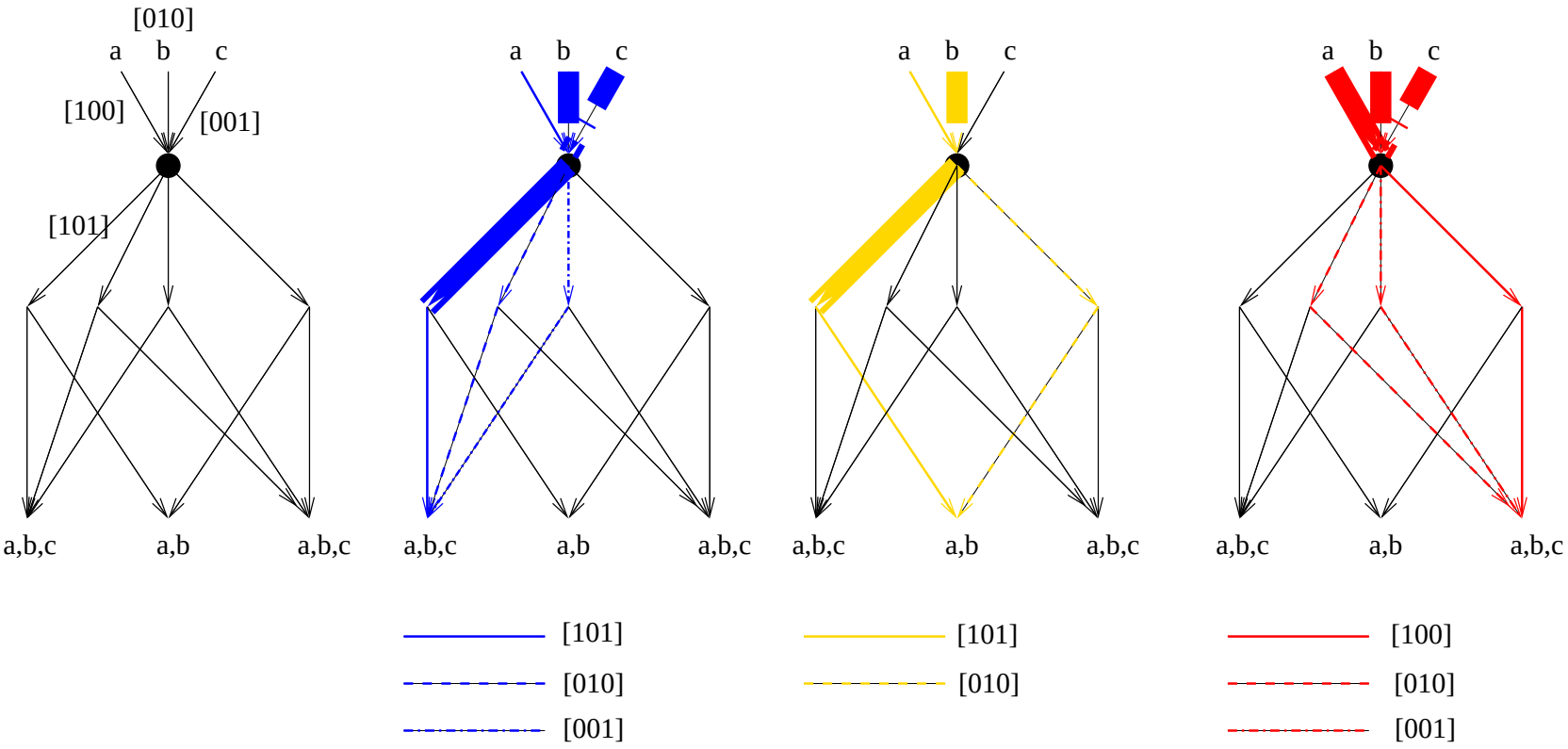
(In practice - still just try the random approach...)

The "link growth" algorithm can be modified such that it is applicable to all the generalizations of the previous session. Example: "Two-Level Multicast"

Two-Level Multicast

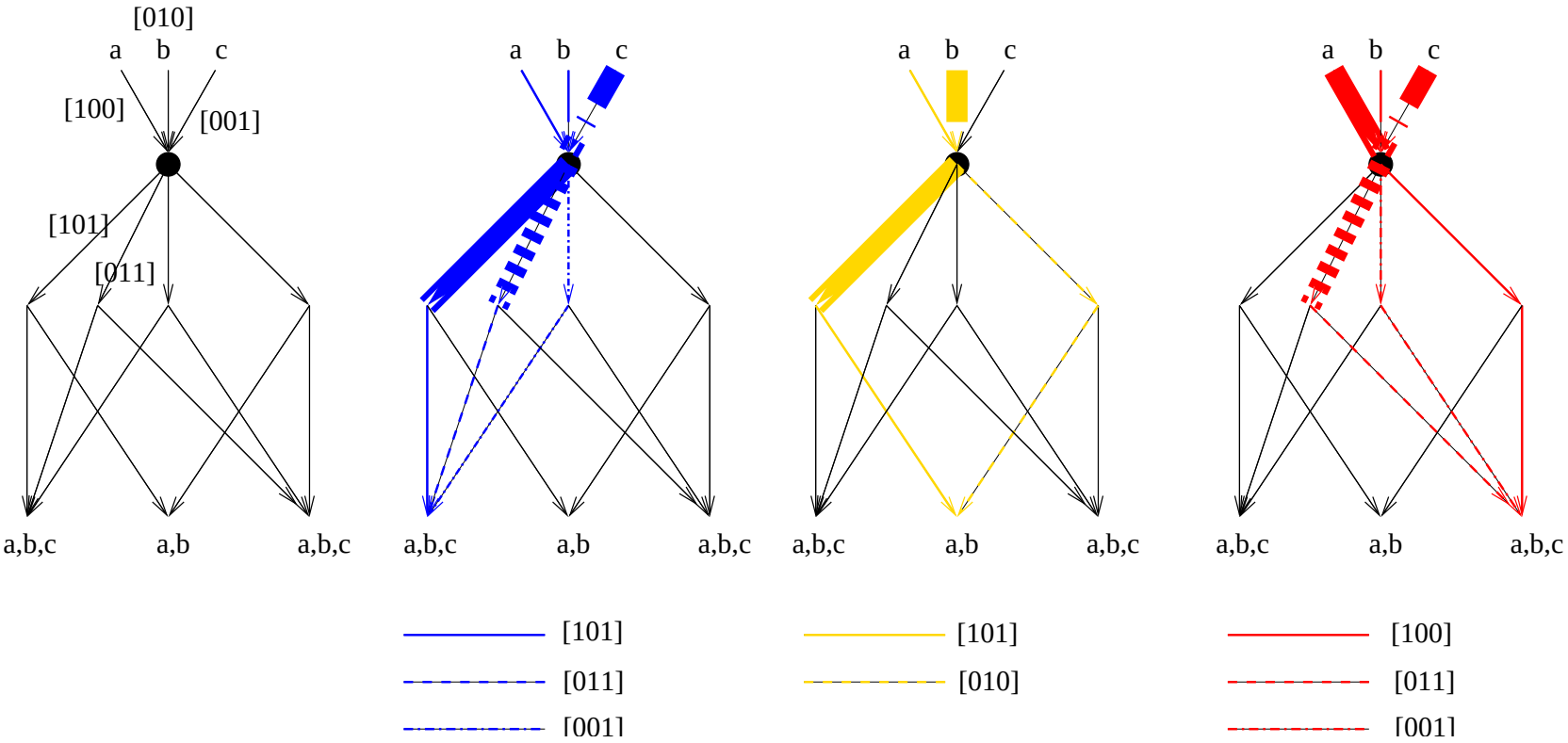


Two-Level Multicast

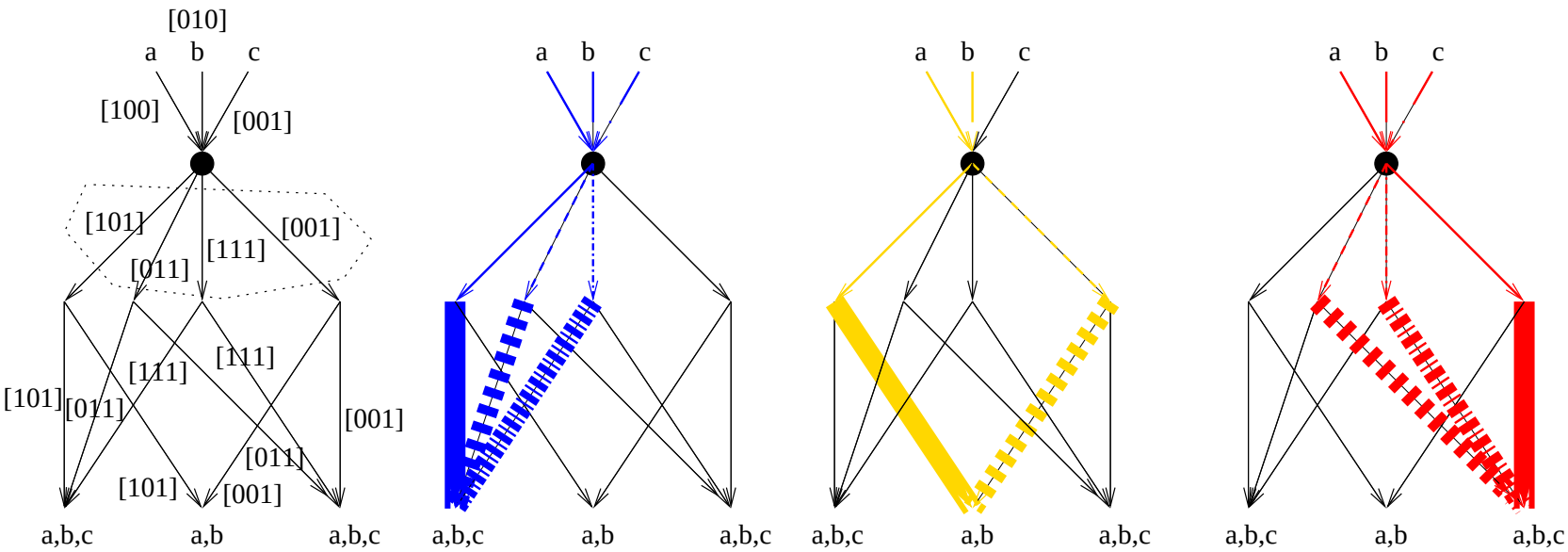


full rank for all frontier sets

Two-Level Multicast



Two-Level Multicast



[101]
[011]
[111]
[001]

[101]
[011]
[111]

[101]
[001]

[001]
[011]
111]

Reencode [a,b,c] as A[abc] ^{T}such that $\begin{matrix}[101]\\[001]\end{matrix}$ A= $\begin{matrix}[100]\\[010]\end{matrix}$

An anlysis of random assignments is done in the next session.

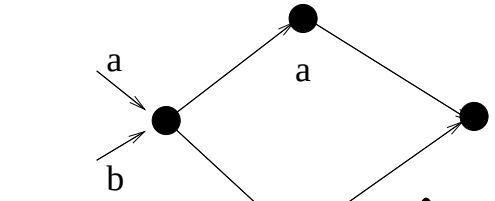
The flow based algorithm is inherently more efficient than a pure random assignment.

How do we pack flows with as much overlap as possible?

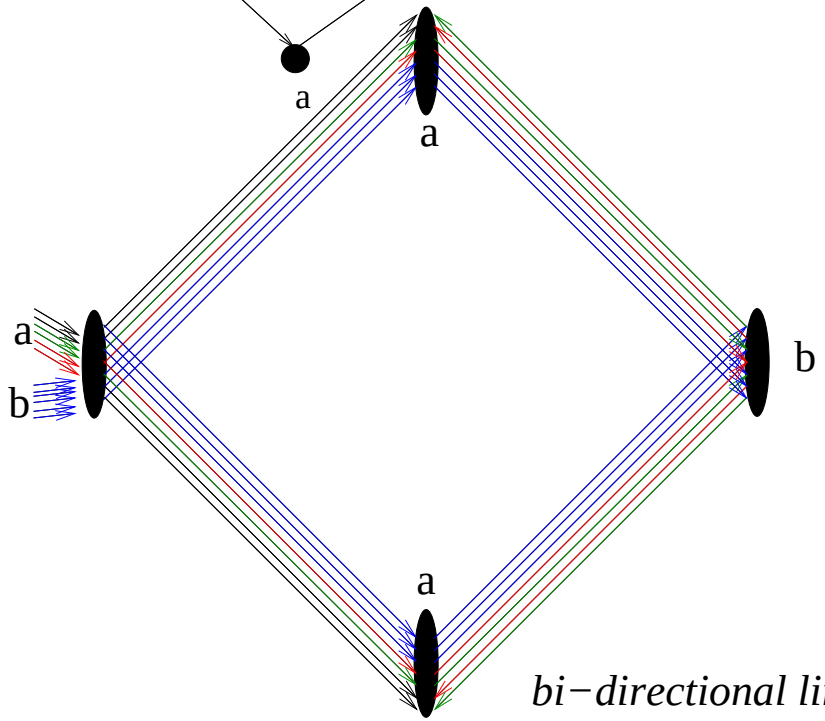
But first: The case of bidirectional links!

(Zongpeng Li, Baochun Li, Dan Jiang, Lap Chi Lau. "On Achieving Optimal End-to-End Throughput in Data Networks: Theoretical and Empirical Studies," Technical Report, University of Toronto, May 2004)

Bidirectional links — A case where network coding does not help



directional links: rate of transmission 0.5 symbols per time unit



bi−directional links:Rate of transnmission is bounded by 6/7

Bidirectional links

Steiner Tree Packing for Multicast Problems:

Find the set of all Steiner trees $\mathscr{T}$, i.e. trees connecting all receivers with a source in a multicast group.

For a link e and $T \in \mathscr{T}$:

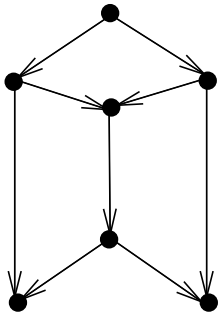
$$I(e,T) = \begin{cases} 1 & e \text{ is part of } T \\ 0 & \text{otherwise} \end{cases}$$

The central problem: Find $\lambda(T) \in \mathbb{R}_+$ maximizing

$$\sum_{T \in \mathscr{T}} \lambda(T) \quad \text{such that} \sum_{T \in \mathscr{T}} \lambda(T) I(e,T) \leq C(e)$$

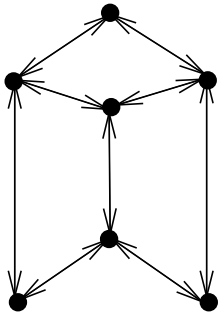
for all links e.

Bidirectional links



Without network coding: One symbol per time unit

With network coding: Two symbols per time unit

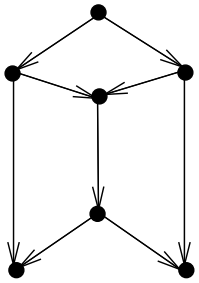


Without network coding: ?

With network coding: ?

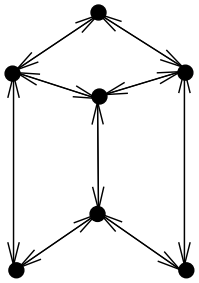
A case where network coding does help (even though it's not much)

Bidirectional links



Without network coding: One symbol per time unit

With network coding: Two symbols per time unit

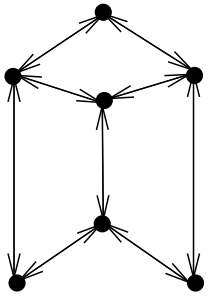


Without network coding: ?

With network coding: Two symbols per time unit (min cut)

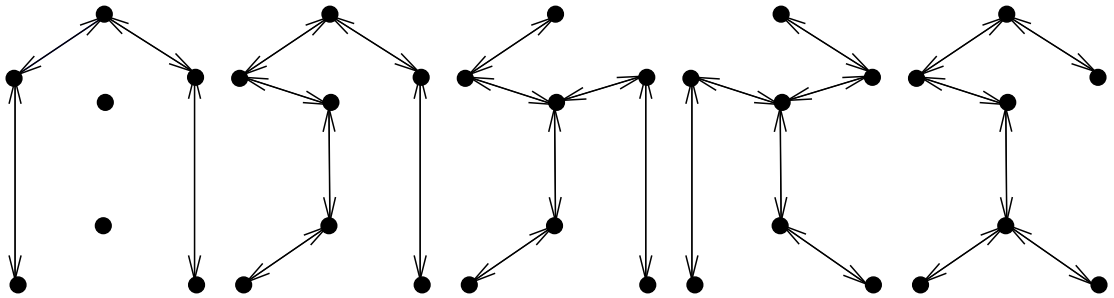
A case where network coding does help (even though it's not much)

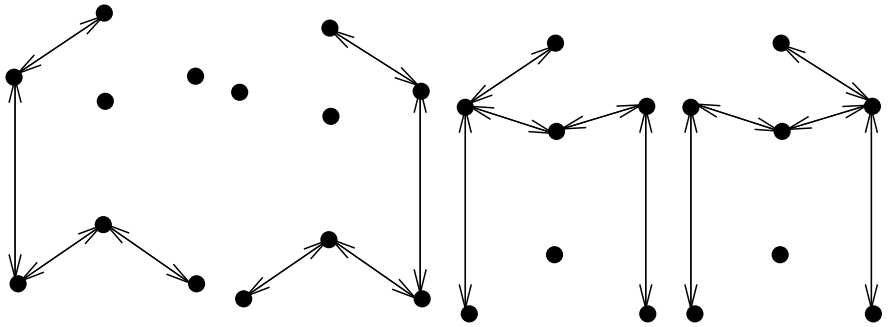
Bidirectional links



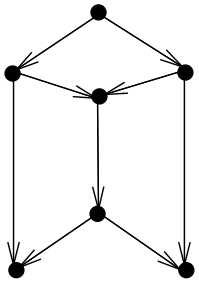
Packing the below trees yields
a rate of 1.5 symbols per time unit

(1.875 optimal [Li,Li,Lau])



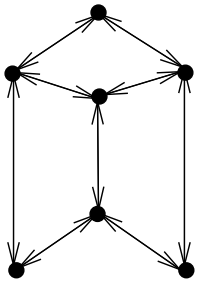


Bidirectional links



Without network coding: One symbol per time unit

With network coding: Two symbols per time unit



Without network coding: 1.875symbols per time unit

With network coding: Two symbols per time unit (min cut)

A case where network coding does help (even though it's not much)

Bidirectional links

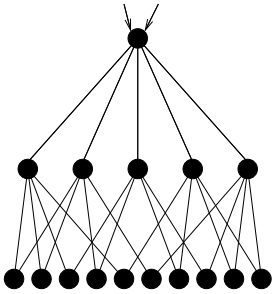
[Li,Li,Lau] The ratio between the multicast rates achievable with or without network coding in bidirectional networks is bounded by a factor of two.

Bidirectional links

[Li,Li,Lau] The ratio between the multicast rates achievable with or without network coding in bidirectional networks is bounded by a factor of two.

(The point of network coding here is really complexity!)

Bidirectional links - An Example



With network coding we achieve a capacity of 2 symbols per time unit

Without network coding we achieve a throughput of 1.786 symbols per unit time

This comes at a cost of optimizing over 119104 Steiner trees [Li,Li,Lau]

Bidirectional links

He crucial step in a network coding solution for the multicast problem in bidirectional links is to find the best (bidirectional) flows corresponding to each receiver. To this end we formalute a linear program:

Each link e carries two flows (direction $+$ and $-$) $f_+^{(\ell)}(e)$ and $f_-^{(\ell)}(e)$ due to receiver ℓ.

Maximize: f

Constraints for all ℓ

$$f_+^{(\ell)}(e) + f_-^{(\ell)}(e) \leq c(e)$$

$$\sum_{f^{(\ell)} \text{ flowing into reveiver } \ell} f^{(\ell)} = f$$

$$\sum_{f^{(\ell)} \text{ flowing out of the source}} f^{(\ell)} = f$$

$$\sum_{f^{(\ell)} \text{ flowing into node } i} f^{(\ell)} = \sum_{f^{(\ell)} \text{ flowing out of node } i} f^{(\ell)}$$

Summary:

- For directed networks the "coding gain" is unbounded
- We "really" need codes
- The necessary multicast fieldsize is bounded as $\sqrt{T} \leq |\mathbb{F}| \leq T$
- Two basic methods to find solutions: algebraic and recursively assigning edges
- A natural method: "random assignment" (more about this shortly)

- For bidirectional link the coding gain is bounded by 2
- The main advantage of network coding in complexity.
- More about linear programs shortly!

Decentralized code construction and network coding for multicast with a cost criterion

Overview

Randomized construction and its error behavior

Performance of distributed randomized construction - case studies

Traditional methods based on flows - a review

Trees for multicasting - a review

Network coding with a cost criterion - flow-based methods for multicasting through linear programming

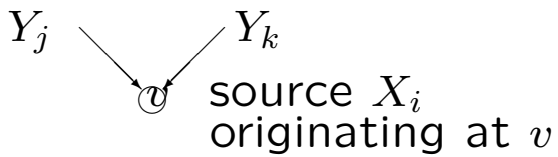
Distributed operation - one approach

A special case - wireless networks

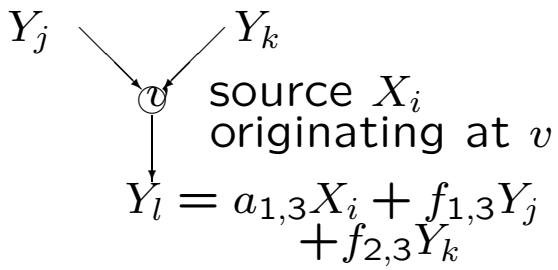
Sample ISPs

Linear network coding

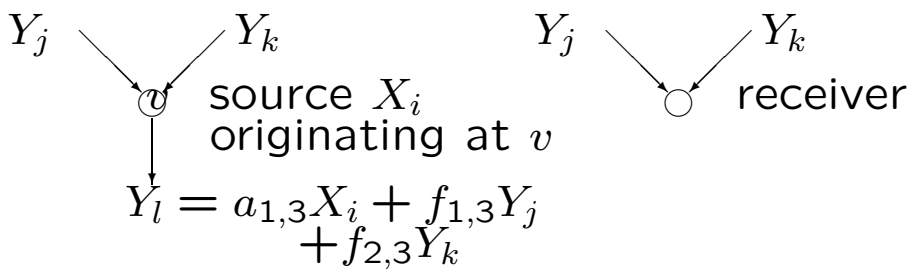
Linear network coding for multicast



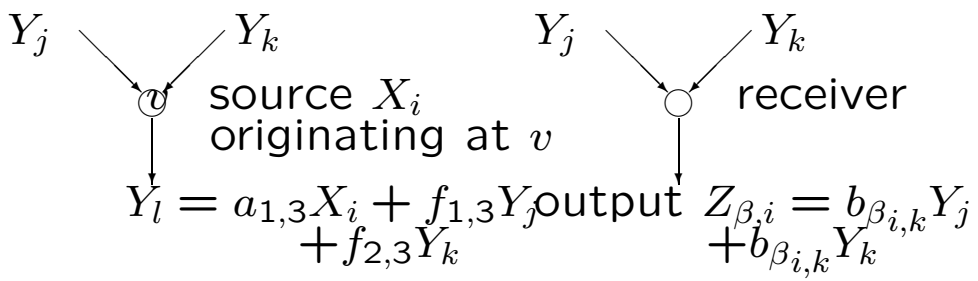
Linear network coding for multicast



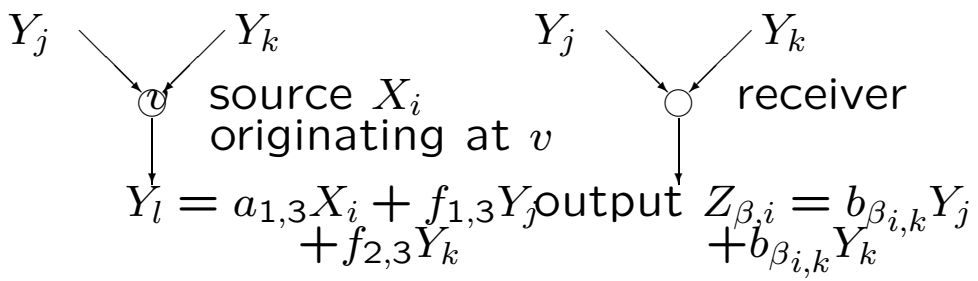
Linear network coding for multicast



Linear network coding for multicast



Linear network coding for multicast

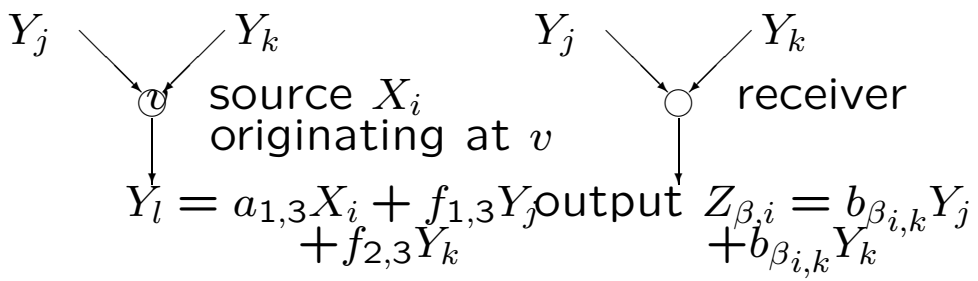


Coefficients $\{a_{i,j}, f_{l,j}, b_{\ i,l}\}$ give network-constrained transfer matrices $(A, F, \{B\ \})$, a network code

Matrix $M = A(I \quad F)^{-1}B^T$ gives transfer function from sources to outputs [KM01]:

$$[X_1\ X_2 \ldots X_r]\ M\ = [Z_{\ ,1}\ Z_{\ ,2} \ldots Z_{\ ,r}]$$

Linear network coding for multicast



Coefficients $\{a_{i,j}, f_{l,j}, b_{\ i,l}\}$ give network-constrained transfer matrices $(A, F, \{B\ \})$, a network code

Matrix $M = A(I \quad \mathbf{D}F)^{-1}B^T$ gives transfer function from sources to outputs [KM01]:

$$[X_1\ X_2 \ldots X_r]\ M\ = [Z_{\ ,1}\ Z_{\ ,2} \ldots Z_{\ ,r}]$$

Feasibility and code construction

Determining feasibility

- min-cut max-flow bound satisfied for each receiver [ACLY00]
- transfer matrix $A(I \quad F)^{-1}B^T$ for each receiver β is non-singular [KM01]

Constructing linear solutions

- Centralized

- Direct algebraic solution using transfer matrix of [KM01]
- Algorithms using subgraph consisting of flow solutions to individual receivers [SET03, JCJ03]

Decentralized

- A distributed randomized network coding approach [HKMKE03]

Randomized network coding

Interior network nodes independently choose random linear mappings from inputs to outputs

Coefficients of aggregate effect communicated to receivers

Randomized network coding

Interior network nodes independently choose random linear mappings from inputs to outputs

Coefficients of aggregate effect communicated to receivers

Receiver nodes can decode if they receive as many independent linear combinations as the number of source processes

Success probability

[HKMKE03, HMSEK03] For a feasible d-receiver multicast connection problem on a network with

independent or linearly correlated sources

a network code in which code coefficients $a_{i,j}$, $f_{l,j}$ for η links are chosen independently and uniformly over $\mathbb{F}_q$

the success probability is at least $(1 \quad d/q)^\eta$ for $q > d$. Error bound is of the order of the inverse of the field size, so error probability decreases exponentially with codeword length

Proof outline

Recall transfer matrix $M = A(I \quad F)^{-1}B^T$ for each receiver β must be non-singular

We show an equivalent condition connected with bipartite matching: the Edmonds matrices $\begin{bmatrix} A & 0 \\ I & F & B^T \end{bmatrix}$ (in the acyclic delay-free case) or $\begin{bmatrix} A & 0 \\ I & DF & B^T \end{bmatrix}$ (in the case with delays) are non-singular

This shows that if η links have random coefficients, the determinant polynomial

– has maximum degree η in the random variables $\{a_{x,j}, f_{i,j}\}$

– is linear in each of these variables

Proof outline (cont'd)

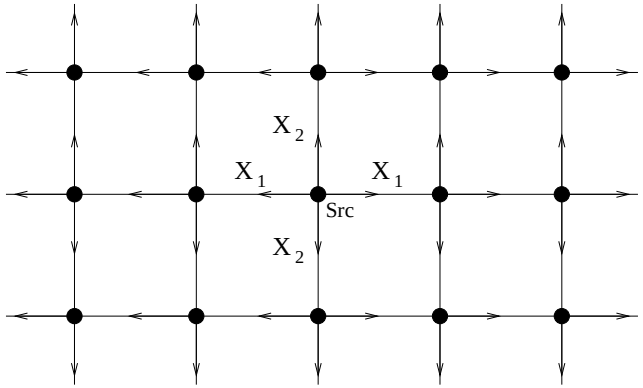
We want the product of the d receivers' determinant polynomials to be nonzero

We can show inductively, using the Schwartz-Zippel Theorem, that for any polynomial $P \in \mathbb{F}[\xi_1, \xi_2, \ldots]$ of degree $d\eta$, in which each ξ_i has exponent at most d, if $\xi_1, \xi_2, \ldots$ are chosen independently and uniformly at random from $\mathbb{F}_q$ $\mathbb{F}$, then $P = 0$ with probability at most 1 $(1$ $d/q)^\eta$ for $d < q$

Particular form of the determinant polynomials gives rise to a tighter bound than the Schwartz-Zippel bound for general polynomials of the same total degree

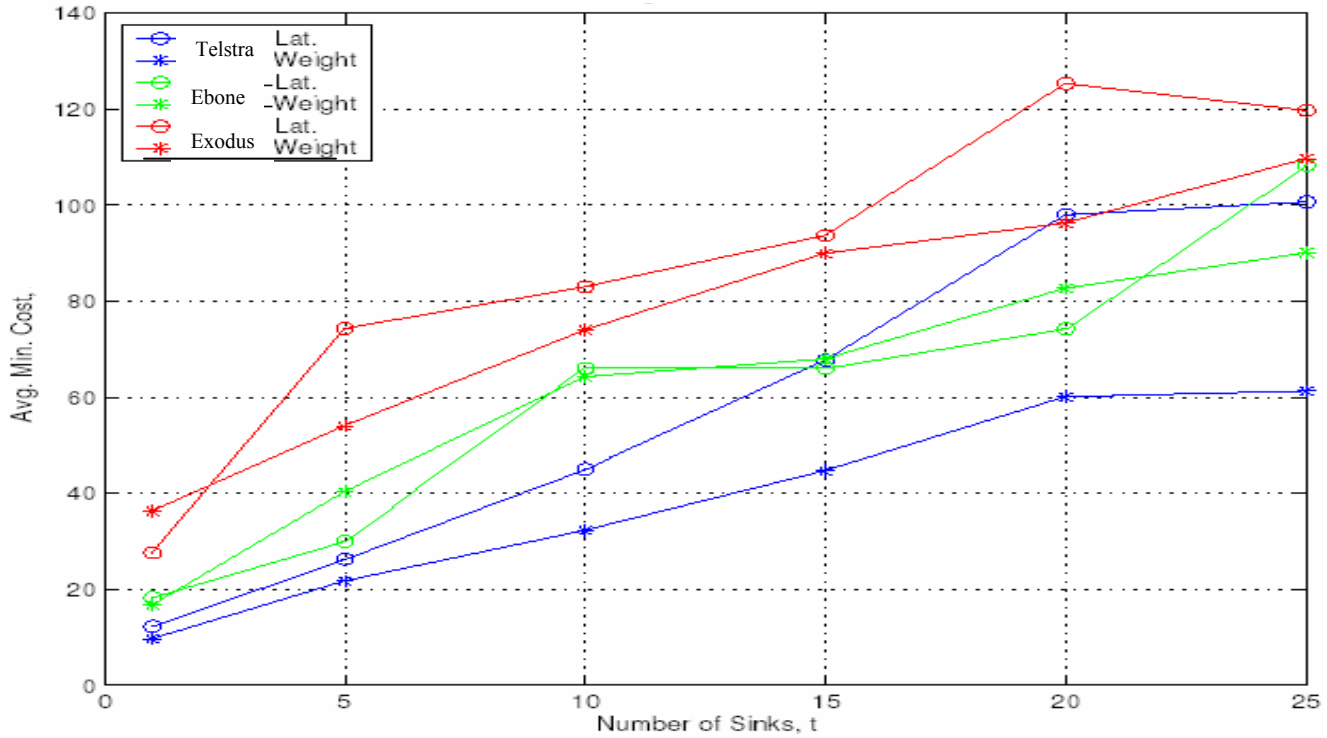
Utility of distributed network coding

Decentralized scenarios



Receiver position		(2,4)	(4,4)	(8,10)	(10,10)
Randomized flooding upper bound		0.563	0.672	0.667	0.667
Randomized Coding	$\mathbb{F}_{2^6}$ lower bound	0.882	0.827	0.604	0.567
	$\mathbb{F}_{2^8}$ lower bound	0.969	0.954	0.882	0.868

Running the LP on sample ISPs (Rocketfuel)



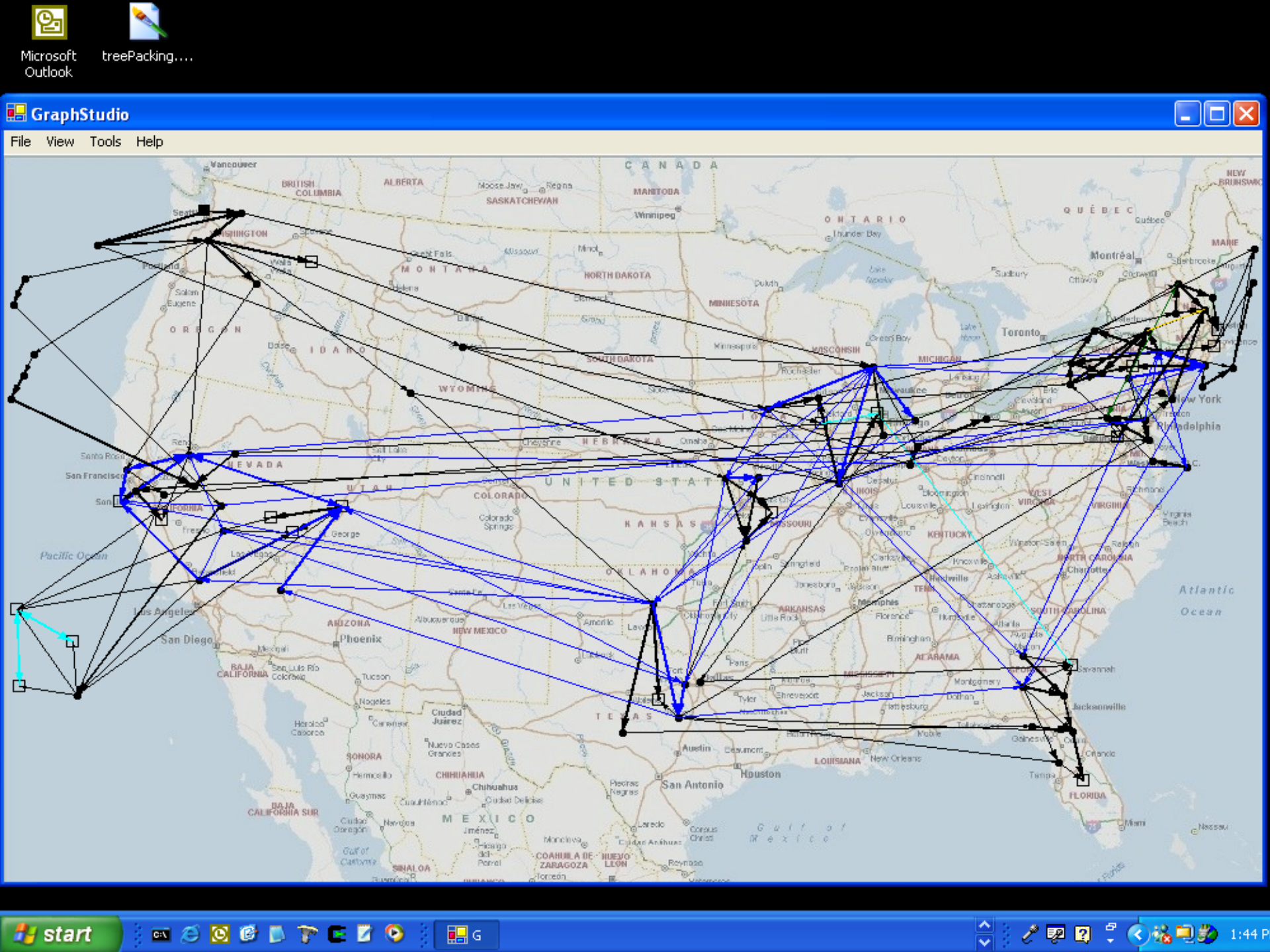
Telstra, V= 108, E = 306

Ebone, V=88, E=323

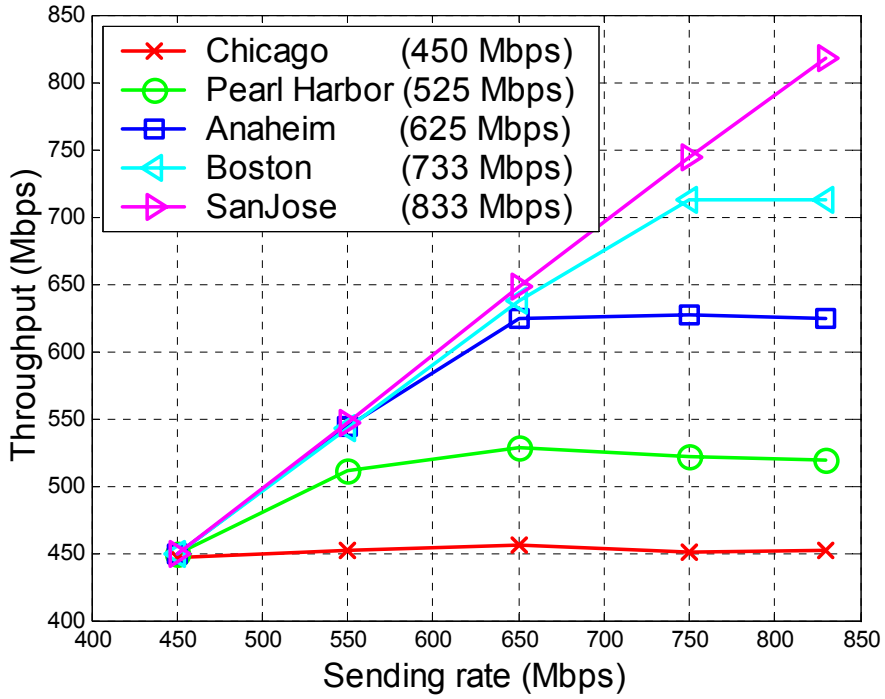
Exodus, V=79, E=294

Another case study

- Results of Chou, Wu and Jain 2003
- Implemented event-driven simulator in C++
- Six ISP graphs from Rocketfuel project (UW)
  - SprintLink: 89 nodes, 972 bidirectional edges
  - Edge capacities: scaled to 1 Gbps / "cost"
  - Edge latencies: speed of light x distance
- Sender: Seattle; Receivers: 20 arbitrary (5 shown)
  - Broadcast capacity: 450 Mbps; Max 833 Kbps
  - Union of maxflows: 89 nodes, 207 edges
- Sent 20000 packets in each experiment::
  - field size: 2^{16}; generation size (group of packets):100; interleaving length: 100


Microsoft Outlook
treePacking....
GraphStudio
File View Tools Help
CANADA
UNITED STATES
MEXICO
Vancouver
Seattle
Portland
San Francisco
Los Angeles
San Diego
Las Vegas
Phoenix
Tucson
Albuquerque
Denver
Colorado Springs
Salt Lake City
Boise
Minneapolis
Winnipeg
Toronto
Montréal
Chicago
Detroit
Cleveland
Pittsburgh
New York
Philadelphia
Boston
Washington D.C.
Atlanta
Charlotte
Jacksonville
Miami
Tampa
New Orleans
Houston
San Antonio
Dallas
Austin
Kansas City
St. Louis
Memphis
Nashville
Louisville
Cincinnati
Indianapolis
Omaha
Des Moines
Milwaukee
Buffalo
Pacific Ocean
Atlantic Ocean
Gulf of Mexico
start
1:44 P

Throughput


Chicago (450 Mbps)
Pearl Harbor (525 Mbps)
Anaheim (625 Mbps)
Boston (733 Mbps)
SanJose (833 Mbps)
Throughput (Mbps)
Sending rate (Mbps)
400
450
500
550
600
650
700
750
800
850

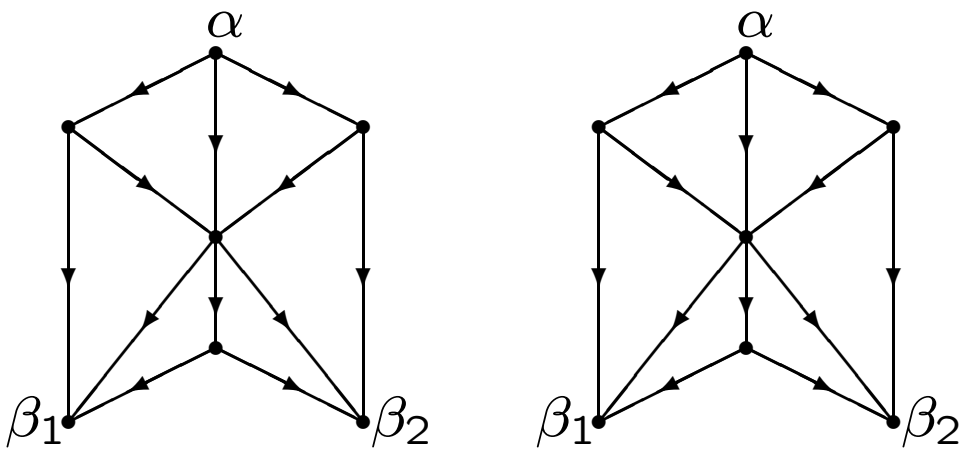
Tighter bounds on coding success probability

Previous bound in terms of number of receivers and coding links is very general

Can obtain tighter bounds based on more specific network characteristics

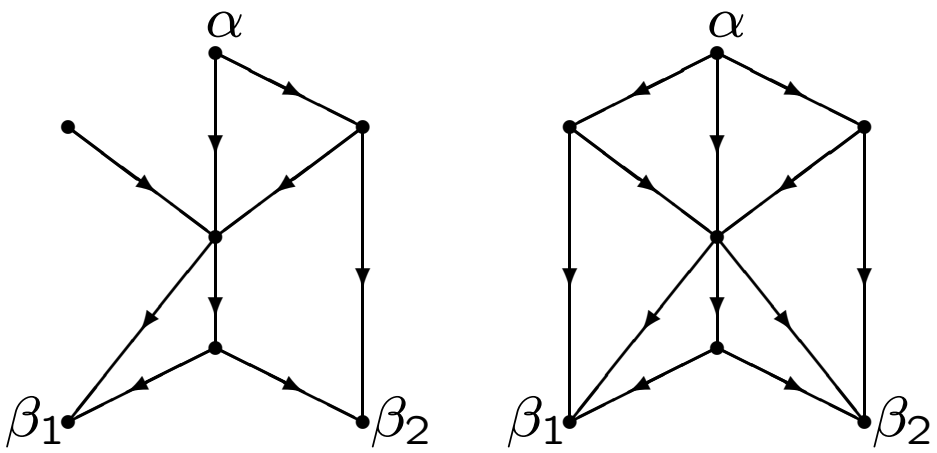
Tighter bounds on coding success probability

[HMSEK03]For a d-receiver multicast problem on an acyclic network, coding success probability with field size q is lower bounded by the probability that the connections remain feasible after deleting each link of the original graph with probability $\frac{d}{q}$



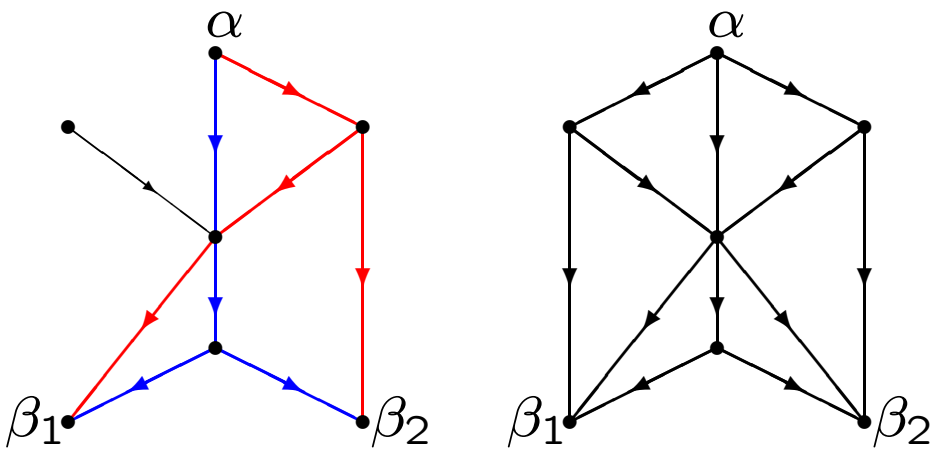
Tighter bounds on coding success probability

[HMSEK03]For a d-receiver multicast problem on an acyclic network, coding success probability with field size q is lower bounded by the probability that the connections remain feasible after deleting each link of the original graph with probability $\frac{d}{q}$



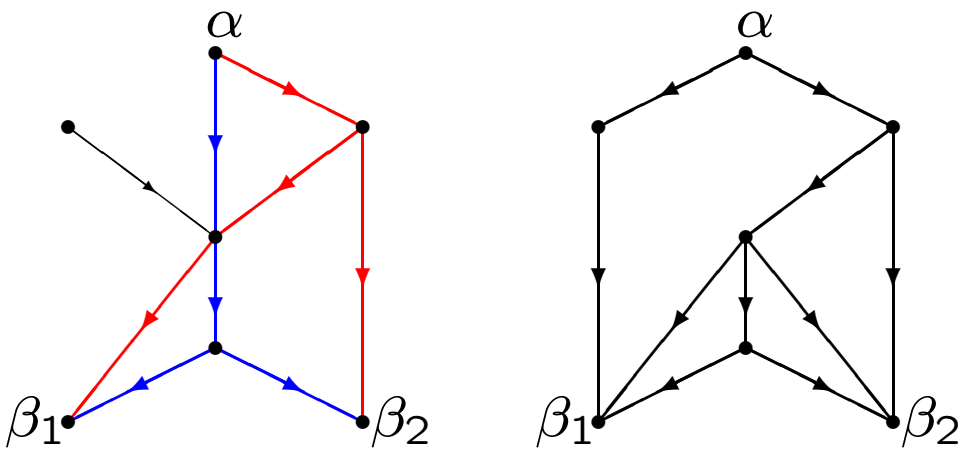
Tighter bounds on coding success probability

[HMSEK03]For a d-receiver multicast problem on an acyclic network, coding success probability with field size q is lower bounded by the probability that the connections remain feasible after deleting each link of the original graph with probability $\frac{d}{q}$



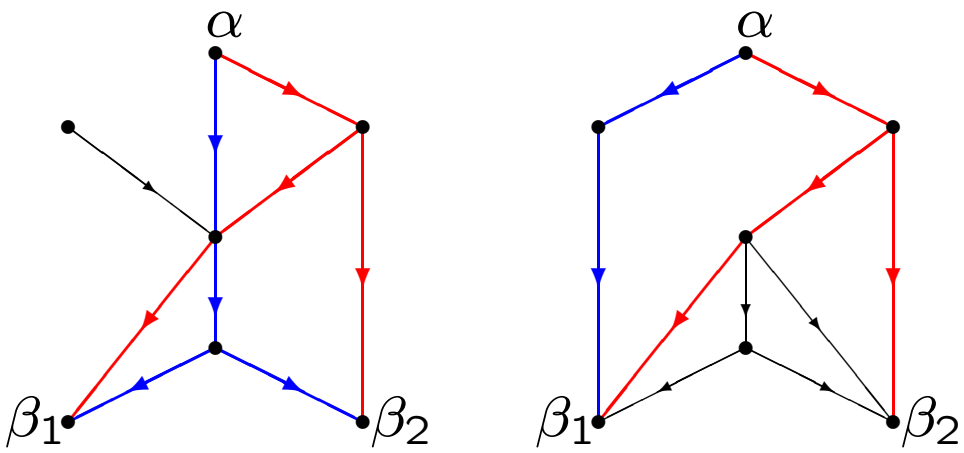
Tighter bounds on coding success probability

[HMSEK03]For a d-receiver multicast problem on an acyclic network, coding success probability with field size q is lower bounded by the probability that the connections remain feasible after deleting each link of the original graph with probability $\frac{d}{q}$



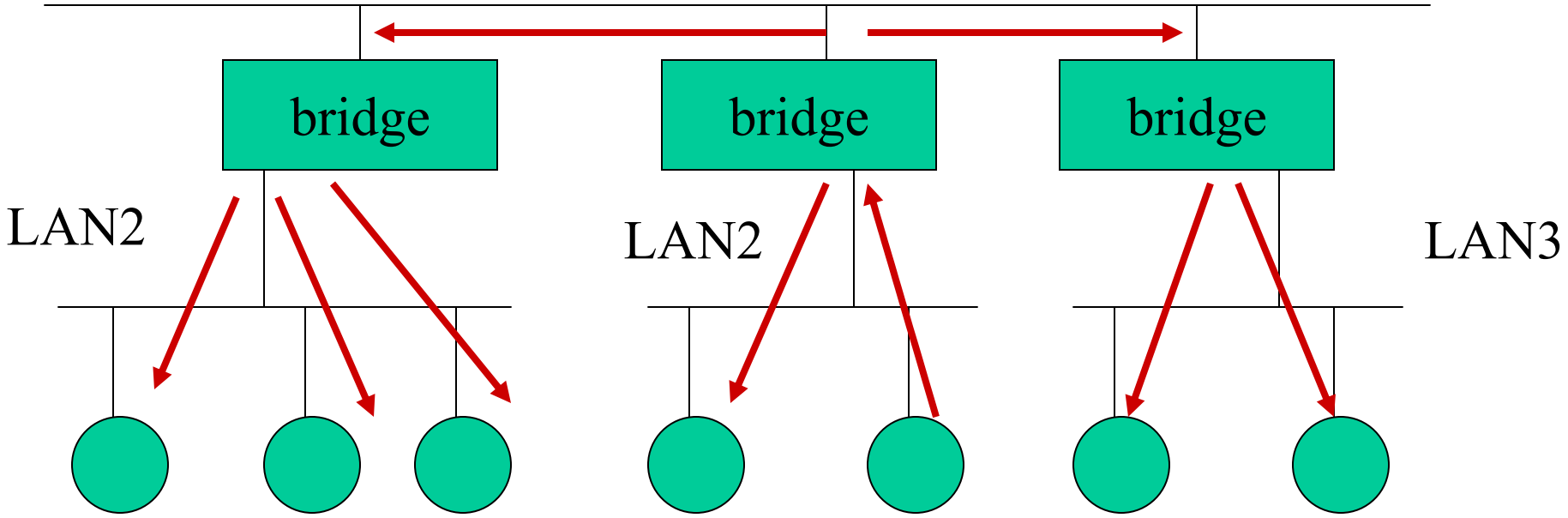
Tighter bounds on coding success probability

[HMSEK03]For a d-receiver multicast problem on an acyclic network, coding success probability with field size q is lower bounded by the probability that the connections remain feasible after deleting each link of the original graph with probability $\frac{d}{q}$



Multicasting

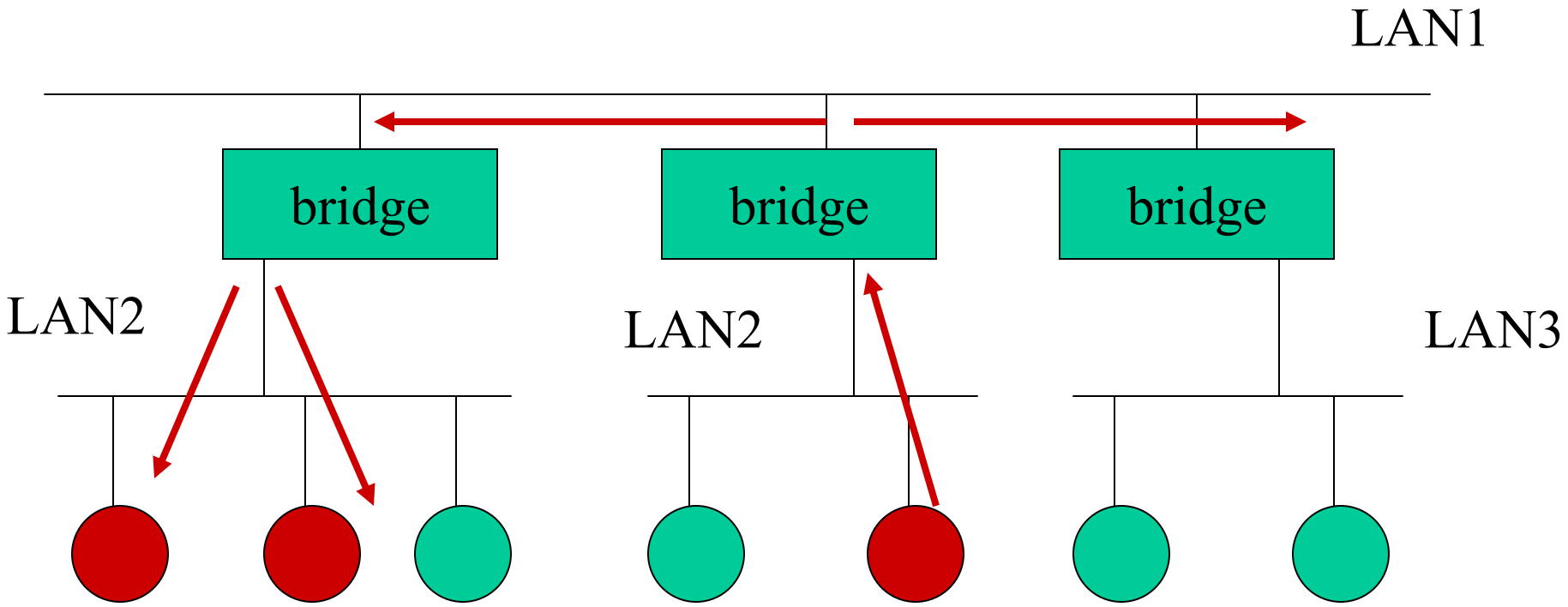
- Example: IEEE 802.1P and 802.1Q
- 802.1P: bridge specification for traffic class expediting, registration protocol, multicast group filters



GARP: generic attribute registration protocol, uses broadcasting

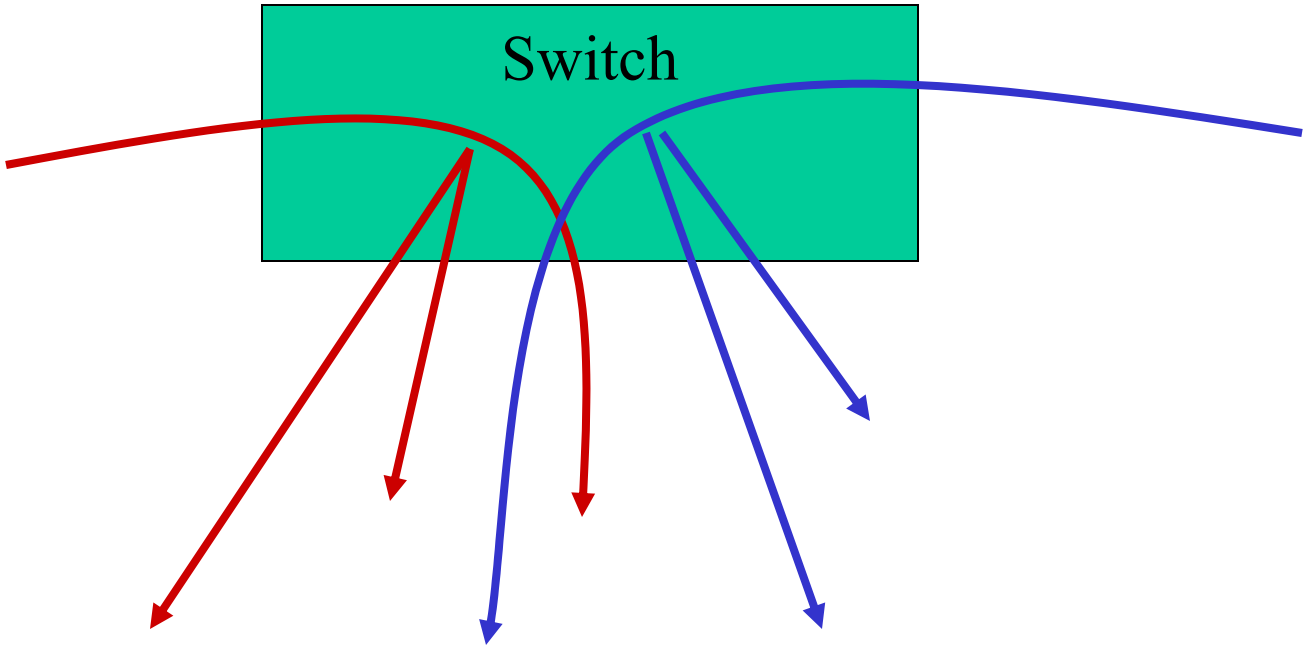
Multicasting

- GMRP: GARP for multicast
- allows users to join a multicast group



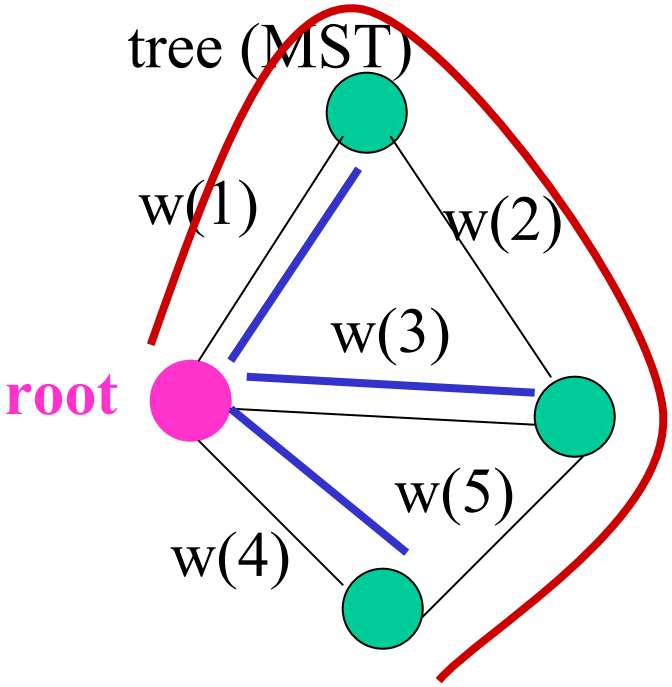
Multicasting

- 802.1Q: virtual local area networks (VLAN)s
- Every VLAN is a broadcast domain
- Manually (in general) setting up of VLANs through a switch



Minimum spanning tree

- Represent a network as graph: links are edges and nodes are vertices
- Weights w(i) may represent cost of sending down the link, cost of rental, delay/length
- Want to minimize sum of w(i)s to yield a minimum spanning tree (MST)



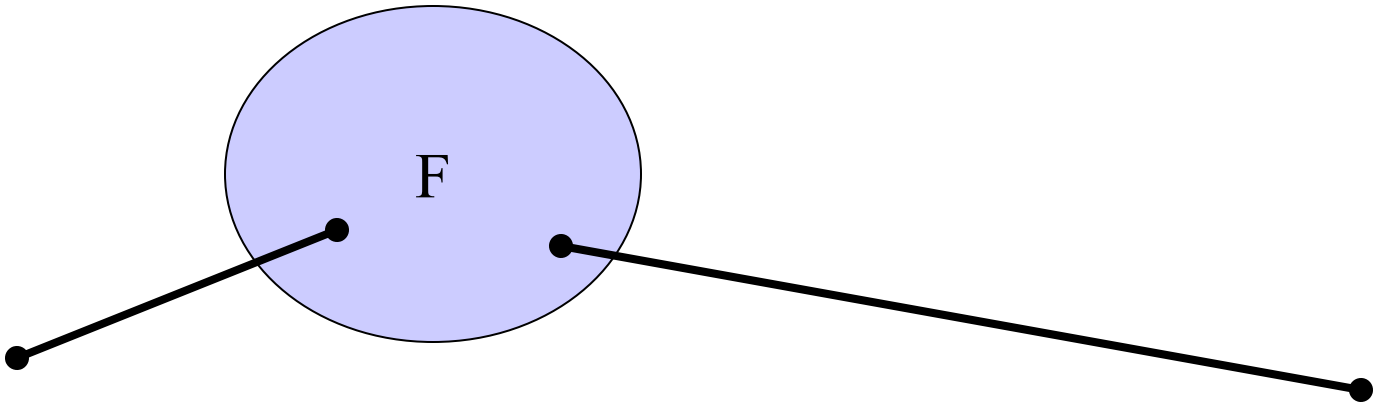
Two valid trees: **tree 1** and **tree 2**

Fragment: a subtree

We can use fragments to build trees progressively

Augmenting a fragment

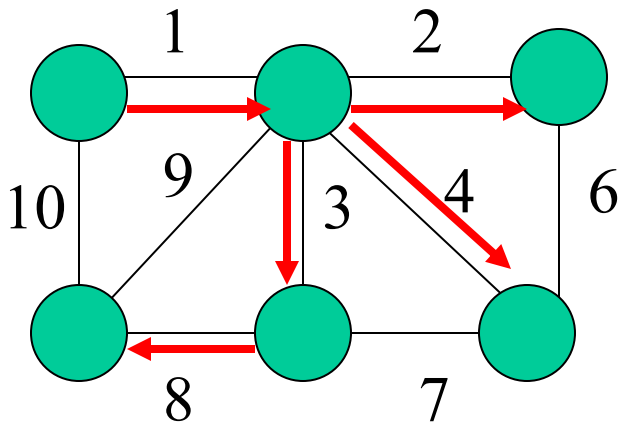
- Let us take a fragment F of an MST
- Adding a minimum weight link to F yields another MST fragment
- To see why: add that minimum weight link to the MST, creating a cycle, and then replace another link with it
- If all the weights are different, there exists a single MST



Minimum weight link

Algorithms relying on augmenting fragments

- Prim-Dijkstra: start from the root node and gradually augment it until all nodes are in the MST
- Kruskal: every node is a fragment and fragments are successively joined
- Example:



- The problem of performing multicast rather than broadcast is difficult - Steiner tree problem, NP-complete

Steiner Tree Problem

- The Steiner tree of some subset of the vertices of a graph *G* is a minimum-weight connected subgraph of *G* that includes all the vertices. It is always a tree.
- The determination of a Steiner tree is NP-complete (both NP (verifiable in nondeterministic polynomial time) and NP-hard (any other NP-problem can be translated into this problem)) and hard even to approximate. There is 1.55-approximate algorithm (Robins and Zelikovski 2000), but approximation within 95/94 is known to be NP-hard (Chlebik and Chlebikova 2002).

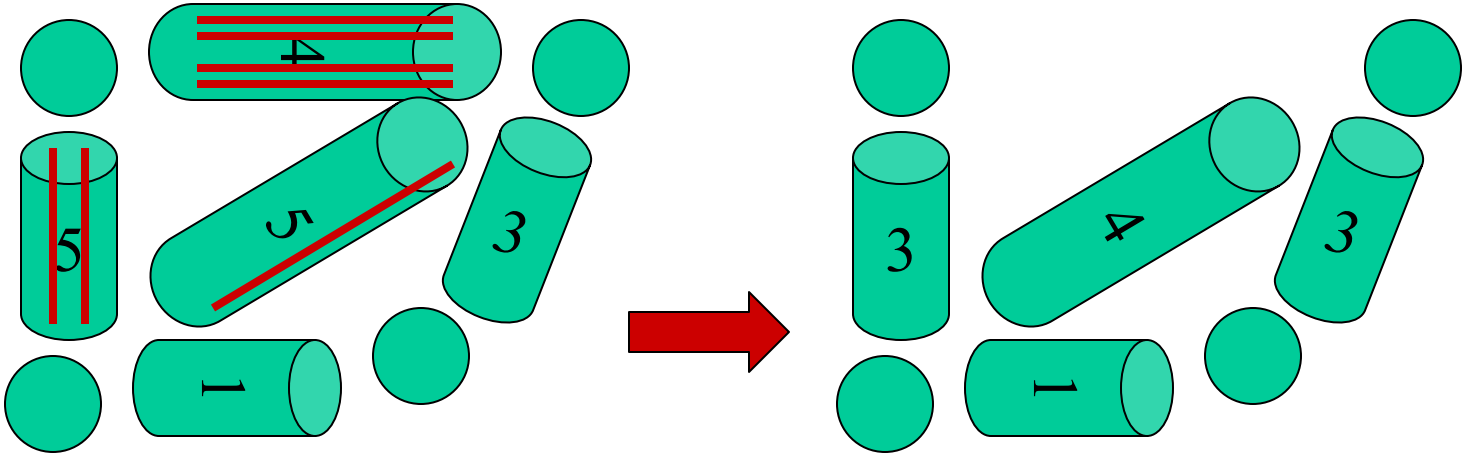
- Chlebik, M. and J.Chlebikova, J. "Approximation Hardness of the Steiner Tree Problem on Graphs." *Proc. 8th Scandinavian Workshop on Algorithm Theory (SWAT)*. Springer-Verlag, pp. 170-179, 2002.
- Robins, G. and Zelikovski, A. "Improved Steiner Tree Approximation in Graphs." In *Proc. 11th ACM-SIAM Symposium on Discrete Algorithms*. pp. 770-779, 2000
- from Eric W. Weisstein. "Steiner Tree." From *MathWorld*--A Wolfram Web Resource. http://mathworld.wolfram.com/SteinerTree

Steiner tree related problems

- The Steiner packing problem is to find the maximum number of edge-disjoint subgraphs of a given graph G that connect a given set of required points, S. Recently an algorithm with an asymptotic approximation factor of |S|/4 has been found (K. Jain, M. Mahdian, and M.R. Salavatipour, "Packing Steiner trees," 14th ACM-SIAM Symposium on Discrete Algorithms (SODA), 2003)
- Insertion of buffers in Steiner trees (C. J. Alpert, M. Hrkic, J. Hu, A. B. Kahng, J. Lillis, B. Liu, S. T. Quay, S. S. Sapatnekar, A. J. Sullivan, P. Villarrubia, "Buffered Steiner trees for difficult instances", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume: 21 Issue: 1 , January 2002, pp. 3-14; code available at http://ece.tamu.edu/~cnsze/GSRC/ctree.html)
- Dynamic Steiner tree problem and updates (rearrangeability issues). Recently polynomial time algorithms within a factor of 2 from the proven lower bound have been found (Piotr Berman and Chris Coulston, " On-line algorithms for Steiner tree problems", in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 344-353, El Paso, Texas, 4-6 May 1997)

Routing based on flows - a review

- Tree-based methods concentrate on finding a path, taking into consideration cost but not actual flow (tree packing does so indirectly)
- What happens when we add capacity considerations?
- If we have circuits or virtual circuits, then we can create a topology that takes into account the presence of other users



- If we have a probabilistic description of traffic progression, then we could use dynamic programming

Routing based on flows

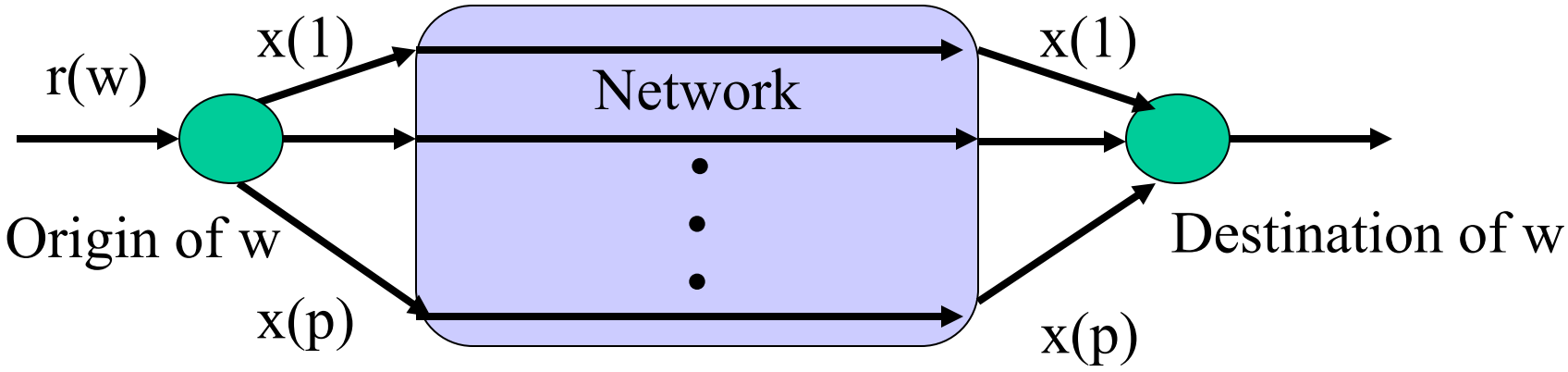
- When we have packets or fine granularity virtual circuits (VCs), we can assume that we have roughly fluid flows
- We will try to optimize a cost function related to the flow F(i,j) (arrival rate in terms of of what we may be considering) on the link (i,j)
- A reasonable metric is $\sum_{(i,j)} D(F(i,j))$

  where D is some monotonically increasing function that grows very sharply when F(i,j) approaches the link capacity
- These type of models are called flow models - they look only at mean flow, they do not consider higher moments
- Example (based on Kleinrock approximation)

$$D(F(i,j)) = \frac{F(i,j)}{C(i,j) - F(i,j)} + d(i,j)\,F(i,j)$$

Optimal flow routing problem

- For every OD pair w=(i,j), we are given the arrival rate r(w)
- We denote:
  - W: the set of all OD pairs w
  - P(w): a set of paths available for routing for OD pair w (possibly all paths)
  - x(p): the flow of path p (in units per second)



Problem statement

- We want to choose the flows so that we minimize

$$\sum_{(i,j)} D(F(i,j))$$

subject to the constraints

$$F(i,j) = \sum_{\text{all paths p containing } (i,j)} x(p)$$

$$\sum_{p \text{ in } P(w)} x(p) = r(w) \text{ for al OD pairs w in W}$$

$$x(p) \geq 0$$

we assume that D is convex and that its first and second derivatives exist

Flow problem

- We can perform a substitution in the following manner: consider the cost

$$D(x) = \sum_{(i,j)} D\left(\sum_{\text{all paths p containing } (i,j)} x(p) \right)$$

consider the partial derivatives

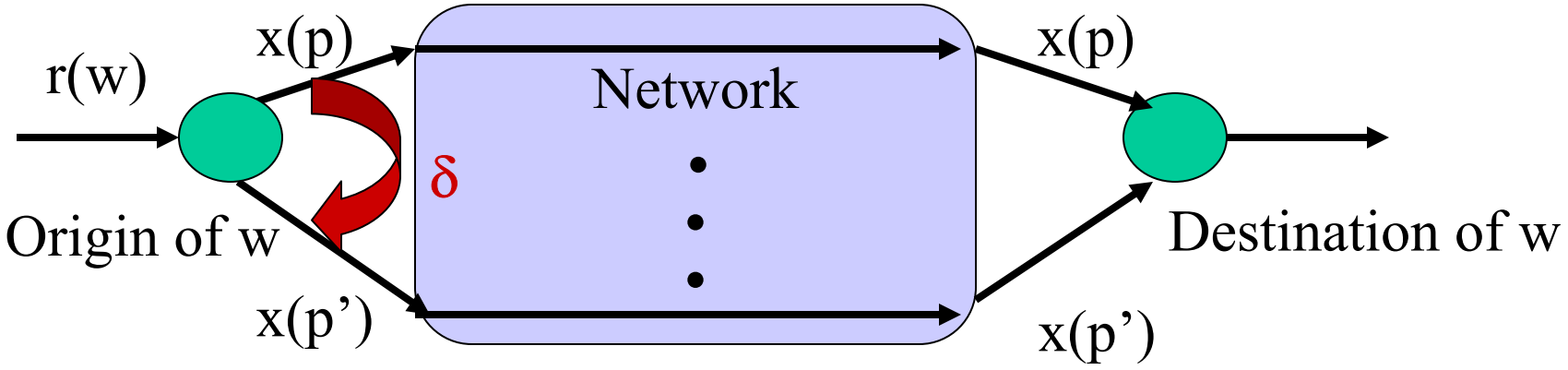
$$\frac{\partial D(x)}{\partial x(p)} = \sum_{\text{all links } (i,j) \text{ on path p}} D'\left(\sum_{\text{all paths p containing } (i,j)} x(p) \right)$$

partial derivative of $x(p)$ is the length of p

when link lengths are the first link cost derivatives D'

Characterization of optimal flows

- A set of path flows x is optimal if the cost cannot be improved by making a feasible change of flow
- Consider shifting an increment δ from path p to path p' of the OD pair w



- To a first order approximation, the change in cost is

$$-\frac{\partial D(x)}{\partial x(p)}\delta + \frac{\partial D(x)}{\partial x(p')}\delta$$

Characterization of optimal flows

- Cost change due to shifting traffic from p to p' must be detrimental in cost in the solution was optimal
- So

$$\frac{\partial D(x)}{\partial x(p)} \leq \frac{\partial D(x)}{\partial x(p')}$$

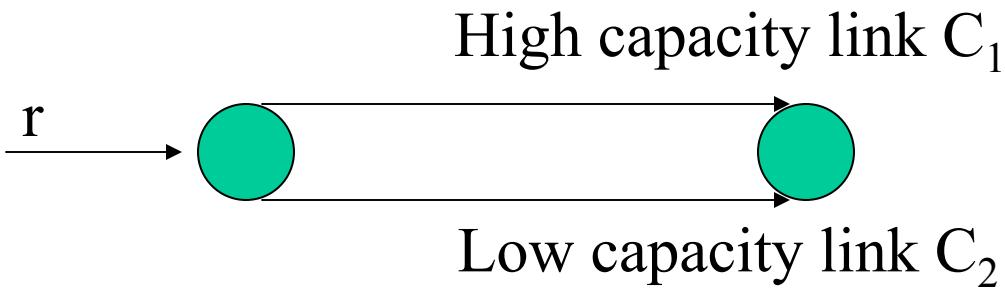
- This condition is the same as rewriting for all OD pairs that for the optimal flow x*,

$$x^*(p) > 0 \text{ only if } \frac{\partial D(x^*)}{\partial x(p)} \leq \frac{\partial D(x^*)}{\partial x(p')} \text{ for all } p' \text{ in } P(w)$$

- So optimal path flows are positive only on paths that are shortest with respect to first derivative lengths

Solutions

- The link lengths depend on the link flows
- The first derivative lengths depend on the unknown optimal path flows, so the problem is harder than a shortest path problem
- Sometimes we can solve the problem using a closed form solution, but generally more involved methods may be necessary



$$\text{cost } D(x(i)) = \frac{x(i)}{C_i - x(i)}$$

$C_1 \geq C_2$, so two cases are possible

Solution example

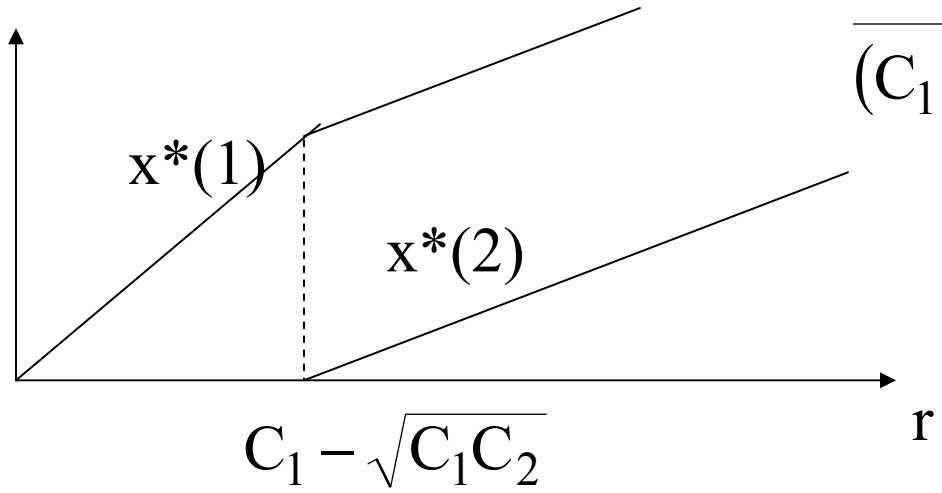
- Case 1: all the flow goes along 1

$$\frac{\partial D(x^*)}{\partial x(p)} \le \frac{\partial D(x^*)}{\partial x(p')} \text{ for all } p' \text{ in } P(w)$$

$$\text{gives } \frac{C_1}{(C_1 - r)^2} \le \frac{1}{C_2} \Rightarrow r \le C_1 - \sqrt{C_1 C_2}$$

- Case2 : Flow goes along both paths

$$\frac{C_1}{(C_1 - x(1))^2} \le \frac{C_2}{(C_2 - x(2))^2}$$



Solution methods

- In general finding solutions may be difficult and different types of methods may need to be applied
- Reduce cost while maintaining feasibility
- A common theme in the methods for improving solutions is to use minimum first derivative lengths and mix, in some way, the current solution with the solution using only minimum first derivative lengths (MFDLs)
- General problem:

$$\text{Feasibility}: \sum_{p \text{ in } P(w)} \Delta x(p) = 0 \text{ for all w in W}$$

$$\text{Descent direction}: \sum_{w \text{ in } W} \sum_{p \text{ in } P(w)} \frac{\partial D(x)}{\partial x(p)} \Delta x(p) \leq 0$$

Network Coding with a Cost Criterion

- We consider applying network coding in settings where there is a cost associated with network use.
- Main results:
  - We specify a linear optimization problem that determines minimum-cost routing of single multicast connections.
  - We describe a distributed algorithm for solving this problem.
  - We show how to apply our approach to the problem of minimum-energy multicast in wireless networks with omnidirectional antennas.
  - For general connections, we specify a linear optimization problem that promises a non-trivial cost improvement over any solution without coding.

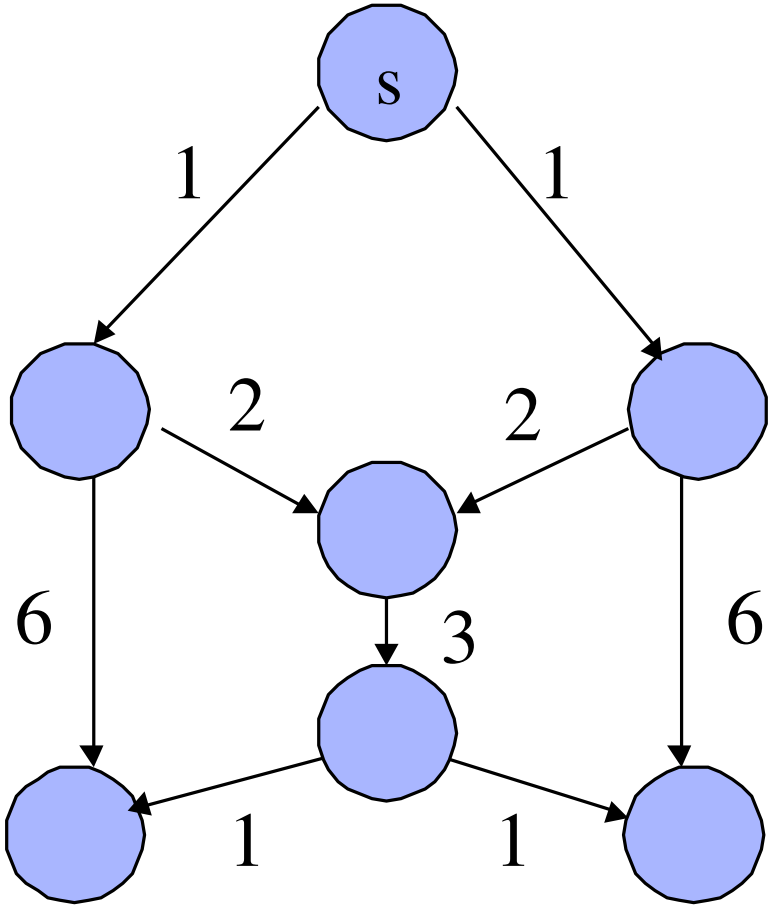
Multicast Connections

- For multicast problem where all sink nodes receive the same data a set of connections is feasible if and only if the connection rate satisfies the max-flow min-cut bound and can be achieved using linear coding (ACLY00)
- Linear operations over a sufficiently large finite field on a sufficiently long vector created from the source process (LYC03, KM03, HKMK03, JSCEEJT03)
- How do we incorporate cost in this setting?

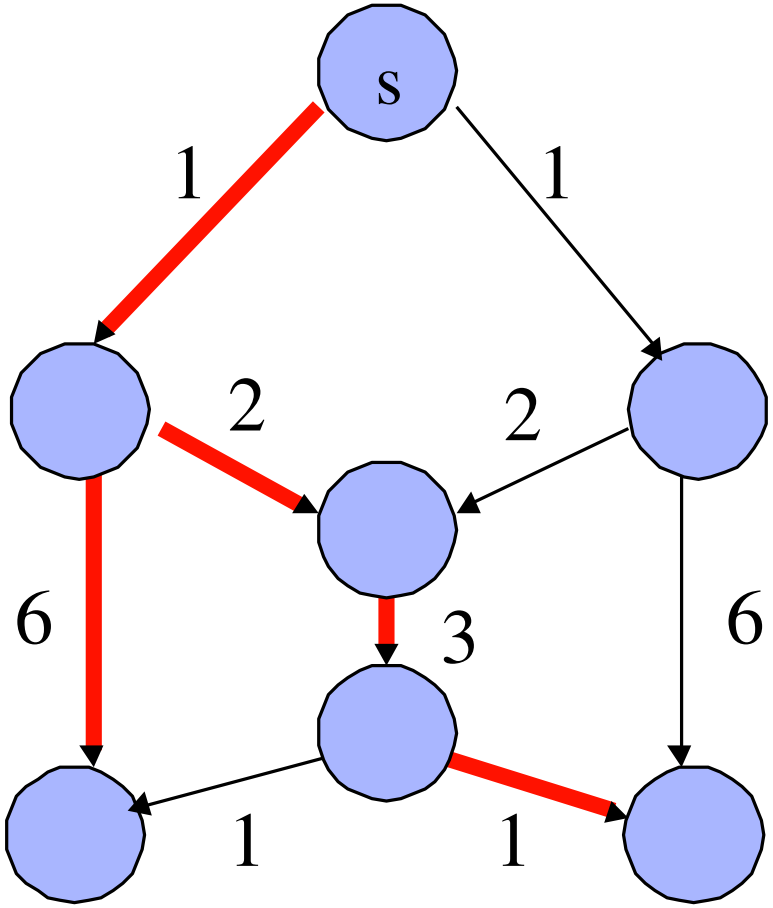
Model

- Directed graph $G = (N, A)$
- Each link (i, j) in A is associated with non-negative numbers a_{ij} and c_{ij}, which are the cost per unit flow and the capacity of the link, respectively
- The total cost of using a link is proportional to the flow on it (we discuss generalizations later)
- Source node s in N produces data at a positive, real rate R to transmit to a non-empty set of terminal nodes T in N.

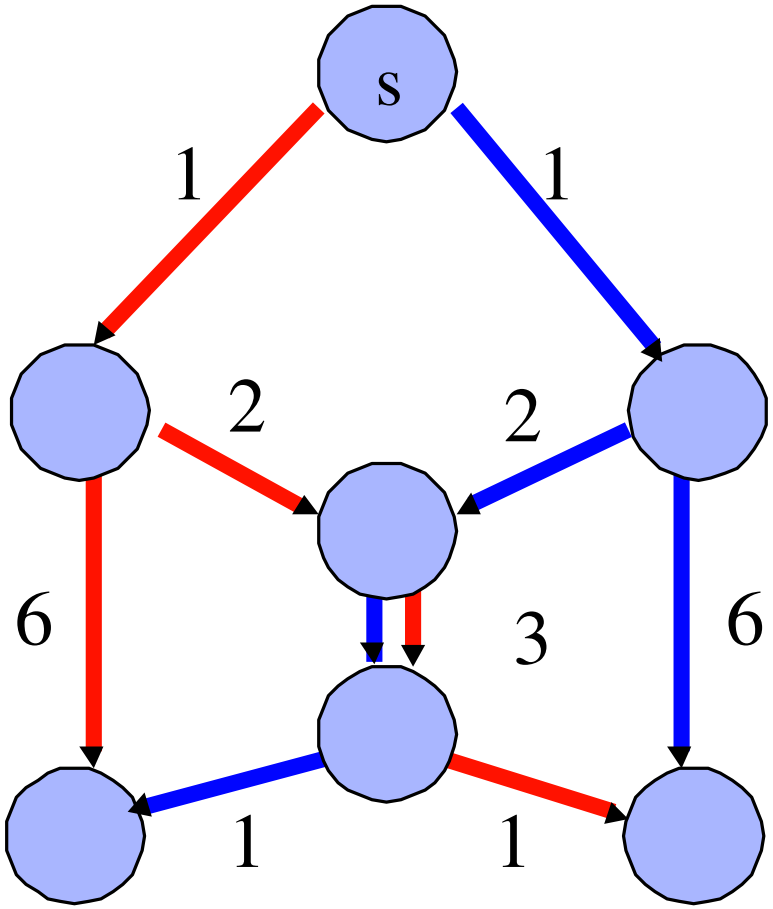
Network coding for cost



Network coding for cost

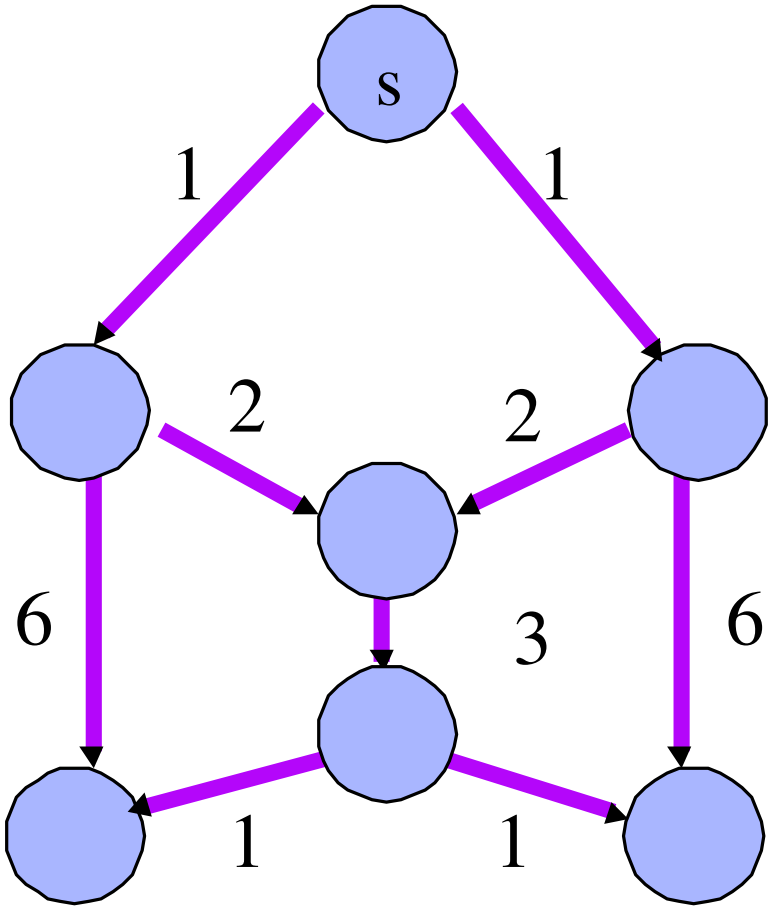


Network coding for cost



Cost of trees = 26

Network coding for cost



Cost of code = 23

Network coding for cost

- Without coding, the problem of multicast is the Steiner tree problem over dags, possibly with decompositions into several trees
- An immediately attractive approach would be to overlay trees to create codes, attempting to increase overlaps and counting only once several uses of a link - code is built automatically
- Complexity is high and does not make use of distributed random code construction, which works well in practice
- We have proposed a linear program statement of the problem which is polynomial-time, and can be solved in a **distributed manner**

A LP-based Solution [LMHK04]

$$
\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{(i,j)\in A} a_{ij} z_{ij} \\
\text{subject to} \quad & z_{ij} \geq x_{ij}^{(t)}, \ \forall\ (i,j) \in A,\ t \in T, \\
& \textstyle\sum_{\{j|(i,j)\in A\}} x_{ij}^{(t)} - \sum_{\{j|(j,i)\in A\}} x_{ji}^{(t)} = \\
& \begin{cases} R \text{ if } i = s, \\ -R \text{ if } i = t, \\ 0 \text{ otherwise,} \end{cases} \quad \forall\ i \in N,\ t \in T, \\
& c_{ij} \geq x_{ij}^{(t)} \geq 0, \ \forall\ (i,j) \in A,\ t \in T.
\end{aligned}
\tag{1}
$$

A LP-based Solution

- The vector z is part of a feasible solution for the LP problem if and only if there exists a network code that sets up a multicast connection in the graph G at rate arbitrarily close to R from source s to terminals in the set T and that puts a flow arbitrarily close to z_{ij} on each link (i, j)
- Proof follows almost immediately from min-cut max-flow necessary and sufficient conditions
- Polynomial-time
- Steiner-tree problem can be seen to be this problem with extra integrality constraints

Usefulness of LP

- We can extend this approach to other types of cost functions, for instant typical cost functions used to represent cost of congestion
- Can use to obtain equivalence of distances in networks, possibly extend minimum first derivative length approaches
- Concept of distances particularly useful for distributed routing - does the same apply here?

A Distributed Approach

Consider the dual problem

$$\begin{array}{lll} \text{maximize} & \sum_{t\in T} q^{(t)}(p^{(t)}) & \\ \text{subject to} & \sum_{t\in T} p_{ij}^{(t)} = a_{ij} & \forall\ (i,j)\in A, \\ & p_{ij}^{(t)} \geq 0 & \forall\ (i,j)\in A,\ t\in T, \end{array} \tag{2}$$

where

$$q^{(t)}(p^{(t)}) = \min_{x^{(t)}\in F^{(t)}} \sum_{(i,j)\in A} p_{ij}^{(t)} x_{ij}^{(t)}, \tag{3}$$

and $F^{(t)}$ is the bounded polyhedron of points $x^{(t)}$ satisfying the conservation of flow constraints and capacity constraints

Subgradient Approach

- Consider a subgradient approach (NW99)
- Start with an iterate $p[0]$ in the feasible set
- Solve subproblem (3) for each t in T to obtain $x[n]$
- $$p_{ij}[n+1] :=_{v \in P_{ij}} \sum_{t \in T} (v^{(t)} - (p_{ij}^{(t)}[n] + \theta[n] x_{ij}^{(t)}[n]))^2 \qquad (4)$$

  for each $(i,j) \in A$, where P_{ij} is the $|T|$-dimensional simplex

  $$P_{ij} = \left\{ v \left| \sum_{t \in T} v^{(t)} = a_{ij},\ v \geq 0 \right. \right\} \qquad (5)$$

  and $\theta[n] > 0$ is an appropriate step size
- $p_{ij}[n+1]$ is set to be the Euclidean projection of $p_{ij}[n] + \theta[n] x_{ij}[n]$ onto P_{ij}

Step Size Selection

- $u := p_{ij}[n] + \theta[n]x_{ij}[n]$
- We index the elements of T such that $u^{(t_1)} \geq u^{(t_2)} \geq \ldots \geq u^{(t_{|T|})}$
- Take k^* to be the smallest k such that

$$\frac{1}{k}\left(a_{ij} - \sum_{r=1}^{t_k} u^{(r)}\right) \leq -u^{(t_{k+1})} \tag{6}$$

  or set $k^* = |T|$ if no such k exists

- Projection is achieved by

$$p_{ij}^{(t)}[n+1] = \begin{cases} u^{(t)} + \frac{1}{k^*}\left(a_{ij} - \sum_{r=1}^{t_{k^*}} u^{(r)}\right) & \text{if } t \in \{t_1, \ldots, t_{k^*}\}, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Distributed Approach - Bringing it Together

- Problem of recovering primal from approximation of dual
- Use approach of (SC96) for obtaining primal from subgradient approximation to dual
- The conditions can be coalesced into a single algorithm to iterate in a distributed fashion towards the correct cost
- There is inherent robustness to change of costs, as in classical distributed Bellman-Ford approach to routing

Application - Wireless Networks

- Omnidirectional antennas - when transmitting from node i to node j, we get transmission to all nodes whose distance from i is less than that from i to j "for free"
- We consider energy efficiency
- We do not consider interference (bursty set-up, for instance)
- We impose an ordering $\preceq$ on the set of outgoing links from node i, such that $(i,j) \preceq (i,k)$ if and only if $a_{ij} \leq a_{ik}$
- Typically, the set of outgoing links from i will be the set of all nodes within a certain, fixed radius of i and the cost a_{ij} of the link between nodes i and j will be proportional to their distance raised to some power α, where $\alpha \geq 2$

LP for Wireless Network

- Owing to the omnidirectionality of the antennas, flow can be pushed from i to j by pushing it to any node k such that $(i,k) \in A$ and $(i,k) \succeq (i,j)$
- Thus, the maximum flow $x_{ij}^{(t)}$ that can be pushed for a given t in T is

$$z_{ij} + \sum_{\{k|(i,k)\in A,(i,k)\succeq(i,j)\}\setminus\{j\}} (z_{ik} - x_{ik}^{(t)}) \tag{8}$$

- Hence

$$\sum_{\{k|(i,k)\in A,(i,k)\succeq(i,j)\}} (z_{ik} - x_{ik}^{(t)}) \geq 0 \tag{9}$$

for all $t \in T$.

LP for Wireless Network

$$
\begin{aligned}
&\text{minimize} \sum_{(i,j)\in A} a_{ij} z_{ij} \\
&\text{subject to} \\
&\sum_{\{k|(i,k)\in A,(i,k)\succeq(i,j)\}} (z_{ik} - x_{ik}^{(t)}) \geq 0, \quad \forall\ (i,j) \in A',\ t \in T, \\
&\sum_{\{j|(i,j)\in A\}} x_{ij}^{(t)} - \sum_{\{j|(j,i)\in A\}} x_{ji}^{(t)} = \\
&\qquad \begin{cases} R \text{ if } i = s, \\ -R \text{ if } i = t, \quad \forall\ i \in N,\ t \in T, \\ 0 \text{ otherwise}, \end{cases} \\
&\qquad c_{ij} \geq x_{ij}^{(t)} \geq 0, \quad \forall\ (i,j) \in A,\ t \in T,
\end{aligned}
\tag{10}
$$

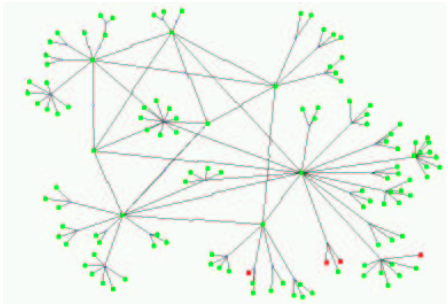
Beyond Optimality - Non-Multicast Connections

- M source processes $X_1, \ldots, X_M$ with rates $R_1, \ldots, R_M$, respectively, which are generated at (possibly different) nodes $s_1, \ldots, s_M$ in N
- Each sink $t \in T$ demands a subset of the source process that are generated in the network, which we specify with the set $D(t) \subset \{1, \ldots, M\}$
- Optimization may not be possible in the case of multiple multicast - indeed the sufficiency of linearity is itself in question
- Can we at least present a solution for finding a low-cost subgraph and a linear code that will perform no worse than routing?

Conclusions

- Network coding with a cost criterion naturally brings in traditional cost approaches for efficient networking
- For multicast, we establish a LP that can be solved in a distributed fashion
- For non-multicast, a LP approach can still perform at least as well as routing, with an explicit code construction
- We propose an optimistic view of network coding to consider the many possibilities afforded by it over routing, without requiring the full knowledge of sufficiency conditions in the non-multicast set-up

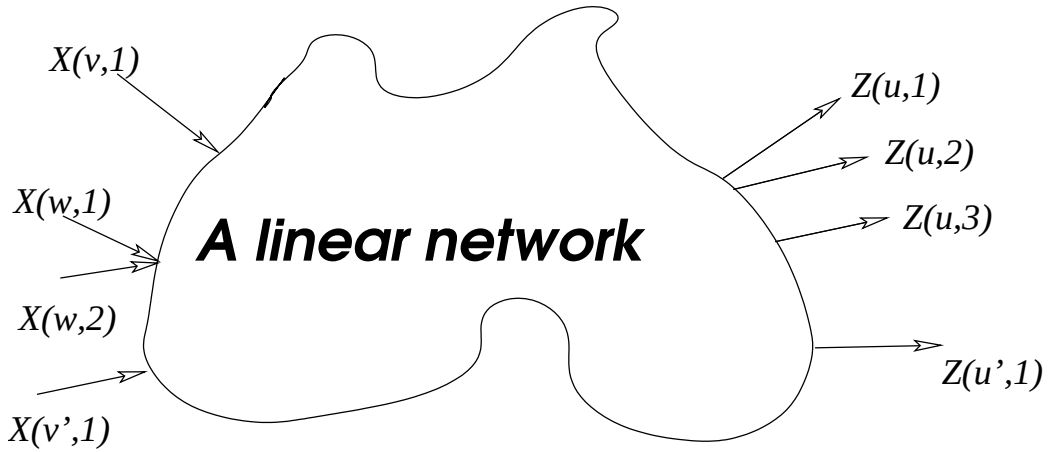
VI — The non multicast case



Questions

- What do we know about the non-multicast case?
- Vector solutions versus instantaneous solutions.
- The issue of linearity!
- Is the non-multicast case interesting?

The algebraic setup



Input vector: $\underline{x}^T = (X(v,1), X(v,2), \ldots, X(v', \mu(v')))$

Output vector: $\underline{z}^T = (Z(u,1), Z(u,2), \ldots, Z(u', \nu(u')))$

Transfer matrix: M, $\underline{z} = M\underline{x} = B \cdot G \cdot A\ \underline{x}$

$\underline{\xi} = (\xi_1, \xi_2, \ldots,) = (\ldots, \alpha_{e,l}, \ldots, \beta_{e',e}, \ldots, \varepsilon_{e',j}, \ldots)$

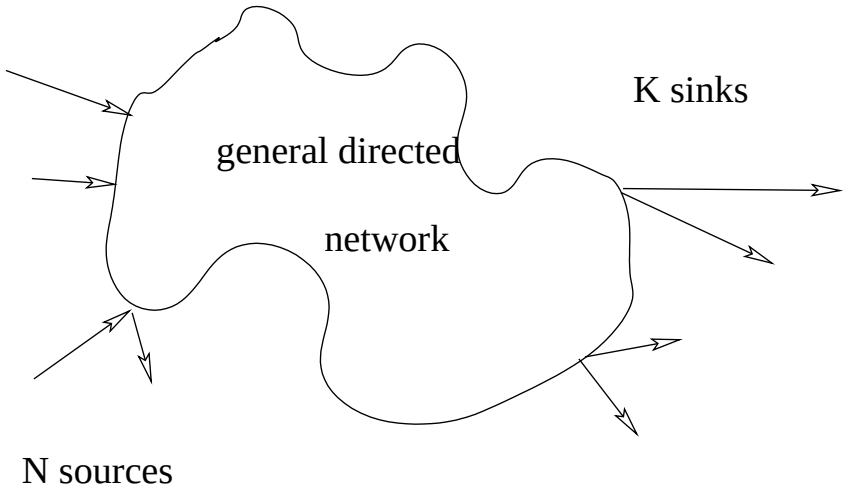
$$\underline{z} = M\underline{x} = B \cdot \underbrace{(I - F^T)^{-1}}_{G^T} \cdot A\ \underline{x}$$

$$\underline{\xi} = (\xi_1, \xi_2, \ldots,) = (\ldots, \alpha_{e,l}, \ldots, \beta_{e',e}, \ldots, \varepsilon_{e',j}, \ldots)$$

For acyclic networks the elements of G (and hence M) are polynomial functions in variables $\underline{\xi} = (\xi_1, \xi_2, \ldots,)$

$\Rightarrow$ an algebraic characterization of flows....

General Problems $(\mathcal{G}, \mathcal{C})$



$$\mathcal{C} = \{(v_i, u_j, \mathcal{X}(v_i, u_j))\}$$

$$M = \begin{pmatrix} M_{1,1} & M_{1,2} & \dots & M_{1,K} \\ M_{2,1} & M_{2,2} & & M_{2,K} \\ \vdots & & & \vdots \\ M_{N,1} & M_{N,2} & \dots & M_{N,K} \end{pmatrix}$$

$M_{i,j}$ corresponds to $c_{i,j} = (v_i, u_j, \mathcal{X}(v_i, u_j))$.

Theorem [Generalized Min-Cut Max-Flow Condition] Let an acyclic, delay-free scalar linear network problem $(\mathcal{G}, \mathscr{C})$ be given and let $M = \{M_{i,j}\}$ be the corresponding transfer matrix relating the set of input nodes to the set of output nodes. The network problem is solvable if and only if there exists an assignment of numbers to $\underline{\xi}$ such that

1. $M_{i,j} = 0$ for all pairs (v_i, v_j) of vertices such that $(v_i, v_j, \mathscr{X}(v_i, v_j)) \notin \mathscr{C}$.

2. If $\mathscr{C}$ contains the connections $(v_{i_1}, v_j, \mathscr{X}(v_{i_1}, v_j)), (v_{i_2}, v_j, \mathscr{X}(v_{i_2}, v_j)), \ldots, (v_{i_\ell}, v_j, \mathscr{X}(v_{i_\ell}, v_j))$ the determinant of $\left[M^T_{i_1,j} M^T_{i_2,j}, \ldots, M^T_{i_\ell,j}\right]$ is nonzero.

The ideal of $(\mathcal{G}, \mathscr{C})$

Entries in $M_{i,j}$ that have to evaluate to zero: $f_1(\underline{\xi}), f_2(\underline{\xi}), \ldots, f_L(\underline{\xi})$

Determinants of submatrices that have to evaluate to nonzero values: $g_1(\underline{\xi}), g_2(\underline{\xi}), \ldots, g_{L'}(\underline{\xi})$

$$\mathbf{Ideal}((\mathcal{G}, \mathscr{C}))$$

$$= \langle f_1(\underline{\xi}), f_2(\underline{\xi}), \ldots, f_L(\underline{\xi}), 1 - \xi_0 \prod_{i=1}^{L'} g_i(\underline{\xi}) \rangle$$

$$\mathbf{Var}((\mathcal{G}, \mathscr{C})) = \{(a_1, a_2, \ldots, a_n) \in \bar{\mathbb{F}}^n :$$

$$f(a_1, a_2, \ldots, a_n) = 0 \; \forall \; f \in \mathbf{Ideal}((\mathcal{G}, \mathscr{C}))\}.$$

The central Theorem

Theorem Let a scalar linear network problem $(\mathcal{G}, \mathscr{C})$ be given. The network problem is solvable if and only if Var$((\mathcal{G}, \mathscr{C})$ is nonempty or equivalently , the ideal Ideal$((\mathcal{G}, \mathscr{C}))$ is a proper ideal of $\bar{\mathbb{F}}[\xi_0, \underline{\xi}]$, i.e Ideal$((\mathcal{G}, \mathscr{C})) \subsetneq \mathbb{F}_2[\xi_0, \underline{\xi}]$.

So why is the gerenal case so much harder?

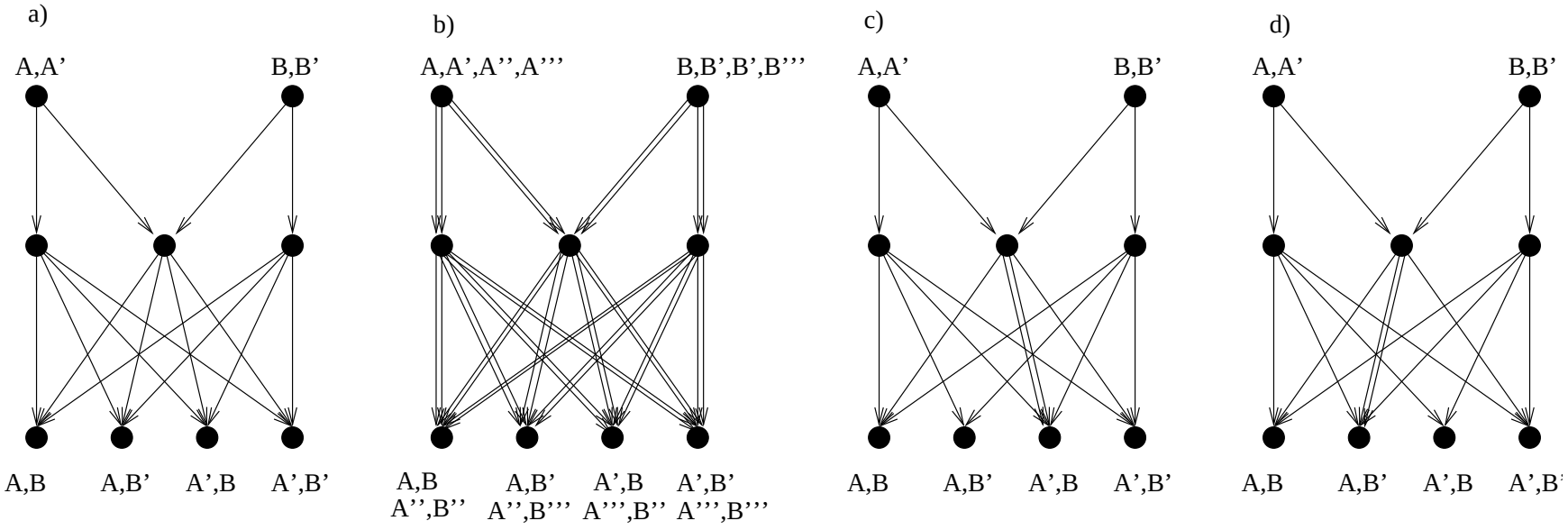
For the general case we need to find solutions to some system of polynomial equations!

For the multicast case we need to find non solutions to some system of polynomial equations!

Another way to phrase this is: In a multicast setup everybody wants everything so the issue of interference is moot!

For the general case we may have carefully balanced solutions where some unwanted information cancels out in clever ways.....

Vector solutions may help

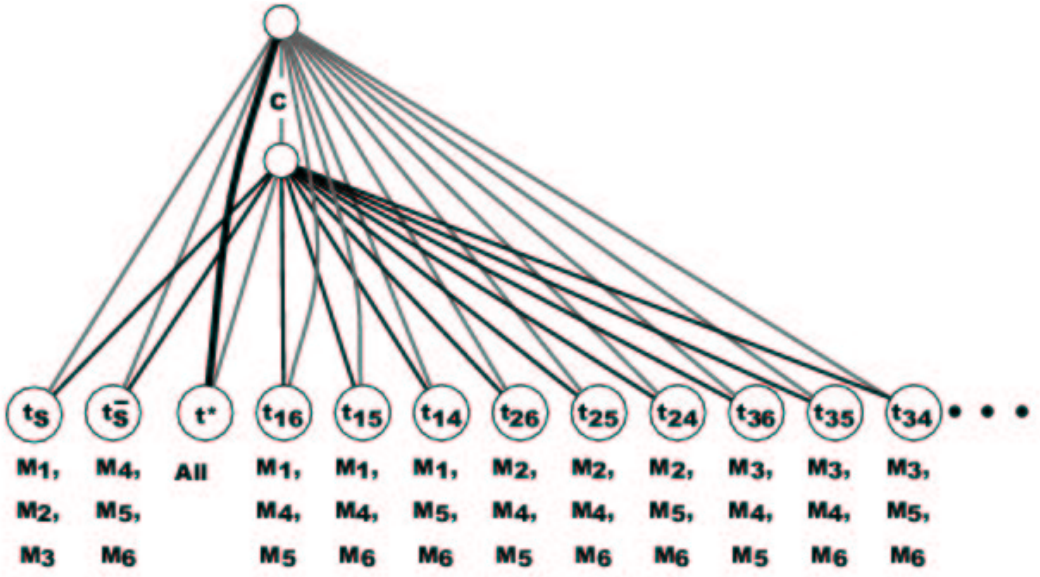


In the general problem a time sharing combination of several solutions which themselves both violate the constraints may be necessary. (This cannot happen in the multicast case!)

Doubling the bandwidth more than doubles capacity! Tripling the bandwidth does not work!

Vector solutions may help

From: A. Rasala-Lehman and E. Lehman, "Vector-Linear Network Codes: Is the Model Broken", preprint, March 2004



Examples of networks that need vector length that are multiples of k for any k.

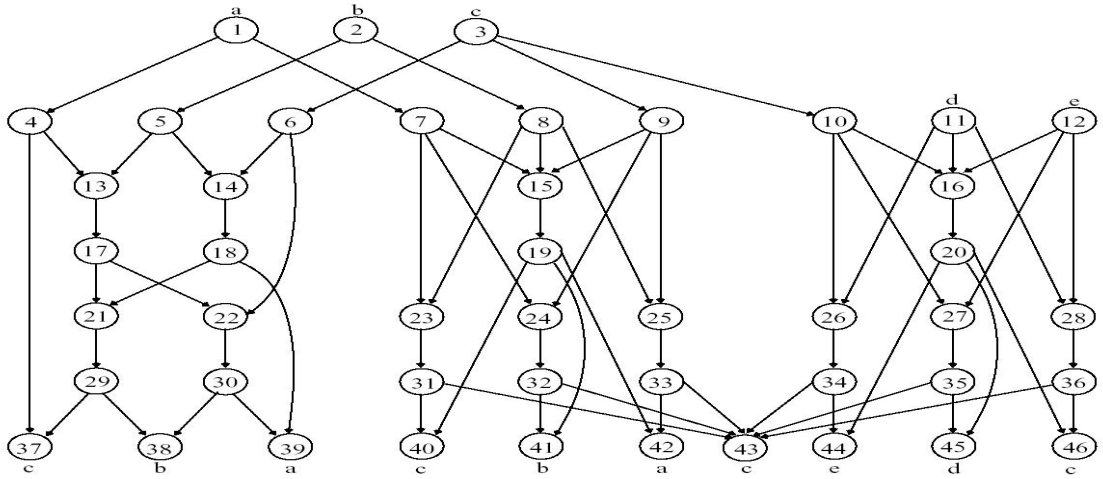
From: A. Rasala-Lehman and E. Lehman, "Vector-Linear Network Codes: Is the Model Broken", preprint, March 2004

By combining networks requiring vector length that are multiples of primes the following bound is derived:

Theorem There exist directed networks with O(n) nodes such that a solution to the network coding problem requires at least an alphabet size of $2^{(e^{\sqrt{n^{1/3}}})}$

More strange news...

R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of Linear Coding in Network Information Flow", preprint, February 2004



This network is not solvable over any Galois field, including vector versions thereof!

(still the network has a distinctly linear feel to it....)

Non-multicast connections -use of cost criterion

- We propose a linear optimization problem whose minimum cost is no greater than the minimum cost of any routing solution
- Moreover, feasible solutions correspond to network codes that perform linear operations on vectors created from the source processes
- Main idea: create a set partition of $\{1, \ldots, M\}$ that represents the sources that can be mixed (combined linearly) on links going into i.
- Code construction steps through the nodes in topological order, examining the outgoing links and defining global coding vectors on them.

Non-multicast connections -use of cost criterion

- For any node i, let $T(i)$ denote the sinks that are accessible from i
- Let $\mathcal{C}(i)$ be a set partition of $\{1, \ldots, M\}$ that represents the sources that can be mixed (combined linearly) on links going into i. For a given $C \in \mathcal{C}(i)$, the sinks that receive a source process in C by way of link (j, i) in A (set of arcs) either receive all the source processes in C or none at all.

Non-multicast connections -use of cost criterion

$$
\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{(i,j)\in A} a_{ij} z_{ij} \\
\text{subject to} \quad & c_{ij} \geq z_{ij} = \textstyle\sum_{C \in \mathcal{C}(j)} y_{ij}^{(C)}, && \forall\, (i,j) \in A, \\
& y_{ij}^{(C)} \geq \textstyle\sum_{m \in C} x_{ij}^{(t,m)}, && \forall\, (i,j) \in A,\, t \in T,\, C \in \mathcal{C}(j), \\
& x_{ij}^{(t,m)} \geq 0, && \forall\, (i,j) \in A,\, t \in T,\, m = 1, \ldots, M,
\end{aligned}
$$

$$
\sum_{\{j|(i,j)\in A\}} x_{ij}^{(t,m)} - \sum_{\{j|(j,i)\in A\}} x_{ji}^{(t,m)} = \begin{cases} R_m & \text{if } v = s_m \text{ and } m \in D(t), \\ -R_m & \text{if } m \in D(i), \\ 0 & \text{otherwise,} \end{cases}
$$

$$
\forall\, i \in A,\, t \in T,\, m = 1, \ldots, M, \quad (1)
$$

where we define $D(i) := \emptyset$ for i in $N \setminus T$. Again, the optimization problem can be easily modified to accommodate convex cost functions.

Is the non-multicast case interesting?

Summary:

The non multicast scenario exhibits far more subtleties than the multicast setup. This is due to the fact that cancellations now need to be carefully arranged.

There are some generalizations to vector solutions which can be incorporated into the algebraic framework.

Not even the principle problem of linearity vs. nonlinear operation is entirely clear.

From a practical point of view a non interacting arrangement of multicast is most interesting and robust.

Network coding for multicast relation to compression and generalization of Slepian-Wolf

Overview

Review of Slepian-Wolf

Distributed network compression

Error exponents Source-channel separation issues

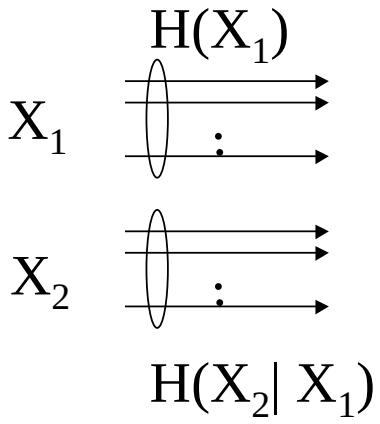
Code construction for finite field multiple access networks

Distributed data compression

Consider two correlated sources $(X, Y) \sim p(x, y)$ that must be separately encoded for a user who wants to reconstruct both

What information transmission rates from each source allow decoding with arbitrarily small probability of error?

E.g.



Distributed source code

A $((2^{nR_1}, 2^{nR_2}), n)$ distributed source code for joint source (X, Y) consists of encoder maps

$$\begin{aligned} f_1 : \mathcal{X}^n &\rightarrow \{1, 2, \ldots, 2^{nR_1}\} \\ f_2 : \mathcal{Y}^n &\rightarrow \{1, 2, \ldots, 2^{nR_2}\} \end{aligned}$$

and a decoder map

$$g : \{1, 2, \ldots, 2^{nR_1}\} \quad \{1, 2, \ldots, 2^{nR_2}\} \rightarrow \mathcal{X}^n \quad \mathcal{Y}^n$$

- X^n is mapped to $f_1(X^n)$
- Y^n is mapped to $f_2(Y^n)$
- (R_1, R_2) is the rate pair of the code

Probability of error

$$P_e^{(n)} = \Pr\{g(f_1(X^n), f_2(Y^n)) \neq (X^n, Y^n)\}$$

Slepian-Wolf

Definitions:

A rate pair (R_1, R_2) is *achievable* if there exists a sequence of $((2^{nR_1}, 2^{nR_2}), n)$ distributed source codes with probability of error $P_e^{(n)} \to 0$ as $n \to \infty$

achievable rate region - closure of the set of achievable rates

Slepian-Wolf Theorem:

For the distributed source coding problem for source (X, Y) drawn i.i.d. $\sim p(x, y)$, the achievable rate region is

$$\begin{aligned} R_1 &\qquad H(X|Y) \\ R_2 &\qquad H(Y|X) \\ R_1 + R_2 &\qquad H(X, Y) \end{aligned}$$

Proof of achievability

Main idea: show that if the rate pair is in the Slepian-Wolf region, we can use a random binning encoding scheme with typical set decoding to obtain a probability of error that tends to zero

Coding scheme:

Source X assigns every sourceword $x \in \mathcal{X}^n$ randomly among 2^{nR_1} bins, and source Y independently assigns every $y \in \mathcal{Y}^n$ randomly among 2^{nR_2} bins

Each sends the bin index corresponding to the message

the receiver decodes correctly if there is exactly one jointly typical sourceword pair corresponding to the received bin indexes, otherwise it declares an error

Random binning for single source compression

An encoder that knows the typical set can compress a source X to $H(X)+\epsilon$ without loss, by employing separate codes for typical and atypical sequences

Random binning is a way to compress a source X to $H(X)+\epsilon$ with asymptotically small probability of error without the encoder knowing the typical set, as well as the decoder knows the typical set

the encoder maps each source sequence X^n uniformly at random into one of 2^{nR} bins

the bin index, which is R bits long, forms the code

the receiver decodes correctly if there is exactly one typical sequence corresponding to the received bin index

Error analysis

An error occurs if:

a) the transmitted sourceword is not typical, i.e. event

$$E_0 = \{\mathbf{X} \notin A_\epsilon^{(n)}\}$$

b) there exists another typical sourceword in the same bin, i.e.event

$$E_1 = \{\exists \mathbf{x}' \neq \mathbf{X} : f(\mathbf{x}') = f(\mathbf{X}), \mathbf{x}' \in A_\epsilon^{(n)}\}$$

Use union of events bound:

$$\begin{aligned} P_e^{(n)} &= \Pr(E_0 \cup E_1) \\ &\quad \Pr(E_0) + \Pr(E_1) \end{aligned}$$

Error analysis continued

$\Pr(E_0) \to 0$ by the Asymptotic Equipartition Property (AEP)

$$
\begin{aligned}
\Pr(E_1) &= \sum_{\mathbf{x}} \Pr\{\exists \mathbf{x}' \neq \mathbf{x} : f(\mathbf{x}') = f(\mathbf{x}), \\
&\qquad\qquad \mathbf{x}' \in A_\epsilon^{(n)}\} \\
&\quad \sum_{\mathbf{x}} \sum_{\substack{\mathbf{x}' \neq \mathbf{x} \\ \mathbf{x}' \in A_\epsilon^{(n)}}} \Pr(f(\mathbf{x}') = f(\mathbf{x})) \\
&= \sum_{\mathbf{x}} |A_\epsilon^{(n)}|\, 2^{-nR} \\
&\quad 2^{-nR}\, 2^{n(H(X)+\epsilon)} \\
&\to 0 \text{ if } R > H(X)
\end{aligned}
$$

For sufficiently large n,

$$\begin{aligned} & \Pr(E_0), \Pr(E_1) < \epsilon \\ \Rightarrow \quad & P_\epsilon^{(n)} < 2\epsilon \end{aligned}$$

Jointly typical sequences

The set $A_\epsilon^{(n)}$ of jointly typical sequences is the set of sequences $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}^n \quad \mathcal{Y}^n$ with probability:

$$2^{-n(H(X)+\epsilon)} \quad p_{\mathbf{X}}(\mathbf{x}) \quad 2^{-n(H(X)-\epsilon)}$$

$$2^{-n(H(Y)+\epsilon)} \quad p_{\mathbf{Y}}(\mathbf{y}) \quad 2^{-n(H(Y)-\epsilon)}$$

$$2^{-n(H(X,Y)+\epsilon)} \quad p_{\mathbf{X},\mathbf{Y}}(\mathbf{x}, \mathbf{y}) \quad 2^{-n(H(X,Y)-\epsilon)}$$

for $(\mathbf{X}, \mathbf{Y})$ sequences of length n IID according to $p_{\mathbf{X},\mathbf{Y}}(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^n p_{X,Y}(x_i, y_i)$

Size of typical set:

$$|A_\epsilon^{(n)}| \quad 2^{n(H(X,Y)+\epsilon)}$$

Proof:

$$
\begin{aligned}
1 &= \sum p(\mathbf{x},\mathbf{y}) \\
&\quad \sum_{A_\epsilon^{(n)}} p(\mathbf{x},\mathbf{y}) \\
&\quad |A_\epsilon^{(n)}| 2^{-n(H(X,Y)+\epsilon)}
\end{aligned}
$$

Conditionally typical sequences

The conditionally typical set $A_\epsilon^{(n)}(X|\mathbf{y})$ for a given typical $\mathbf{y}$ sequence is the set of $\mathbf{x}$ sequences that are jointly typical with the given $\mathbf{y}$ sequence.

Size of conditionally typical set:

$$|A_\epsilon^{(n)}(X|\mathbf{y})| \quad 2^{n(H(X|Y)+\epsilon)}$$

Proof:

For $(\mathbf{x}, \mathbf{y}) \in A_\epsilon^{(n)}(X, Y)$,

$$\begin{aligned} p(\mathbf{y}) &\doteq 2^{-n(H(Y)\pm\epsilon)} \\ p(\mathbf{x}, \mathbf{y}) &\doteq 2^{-n(H(X,Y)\pm\epsilon)} \\ \Rightarrow p(\mathbf{x}|\mathbf{y}) &= \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})} \\ &\doteq 2^{-n(H(X|Y)\pm 2\epsilon)} \end{aligned}$$

Hence

$$\begin{aligned} 1 &\qquad \sum_{\mathbf{x}\in A_\epsilon^{(n)}(X|\mathbf{y})} p(\mathbf{x}|\mathbf{y}) \\ &\qquad |A_\epsilon^{(n)}| 2^{-n(H(X|Y)+2\epsilon)} \end{aligned}$$

Proof of achievability – error analysis

Errors occur if:

a) the transmitted sourcewords are not jointly typical, i.e. event

$$E_0 = \{(X, Y) \notin A_\epsilon^{(n)}\}$$

b) there exists another pair of jointly typical sourcewords in the same pair of bins, i.e. one or more of the following events

$$\begin{aligned} E_1 &= \{\exists \mathbf{x}' \neq \mathbf{X} : f_1(\mathbf{x}') = f_1(\mathbf{X}), (\mathbf{x}', \mathbf{Y}) \in A_\epsilon^{(n)}\} \\ E_2 &= \{\exists \mathbf{y}' \neq \mathbf{Y} : f_2(\mathbf{y}') = f_2(\mathbf{Y}), (\mathbf{X}, \mathbf{y}') \in A_\epsilon^{(n)}\} \\ E_{12} &= \{\exists (\mathbf{x}', \mathbf{y}') : \mathbf{x}' \neq \mathbf{X}, \mathbf{y}' \neq \mathbf{Y}, f_1(\mathbf{x}') = f_1(\mathbf{X}), \\ &\quad f_2(\mathbf{y}') = f_2(\mathbf{Y}), (\mathbf{x}', \mathbf{y}') \in A_\epsilon^{(n)}\} \end{aligned}$$

Use union of events bound:

$$\begin{aligned} P_e^{(n)} &= \Pr(E_0 \cup E_1 \cup E_2 \cup E_{12}) \\ &\quad \Pr(E_0) + \Pr(E_1) + \Pr(E_2) + \Pr(E_{12}) \end{aligned}$$

Error analysis continued

$\Pr(E_0) \to 0$ by the AEP

$$
\begin{aligned}
\Pr(E_1) &= \sum_{(\mathbf{x},\mathbf{y})} \Pr\{\exists \mathbf{x}' \neq \mathbf{x} : f_1(\mathbf{x}') = f_1(\mathbf{x}), \\
&\qquad\qquad (\mathbf{x}', \mathbf{y}) \in A_\epsilon^{(n)}\} \\
&\quad \sum_{(\mathbf{x},\mathbf{y})} \sum_{\substack{\mathbf{x}' \neq \mathbf{x} \\ (\mathbf{x}', \mathbf{y}) \in A_\epsilon^{(n)}}} \Pr(f_1(\mathbf{x}') = f_1(\mathbf{x})) \\
&= \sum_{(\mathbf{x},\mathbf{y})} |A_\epsilon^{(n)}(X|\mathbf{y})|\, 2^{-nR_1} \\
&\quad 2^{-nR_1}\, 2^{n(H(X|Y)+2\epsilon)} \\
&\to 0 \text{ if } R_1 > H(X|Y)
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
\Pr(E_2) \quad & 2^{-nR_2}\, 2^{n(H(Y|X)+2\epsilon)} \\
\rightarrow \quad & 0 \text{ if } R_2 > H(Y|X) \\
\Pr(E_{12}) \quad & 2^{-n(R_1+R_2)}\, 2^{n(H(X,Y)+\epsilon)} \\
\rightarrow \quad & 0 \text{ if } R_1 + R_2 > H(X,Y)
\end{aligned}
$$

Error analysis continued

Thus, if we are in the Slepian-Wolf rate region, for sufficiently large n,

$$\begin{aligned} & \Pr(E_0), \Pr(E_1), \Pr(E_2), \Pr(E_{12}) < \epsilon \\ \Rightarrow \quad & P_\epsilon^{(n)} < 4\epsilon \end{aligned}$$

Since the average probability of error is less than 4ϵ, there exist at least one code (f_1^*, f_2^*, g^*) with probability of error $< 4\epsilon$.

Thus, there exists a sequence of codes with $P_\epsilon^{(n)} \to 0$.

Model for distributed network compression

arbitrary directed graph with integer capacity links

discrete memoryless source processes with integer bit rates

randomized linear network coding over vectors of bits in $\mathbb{F}_2$

coefficients of overall combination transmitted to receivers

receivers perform minimum entropy or maximum a posteriori probability decoding

Distributed compression problem

Consider

two sources of bit rates r_1, r_2, whose output values in each unit time period are drawn i.i.d. from the same joint distribution Q

linear network coding in $\mathbb{F}_2$ over vectors of nr_1 and nr_2 bits from each source respectively

Define

m_1 and m_2 the minimum cut capacities between the receiver and each source respectively

m_3 the minimum cut capacity between the receiver and both sources

L the maximum source-receiver path length

Theorem 1 *The error probability at each receiver using minimum entropy or maximum a posteriori probability decoding is at most* $\sum_{i=1}^{3} p_e^i$, *where*

$$
\begin{aligned}
p_e^1 \;\leq\; & \exp\left\{ -n \min_{X_1,X_2} \; D(P_{X_1X_2}||Q) \right. \\
& \left. + \left| m_1(1-\frac{1}{n}\log L) - H(X_1|X_2) \right|^+ \; + 2^{2r_1+r_2}\log(n+1) \right\} \\
p_e^2 \;\leq\; & \exp\left\{ -n \min_{X_1,X_2} \; D(P_{X_1X_2}||Q) \right. \\
& \left. + \left| m_2(1-\frac{1}{n}\log L) - H(X_2|X_1) \right|^+ \; + 2^{r_1+2r_2}\log(n+1) \right\} \\
p_e^3 \;\leq\; & \exp\left\{ -n \min_{X_1,X_2} \; D(P_{X_1X_2}||Q) \right. \\
& \left. + \left| m_3(1-\frac{1}{n}\log L) - H(X_1X_2) \right|^+ \; + 2^{2r_1+2r_2}\log(n+1) \right\}
\end{aligned}
$$

Distributed compression

Redundancy is removed or added in different parts of the network depending on available capacity

Achieved without knowledge of source entropy rates at interior network nodes

For the special case of a Slepian-Wolf source network consisting of a link from each source to the receiver, the network coding error exponents reduce to known error exponents for linear Slepian-Wolf coding [Csi82]

Proof outline

Error probability $\sum_{i=1}^{3} p_e^i$, where

- p_e^1 is the probability of correctly decoding X_2 but not X_1,
- p_e^2 is the probability of correctly decoding X_1 but not X_2
- p_e^3 is the probability of wrongly decoding X_1, X_2

Proof approach using method of types similar to that in [Csi82]

Types $P_{\mathbf{x}_i}$, joint types $P_{\mathbf{xy}}$ are the empirical distributions of elements in vectors $\mathbf{x}_i$

Proof outline (cont'd)

Bound error probabilities by summing over

sets of joint types

$$\mathcal{P}_n^i = \begin{cases} \{P_{X_1\tilde{X}_1X_2\tilde{X}_2} \mid \tilde{X}_1 \neq X_1, \tilde{X}_2 = X_2\} & i = 1 \\ \{P_{X_1\tilde{X}_1X_2\tilde{X}_2} \mid \tilde{X}_1 = X_1, \tilde{X}_2 \neq X_2\} & i = 2 \\ \{P_{X_1\tilde{X}_1X_2\tilde{X}_2} \mid \tilde{X}_1 \neq X_1, \tilde{X}_2 \neq X_2\} & i = 3 \end{cases}$$

where $X_i, \tilde{X}_i \in \mathbb{F}_2^{nr_i}$

sequences of each type

$$\mathcal{T}_{X_1X_2} = \left\{ [\ \mathbf{x}\ \ \mathbf{y}\ ] \in \mathbb{F}_2^{n(r_1+r_2)} \;\middle|\; P_{\mathbf{xy}} = P_{X_1X_2} \right\}$$

$$\mathcal{T}_{\tilde{X}_1\tilde{X}_2|X_1X_2}(\mathbf{xy}) = \left\{ [\ \tilde{\mathbf{x}}\ \ \tilde{\mathbf{y}}\ ] \in \mathbb{F}_2^{n(r_1+r_2)} \;\middle|\; P_{\tilde{\mathbf{x}}\tilde{\mathbf{y}}\mathbf{xy}} = P_{\tilde{X}_1\tilde{X}_2X_1X_2} \right\}$$

Proof outline (cont'd)

Define

- $P_i, i = 1, 2$, the probability that distinct $(\mathbf{x}, \mathbf{y}), (\tilde{\mathbf{x}}, \mathbf{y})$, where $\mathbf{x} \neq \tilde{\mathbf{x}}$, at the receiver

- P_3, the probability that $(\mathbf{x}, \mathbf{y}), (\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$, where $\mathbf{x} \neq \tilde{\mathbf{x}}$,$\mathbf{y} \neq \tilde{\mathbf{y}}$, are mapped to the same output at the receiver

These probabilities can be calculated for a given network, or bounded in terms of block length n and network parameters

Proof outline (cont'd)

A link with 1 nonzero incoming signal carries the zero signal with probability $\frac{1}{2^{nc}}$, where c is the link capacity

this is equal to the probability that a pair of distinct input values are mapped to the same output on the link

We can show by induction on the minimum cut capacities m_i that

$$P_i \quad 1 \quad (1 \quad \frac{1}{2^n})^{L \quad m_i} \quad \frac{L}{2^n} \quad m_i$$

Proof outline (cont'd)

We substitute in

cardinality bounds

$$\begin{aligned} |\mathcal{P}_n^1| &< (n+1)^{2^{2r_1+r_2}} \\ |\mathcal{P}_n^2| &< (n+1)^{2^{r_1+2r_2}} \\ |\mathcal{P}_n^3| &< (n+1)^{2^{2r_1+2r_2}} \\ |\mathcal{T}_{X_1X_2}| & \quad \exp\{nH(X_1X_2)\} \\ |\mathcal{T}_{\tilde{X}_1\tilde{X}_2|X_1X_2}(\mathbf{xy})| & \quad \exp\{nH(\tilde{X}_1\tilde{X}_2|X_1X_2)\} \end{aligned}$$

probability of source vector of type $(\mathbf{x}, \mathbf{y}) \in \mathcal{T}_{X_1X_2}$

$$Q^n(\mathbf{xy}) = \exp\{ \ n(D(P_{X_1X_2}||Q) + H(X_1X_2))\}$$

Proof outline (cont'd)

and the decoding conditions

minimum entropy decoder:

$$H(\tilde{X}_1\tilde{X}_2) \quad H(X_1X_2)$$

maximum a posteriori probability decoder:

$$D(P_{\tilde{X}_1\tilde{X}_2}||Q) + H(\tilde{X}_1\tilde{X}_2) \quad D(P_{X_1X_2}||Q) + H(X_1X_2)$$

to obtain the result

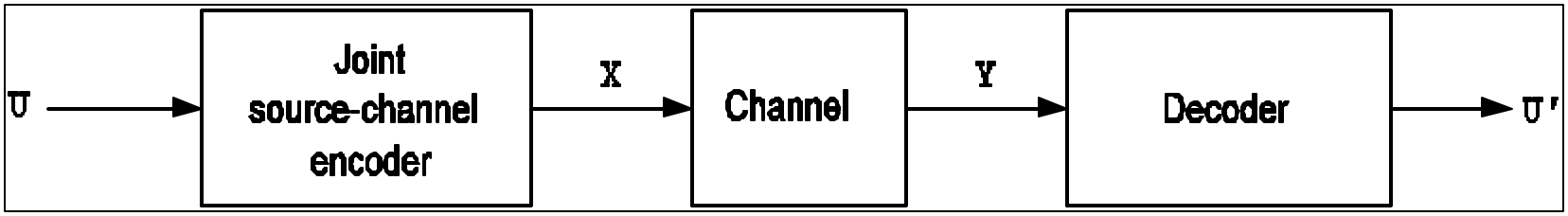
Conclusions

Distributed randomized network coding can achieve distributed compression of correlated sources

Error exponents generalize results for linear Slepian Wolf coding

Further work: investigation of non-uniform code distributions, other types of codes, and other decoding schemes

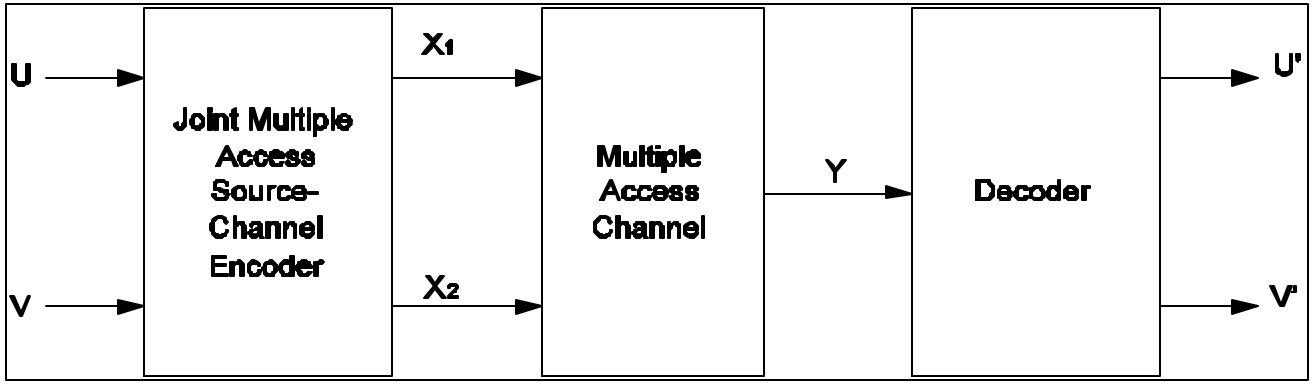
Coding Theorem – Point to Point Communication



We can transmit the discrete source U with arbitrarily small error probability iff

$$H(U) < C = \sup_{P_X} I(X;Y)$$

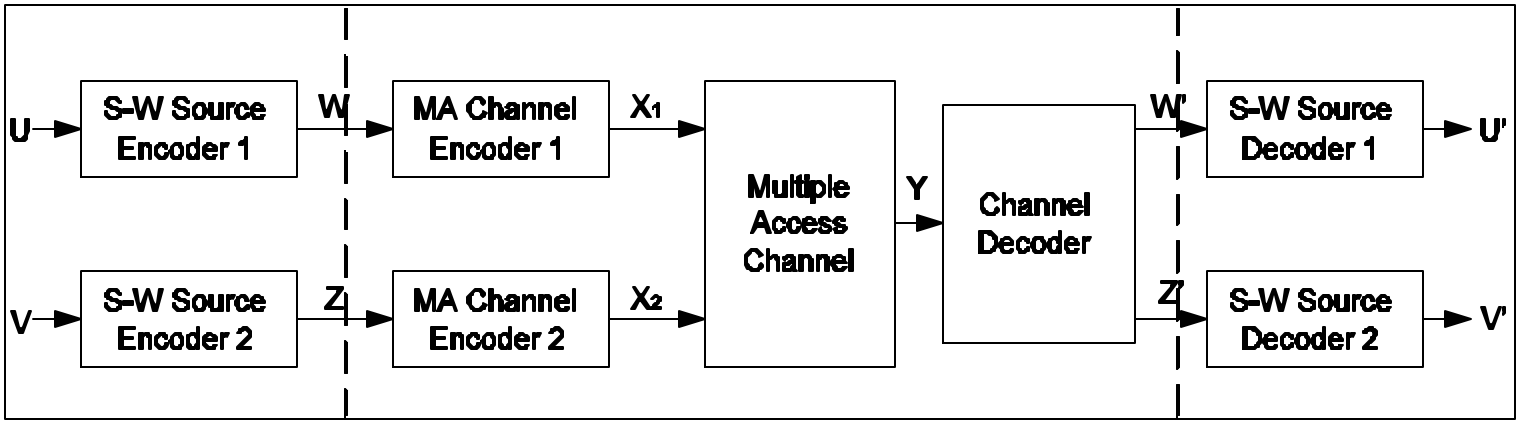
Coding Theorem - Multiple-Access Communication



We can transmit the discrete source pair (U, V) with arbitrarily small error probability iff

$$H(U,V) < C_{\text{joint}} = \sup_{P_{X_1X_2}} I(X_1, X_2; Y)$$

What if we do separate source-channel coding?



- No coordination needed between the two transmitters.
- Compression and channel coding are two separate operations.
- Input to channel encoders (W,Z) are independent.
- With this scheme, source can be reliably transmitted iff Slepian Wolf and Mutiple Access Regions overlap i.e.

$$H(U,V) < C_{\text{separate}} = \sup_{P_{X_1} P_{X_2}} I(X_1, X_2; Y)$$

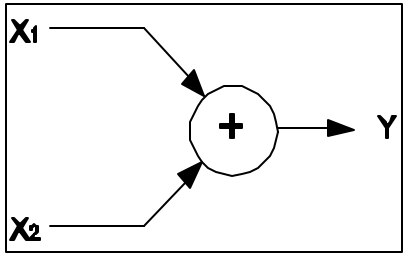
Source-Channel Separation

- Source-channel separation holds when we do not lose optimality by doing the source coding and the channel coding separately.
- This happens when the mutual information between the inputs and output of the channel doesn't increase if we increase the correlation between the channel inputs.
- Mathematically, the necessary and sufficient condition for source-channel separation to hold for a multiple-access channel is:

$$C_{\text{separate}} = C_{\text{joint}}$$

$$\Rightarrow \sup_{P_{X_1} P_{X_2}} I(X_1, X_2; Y) = \sup_{P_{X_1 X_2}} I(X_1, X_2; Y)$$

Example where source-channel separation fails [Cover]



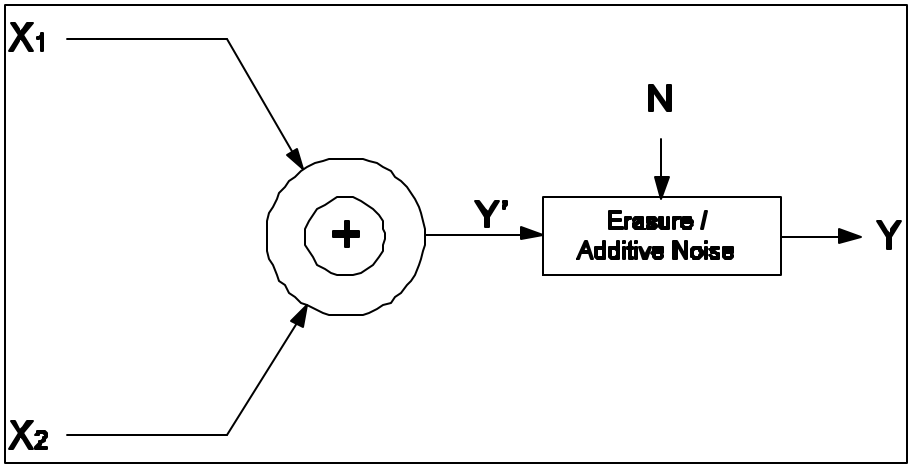
$$\sup_{P_{X_1}P_{X_2}} I(X_1, X_2; Y) = 1.5 \text{ bits}$$

$$\sup_{P_{X_1X_2}} I(X_1, X_2; Y) = \log_2 3 \text{ bits}$$

U \ V	0	1
0	1/3	1/3
1	0	1/3

- Inputs are from {0,1} whereas output is from {0,1,2}.
- Addition is over the reals.
- Source-channel separation fails for this multiple access channel.
- Correlation between the inputs increases the mutual information.
- Example: Source on the left cannot be reliably transmitted if we do separate source and channel coding but can be if it is directly transmitted over the channel.

Multiple Access Channels over Finite Fields



- Inputs X_1 and X_2 are elements of GF(2^k), k=1.
- Addition is over the same field as the inputs.
- Noise can be in the form of erasures or additive.
- Additive noise is also from the same field as the inputs.
- The output Y is an element from the same field but can also take on an additional Erasure symbol when noise is modeled in the form of erasures.

Modeling

- Data packets of length k bits can be considered as elements from GF(2^k). Thus encoding/decoding of packets becomes encoding/decoding of elements from GF(2^k).
- Linear mixing of packets becomes addition of finite field elements.
- We use noise over finite fields to model lossy links.
- When lossy links cause packets to get corrupted and the receiver doesn't know which packets are corrupted, we model the noise as additive and belonging to GF(2^k).
- If the receiver knows which packets are lost/corrupted, we model noise as erasures.
- We therefore have networks operating over finite fields.

Networks over Finite Fields

- Canonical Networks-
  - Point to Point Networks
  - Degraded Broadcast Networks
  - Multiple Access Networks

- Questions-
  - Is linear source coding optimal?
  - Is linear channel coding optimal?
  - Does source-channel separation hold?
  - Are linear joint source-channel codes optimal?

Point to Point Networks

- Source-channel separation holds [Shannon].
- Linear source codes asymptotically achieve the source entropy [Ancheta].
- Linear Channel / Joint source-channel codes achieve Shannon capacity [Elias]

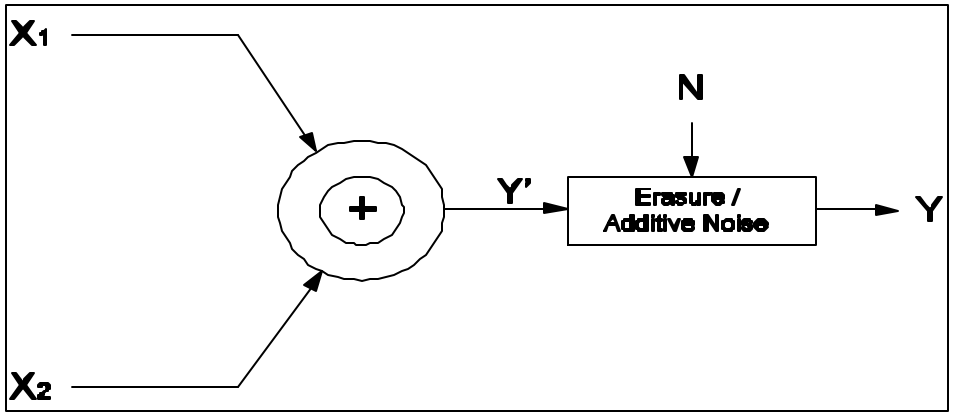
Multiple Access Finite Field Channel (noise independent of inputs)

$$
\begin{aligned}
I(X_1, X_2; Y) &= H(Y) - H(Y \mid X_1, X_2) \\
&= H(Y) - H(N \mid X_1, X_2) \\
&= H(Y) - H(N) \text{ (Ind. Noise)} \\
&= (1-b)H(Y') \\
&= (1-b)H(X_1 \oplus X_2)
\end{aligned}
$$

Erasure Channel

$$
\begin{aligned}
I(X_1, X_2; Y) &= H(Y) - H(Y \mid X_1, X_2) \\
&= H(Y) - H(N \mid X_1, X_2) \\
&= H(Y) - H(N) \text{ (Ind. Noise)} \\
&= H(X_1 \oplus X_2 \oplus N) - H(N)
\end{aligned}
$$

Additive Noise Channel



- ß is the probability of an erasure
- Additions are over a finite field.

When source-channel separation holds

Necessary and sufficient condition shown earlier

$$\sup_{P_{X_1}P_{X_2}} I(X_1, X_2; Y) = \sup_{P_{X_1X_2}} I(X_1, X_2; Y)$$

Source-channel separation holds for erasure finite field channels iff

$$\sup_{P_{X_1}P_{X_2}} H(X_1 \oplus X_2) = \sup_{P_{X_1X_2}} H(X_1 \oplus X_2)$$

Source-channel separation holds for additive noise finite field channel iff

$$\sup_{P_{X_1}P_{X_2}} H(X_1 \oplus X_2 \oplus N) = \sup_{P_{X_1X_2}} H(X_1 \oplus X_2 \oplus N)$$

When source-channel separation holds

- Addition of independent finite field random elements over the same field corresponds to circular convolution of their pmfs.
- Thus, even if one of the elements being added is independent of the others and is uniformly distributed, the sum has a uniform pmf.
- If X_1 and X_2 are independent and uniformly distributed random variables from GF(2^k):

$$H(X_1 \oplus X_2) = k$$

$$H(X_1 \oplus X_2 \oplus N) = k$$

- For a discrete random variable from a finite field of size 2^k, maximum value that that the entropy function can take is k. Thus:

$$\text{Erasure Channel: } \sup_{P_{X_1}P_{X_2}} I(X_1, X_2; Y) = \sup_{P_{X_1X_2}} I(X_1, X_2; Y) = (1-b)k$$

$$\text{Additive Noise Channel: } \sup_{P_{X_1}P_{X_2}} I(X_1, X_2; Y) = \sup_{P_{X_1X_2}} I(X_1, X_2; Y) = k - H(N)$$

Source-channel separation holds for the finite field erasure and additive noise channels when noise does not depend on inputs.

Separation when additive noise is input dependent

- We consider channels over a field size of 2 (binary).
- Source-channel separation may not hold.
- Maximum loss due to separation failure is ½ bit per channel use.
- The probability that separation fails for a channel chosen uniformly and randomly from the ensemble of all channels of this class is bounded by 1/3.

Example of source-channel separation not holding

Consider the channel: $Y = X_1 + X_2 + N$.

$\Pr(N = 1 \mid X_1 = 0, X_2 = 0) = 1$

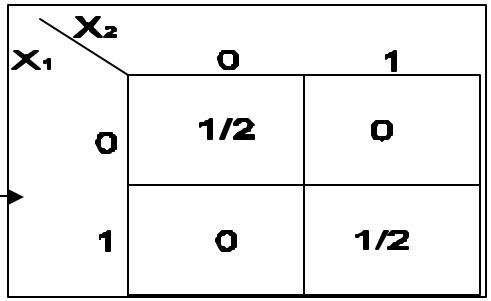
$\Pr(N = 1 \mid X_1 = 0, X_2 = 1) = 0.5$

$\Pr(N = 1 \mid X_1 = 1, X_2 = 0) = 0.5$

$\Pr(N = 1 \mid X_1 = 1, X_2 = 1) = 0$

(The noise is input dependent.)

$$\sup_{P_{X_1X_2}} I(X_1, X_2; Y) = 1$$



$$\sup_{P_{X_1}P_{X_2}} I(X_1, X_2; Y) = 0.5 \quad \longleftrightarrow \quad \Pr(X_1 = 0) = 0.5, \ \Pr(X_2 = 0) = 0.5$$

Source Channel separation fails!

Definitions

For any additive noise finite field channel:

$$\Pr(N=0 \mid X_1=0, X_2=0) = a_{00}$$
$$\Pr(N=1 \mid X_1=0, X_2=1) = a_{01}$$
$$\Pr(N=1 \mid X_1=1, X_2=0) = a_{10}$$
$$\Pr(N=0 \mid X_1=1, X_2=1) = a_{11}$$

- $(a_{00}, a_{01}, a_{10}, a_{11})$ characterizes a particular channel.
- Denote $G(a_{00}, a_{01}, a_{10}, a_{11})$ as a lower bound on the loss in capacity due to separation failure for a channel (note: not per source)

$$G(a_{00}, a_{01}, a_{10}, a_{11}) = \sup_{P_{X_1X_2}} I_a(X_1, X_2; Y) - \sup_{P_{X_1}P_{X_2}} I_a(X_1, X_2; Y)$$

  - $0 = G(a_{00}, a_{01}, a_{10}, a_{11}) = 1$
  - $G(a_{00}, a_{01}, a_{10}, a_{11})$ is not a convex/concave function (hard to analyze).
  - Maximum loss due to separation failure:

$$\max_{a_{00}, a_{01}, a_{10}, a_{11}} G(a_{00}, a_{01}, a_{10}, a_{11}) = \frac{1}{2}$$

Outline of Proof:

Define :

$$R_{JC}(a_{00}, a_{01}, a_{10}, a_{11}) = \sup_{P_{X_1 X_2}} I_a(X_1, X_2; Y)$$

$$R_{SC}(a_{00}, a_{01}, a_{10}, a_{11}) = \sup_{P_{X_1} P_{X_2}} I_a(X_1, X_2; Y)$$

- $R_{JC}(\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11}) = R_{JC}(\alpha_{00}, 0.5, 0.5, \alpha_{11})$
- $R_{SC}(\alpha_{00}, \alpha_{01}, \alpha_{10}\,\alpha) = R_{SC}(\alpha_{00}, 0.5, 0.5, \alpha_{11})$
- Thus, $G(\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11}) = G(\alpha_{00}, 0.5, 0.5, \alpha_{11}) = G(\alpha, ß)$ $\alpha, ß$ in $[0,1]$

$$\Rightarrow \max_{a_{00}, a_{01}, a_{10}, a_{11}} G(a_{00}, a_{01}, a_{10}, a_{11}) \le \max_{a, b} G(a, b)$$

For $a, b \in (0,1)$

$$\frac{\partial G(a,b)}{\partial a} \neq 0 \quad \frac{\partial G(a,b)}{\partial b} \neq 0$$

$$a = 0,\ b \in [0,1] \ : \frac{\partial G(0,b)}{\partial b} > 0$$

$$b = 1,\ a \in [0,1] \ : \frac{\partial G(0,b)}{\partial a} < 0$$

$$a = 1,\ b \in [0,1] \ : \frac{\partial G(0,b)}{\partial b} < 0$$

$$b = 0,\ a \in [0,1] \ : \frac{\partial G(0,b)}{\partial a} > 0$$

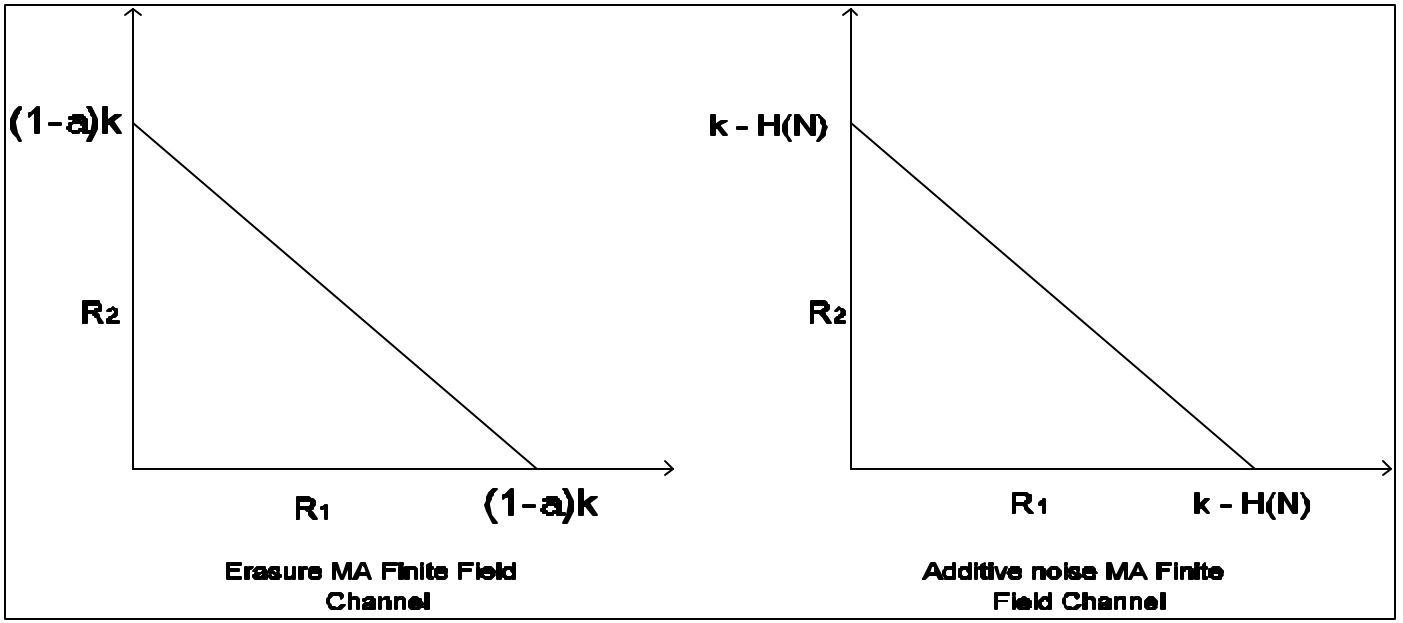
Maximum loss in capacity due to separation failure

$$\max_{a,b} G(a,b) = G(0,1) = G(1,0) = \frac{1}{2}$$

$$\Rightarrow \max_{a_{00},a_{01},a_{10},a_{11}} G(a_{00},a_{01},a_{10},a_{11}) \leq \frac{1}{2}$$

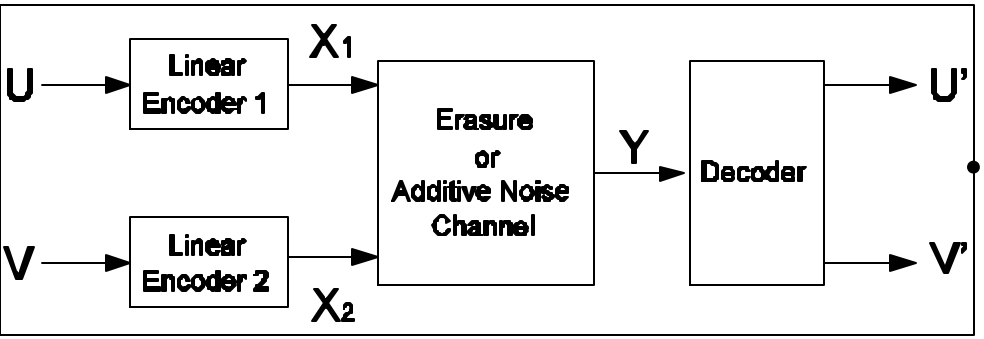
- But the example we considered earlier had a loss due to separation failure as ½ bit per channel use.
- Hence, the maximum loss in capacity due to separation failing for a additive noise multiple access finite field channel with input dependent noise is ½ bit per channel use.

Optimality of Linear Multiple Access Channel Codes



- The multiple access capacity is equal to the corresponding single-transmitter, single-receiver capacity.
- Time sharing achieves all points in the capacity region.
- We know that linear channel codes achieve all rates till capacity for the single-tansmitter, single-receiver network.
- But, time sharing between linear codes can itself be described as a linear multiple access channel code.
- Hence, linear multiple access channel codes achieve achieve all points in the capacity region and are optimal.

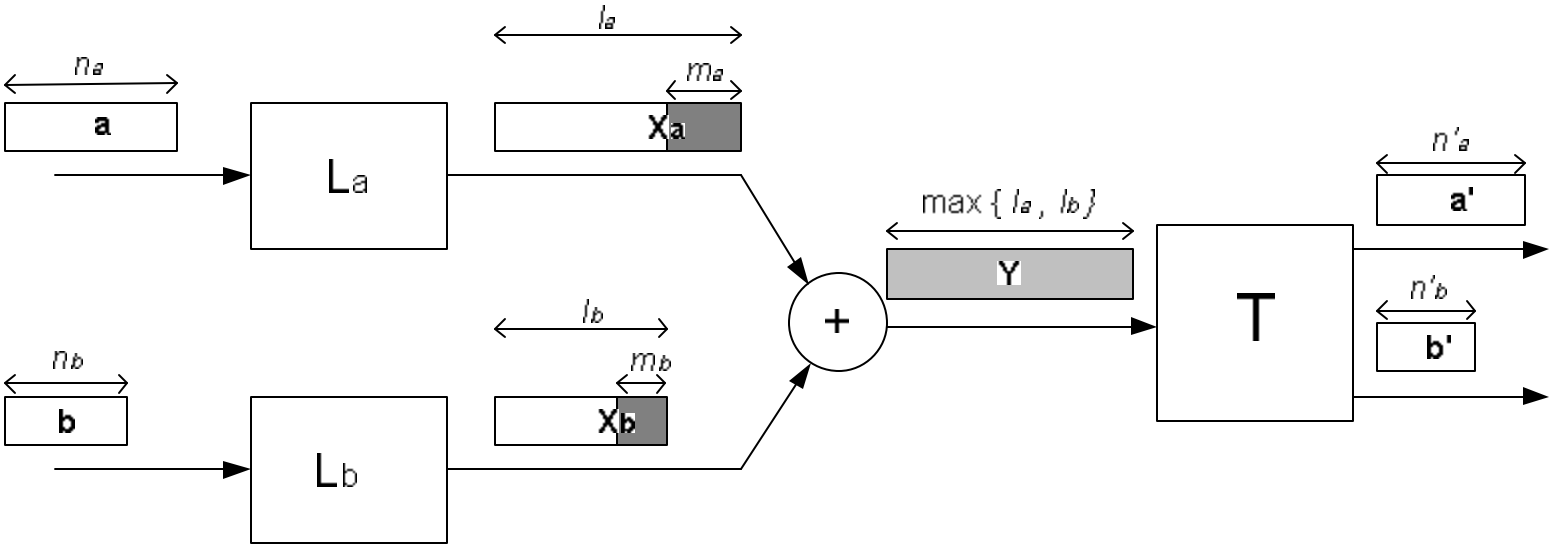
Optimality of joint source-channel codes for the multiple access channel (erasure/additive noise)

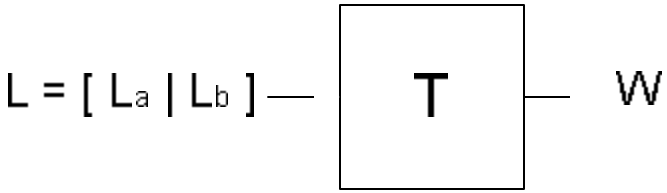


The linear joint encoders at the two transmitters do not need to coordinate

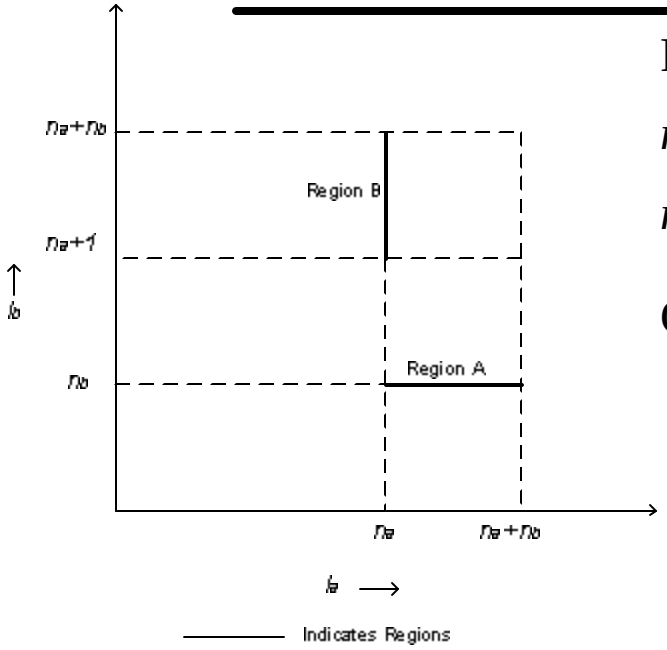
- We know that using linear source codes, we can compress the sources upto H(U,V), i.e. $H(U,V) < R_1 + R_2$.
- Linear multiple access channel codes can be used to achieve all sum rates till capacity, i.e. $R_1 + R_2 < C_{separate}$.
- By combining the linear multiple access source and channel codes, we get a linear joint source-channel code that can transmit any source pair where $H(U,V) < C_{separate}$.
- Any source pair (U,V) can be transmitted across a multiple access channel iff $H(U,V) < C_{joint}$.
- But we have shown that source-channel separation hold , i.e $C_{joint}=C_{separate}$.
- Thus linear joint source-channel codes are optimal.

Code Construction





Construction and Performance



Region A

$$n_a^{'} = m_a + n_a - n_b$$

$$n_b^{'} = m_a$$

$$\text{Code Rate} = \frac{2m_a + n_a - n_b}{n_a + n_b + m_a}$$

Region B

$$n_a^{'} = m_b$$

$$n_b^{'} = m_b - n_a + n_b$$

$$\text{Code Rate} = \frac{2m_b - n_a + n_b}{n_a + n_b + m_b}$$

Performance:

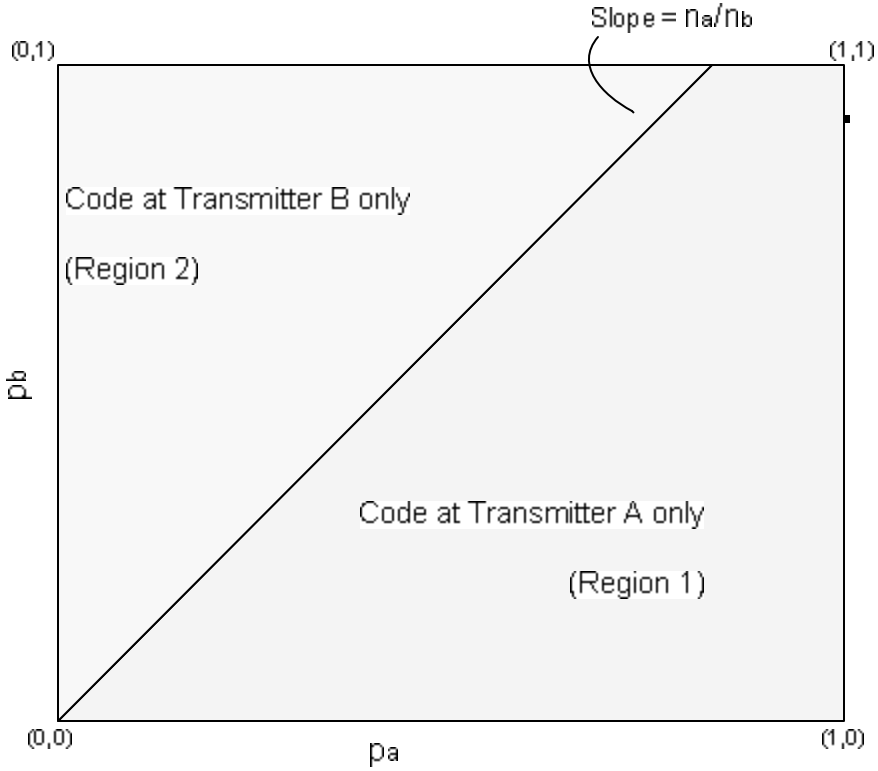
$$\text{Highest Code Rate} = \frac{n_a + n_b}{n_a + 2n_b} \qquad R_{sum} = k$$

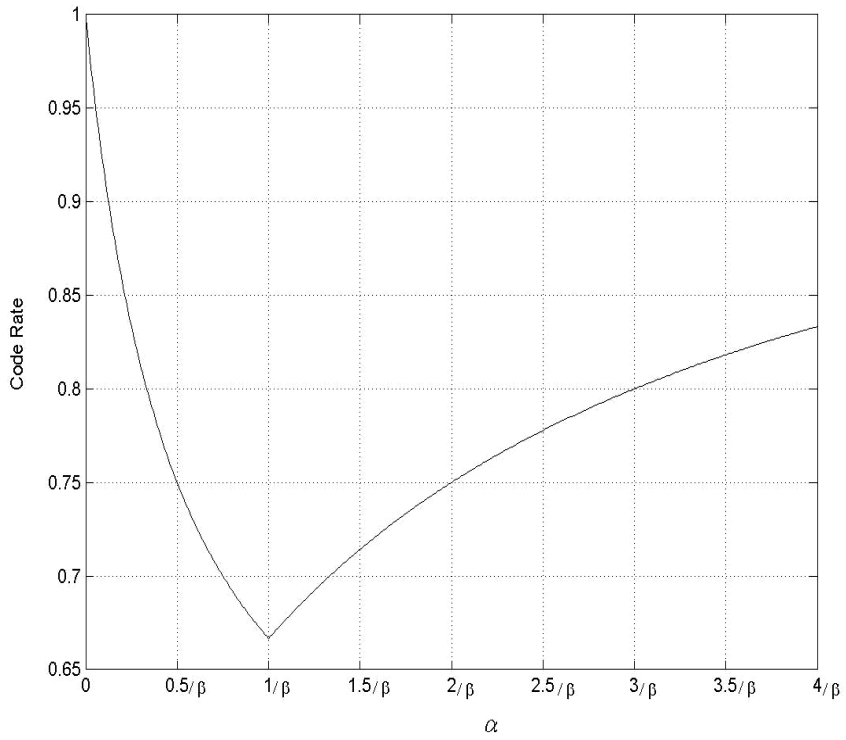
$$\Rightarrow \text{Code is an optimal code}$$

- It is sufficient to add redundancy at only one node.
- If the transmitter is allowed to code over the entire transmit vector, coding on the larger transmit vector gives the highest code rate.
- Code has the property of an optimal code - Achieves the maximum code rate and capacity and adds no redundancy to the smaller transmit vector.

When nodes become bursty

- The transmissions of nodes become bursty when they do not always have packets to transmit.
- We will model the transmissions as being a Bernoulli random process.
- Codewords now have an expected length and we look at the maximum expected code rate.





- $a = p_a/p_b$, $ß = n_a/n_b$
- For bursty transmissions where information codewords have the same length, redundancy should be added at the less bursty transmitter

Network coding for security and robustness

Outline

Network coding for detecting attacks

Network management requirements for robustness

Centralized versus distributed network management

Byzantine security

Robustness against faulty/malicious components with arbitrary behavior, e.g.

- dropping packets
- misdirecting packets
- sending spurious information

Abstraction as Byzantine generals problem [LSP82]

Byzantine robustness in networking [P88,MR97,KMM98,CL99]

Byzantine detection with network coding [HLKMEK04]

Distributed randomized network coding can be extended to detect Byzantine behavior

Small computational and communication overhead

– small number of hash bits included with each packet, calculated as simple polynomial function of data

Require only that a Byzantine attacker does not design and supply modified packets with complete knowledge of other nodes' packets

Byzantine modification detection scheme

Suppose each packet contains θ data symbols $x_1, \ldots, x_\theta$ and ϕ θ hash symbols $y_1, \ldots, y_\phi$

Consider the function $\pi(x_1, \ldots, x_k) = x_1^2 +$ $+ x_k^{k+1}$

Set

$$\begin{aligned} y_i &= \pi(x_{(i-1)k+1}, \ldots, x_{ik}) \quad \text{for} \quad i = 1, \ldots, \phi \quad 1 \\ y_\phi &= \pi(x_{(\phi-1)k+1}, \ldots, x_\theta) \end{aligned}$$

where $k = \left\lceil \frac{\theta}{\phi} \right\rceil$ is a design parameter trading off overhead against detection probability

Detection probability

[HLKMEK04] If the receiver gets s genuine packets, then the detection probability is at least 1 $\left(\frac{k+1}{q}\right)^s$.

E.g. With 2% overhead ($k = 50$), code length=7, $s = 5$, the detection probability is 98.9%.

with 1% overhead ($k = 100$), code length=8, $s = 5$, the detection probability is 99.0%.

Analysis

Let M be the matrix whose i^{th} row $\underline{m_i}$ represents the concatenation of the data and corresponding hash value for packet i

Suppose the receiver tries to decode using

- s unmodified packets, represented as $C_a\,[M|I]$, where the i^{th} row of the coefficient matrix C_a is the vector of code coefficients of the i^{th} packet

- $r \quad s$ modified packets, represented by $[C_b M + V|C_b]$, where V is an arbitrary matrix

Analysis (cont'd)

Let $C = \left[\frac{C_a}{C_b}\right]$

Decoding is equivalent to pre-multiplying the matrix

$$\left[\begin{array}{c|c} C_aM & C_a \\ \hline C_bM + V & C_b \end{array}\right]$$

with C^{-1}, which gives

$$\left[\begin{array}{c|c} M + C^{-1}\left[\frac{0}{V}\right] & I \end{array}\right]$$

For any C_b and V, since receiver decodes only with a full rank set of packets, possible values of C_a are s.t. C is non-singular

Analysis (cont'd)

We can show that

for each of s packets, the attacker knows only that the decoded value will be one of $q^{\text{rank}(V)}$ possibilities

$$\left\{ \underline{m}_i + \sum_{j=1}^{\text{rank}(V)} \gamma_{i,j}\underline{v}_j \;\middle|\; \gamma_{i,j} \in \mathbb{F}_q \right\}$$

at most $k+1$ out of the q vectors in a set $\{\underline{u} + \gamma\underline{v} | \gamma \in \mathbb{F}_q\}$, where $\underline{u} = (u_1, \ldots, u_{k+1})$ is a fixed length-$(k+1)$ vector and $\underline{v} = (v_1, \ldots, v_{k+1})$ a fixed nonzero length-$(k+1)$ vector, can satisfy the property that the last element of the vector equals the hash of the first k elements.

Network mgt for link failure recovery [HMK02, HMK03]

Structured schemes for link failure recovery, e.g. end-to-end path protection, loopback, generalized loopback

Network coding admits any solution feasible on surviving links

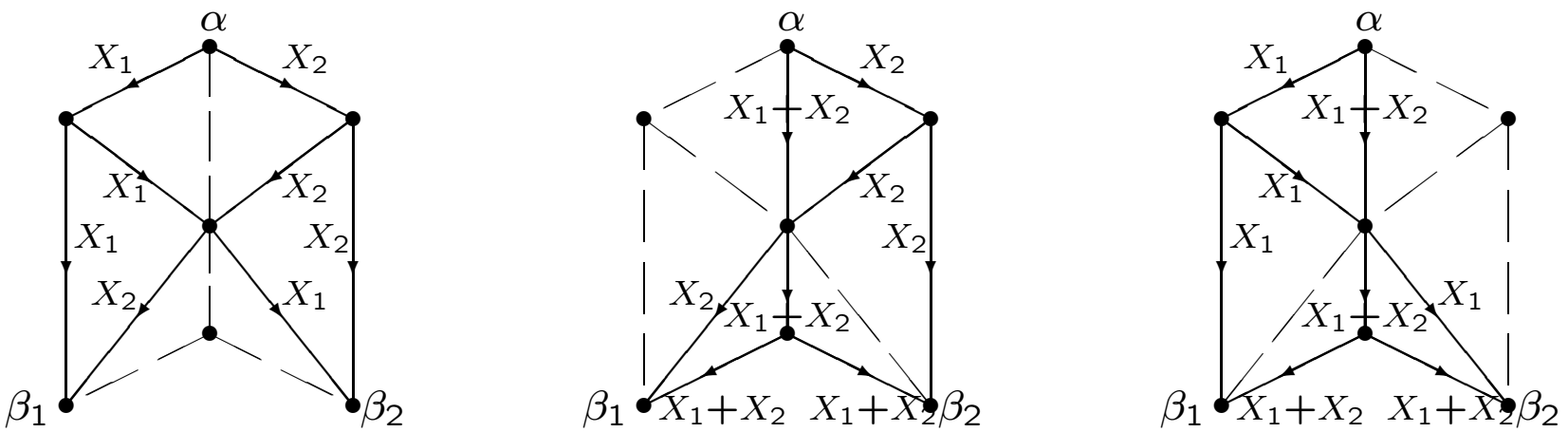
Network management information directs network's response to different link failures

Questions:
-How to quantify fundamental amount of information needed

to direct recovery?
-How do different types of recovery schemes compare in management overhead?

A theoretical framework for network management

Network management information can be quantified by the log of the number of different behaviors (codes) used [tbh]



Allowing general network coding solutions gives fundamental limits on management information required

Classes of failure recovery schemes considered

Receiver-based schemes: only receivers change behavior under different failure scenarios

Network-wide schemes: any node may change behavior, includes receiver based schemes as a special case

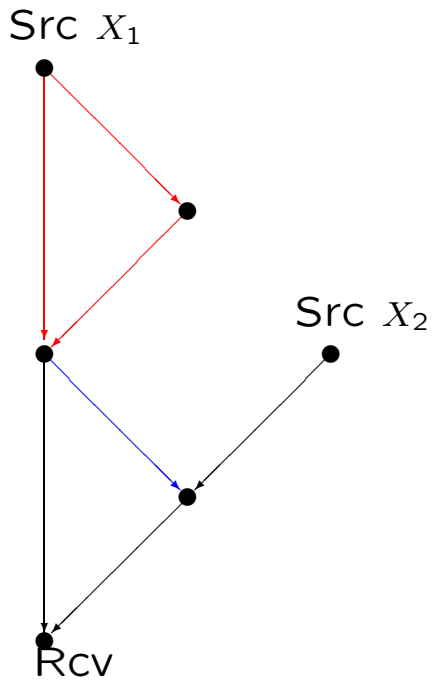
Linear schemes: linear operations at all nodes

Nonlinear receiver-based schemes: nonlinear decoding at receivers

Need for network management

A link h is called *integral* if there exists some subgraph of the network on which the set of source-receiver connections is feasible if and only if h has not failed.

For any network connection problem with at least one integral link whose failure is recoverable, no single linear code can cover the no-failure scenario and all recoverable failures


Src X_1
Src X_2
Rcv

Bounds on network management

Network management for single recoverable link, using network parameters

r, number of source processes transmitted in network;

m, the number of links in a minimum cut between the source nodes and receiver nodes;

d, the number of receiver nodes;

t_{min}, the minimum number of terminal links among all receivers.

Some bounds

Tight lower bounds on no. of linear codes for general case:

receiver-based	$\left\lceil \frac{m}{m-r} \right\rceil$
network-wide	$\left\lceil \frac{m+1}{m-r+1} \right\rceil$

- Tight upper bounds on no. of linear codes for the single-receiver:

receiver-based	$\begin{cases} r+1 & \text{for } r = 1 \text{ or } m-1 \\ r & \text{for } 2 \le r \le m-2 \end{cases}$
network-wide	$\begin{cases} r+1 & \text{for } r = 1,\ r = 2 = m-1 \\ r & \text{for } r = 2 \le m-2, \\ & \quad r = 3,\ r = m-1 \ge 3 \\ r-1 & \text{for } 4 \le r \le m-2 \end{cases}$

- Upper bound on no. of linear codes for multicast: $(r^2+2)(r+1)^{d-2}$

- Tight lower bounds for nonlinear receiver-based codes for multicast:

$$\begin{cases} r & \text{for } 1 < r = t_{\min} - 1 \\ 1 & \text{for } r = 1 \text{ or } r \leq t_{\min} - 2 \end{cases}$$