Overflow Management with Multipart Packets

Yishay Mansour
School of Computer Science
Tel Aviv University
Tel Aviv 69978, Israel
mansour@cs.tau.ac.il

Boaz Patt-Shamir
School of Electrical Engineering
Tel Aviv University
Tel Aviv 69978, Israel
boaz@eng.tau.ac.il

Dror Rawitz
School of Electrical Engineering
Tel Aviv University
Tel Aviv 69978, Israel
rawitz@eng.tau.ac.il

***Abstract*—We study an abstract setting, where the basic information units (called "superpackets") do not fit into a single packet, and are therefore spread over multiple packets. We assume that a superpacket is useful only if the number of its delivered packets is above a certain threshold. Our focus of attention is communication link ingresses, where large arrival bursts result in dropped packets. The algorithmic question we address is which packets to drop so as to maximize goodput. Specifically, suppose that each superpacket consists of k packets, and that a superpacket can be reconstructed if at most $\beta \cdot k$ of its packets are lost, for some given parameter $0 \leq \beta < 1$. We present a simple online distributed randomized algorithm in this model, and prove that in any scenario, its expected goodput is at least $O(\text{OPT}/(k\sqrt{(1-\beta)\sigma}))$, where OPT denotes the best possible goodput by any algorithm, and σ denotes the size of the largest burst (the bound can be improved as a function of burst-size variability). We also analyze the effect of buffers on goodput under the assumption of fixed burst size, and show that in this case, when the buffer is not too small, our algorithm can attain, with high probability, $(1-\epsilon)$ goodput utilization for any $\epsilon > 0$. Finally, we present some simulation results that demonstrate that the behavior of our algorithm in practice is far better than our worst-case analytical bounds.**

I. Introduction

Context and Goal. The following basic situation occurs on many levels in communication systems. There is a data unit that we want to send, but it's too big for the available communication primitive. It is therefore broken into a number of packets, which are sent separately. Since communication may be unreliable, some packets may not arrive at the receiver. However, the data unit is useful at the receiver only if all its parts are delivered.

There are many variants of the basic problems, and a few solution approaches. For example, in the transport layer it is common to use automatic repeat request (ARQ); in the application layer, forward error correction (FEC) is sometimes used. In this paper we concentrate on the network layer. As a concrete example, the reader may think about an MPEG video stream transmitted over the Internet in UDP (see, e.g., [1]). In this case, the basic data unit is a "frame slice," whose size may be larger than the MTU (maximal transfer unit) of the end-to-end connection, and thus it is common that each slice is broken into a few UDP packets. If any of its packets is lost, the slice becomes completely worthless. In the network layer, losses are primarily due to buffer overflows. This motivates our main basic research question in this paper, which can be informally described as follows. Suppose that the basic data units (which we call "superpackets") consist of some $k \geq 1$ packets. Consider link ingresses. When the arrival rate of packets exceeds the link capacity, which packets should be dropped so as to maximize the number of completed superpackets?

Research supported in part by the Next Generation Video (NeGeV) Consortium, Israel (www.negev-initiative.org). The first authors is supported in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886, by a grant from the Israel Science Foundation (grant No. 709/09) and grant No. 2008-321 from the United States-Israel Binational Science Foundation (BSF) and by the Israel Ministry of Science and Technology. This publication reflects the authors' views only. The second author is supported in part by Israel Science Foundation (grant 1372/09) and by the Israel Ministry of Science and Technology.

The basic idea and overview of results. We study this problem mainly from the theoretical perspective. Our basic model is the following (see Section II for more details). Each superpacket consists of k packets. Time is slotted, and in each time step t, an arbitrary set of $\sigma(t)$ packets, called the *burst* of step t, arrives at a server (the server models a link). The server can serve at most c packets at a step (c is the link speed). The task of the scheduling algorithm is to select, in an online fashion, which packets to serve and which to drop, due to buffer space constraints. A superpacket is said to be *completed* if at least $(1-\beta)k$ is its constituent packets are served, for some given redundancy parameter $0 \leq \beta < 1$. The goal of the scheduling algorithm is to maximize the number of completed packets.

Possibly the simplest algorithm to this problem is dropping packets at random. However, a second thought shows that this is not a good strategy: The probability that all k packets of a superpacket are delivered, under random discards, is exponential in k. We therefore consider the following slightly more structured randomization rule:

Algorithm PRIORITY:
- Each *superpacket* is independently assigned a priority uniformly at random from $[0, 1]$.
- Whenever an overflow occurs, the packets whose superpackets have the smallest priority are retained, and the others are discarded.

Aside from its obvious simplicity, PRIORITY has some other important attractive features. From the implementation point of view, note that it is straightforward to implement in a distributed system, by using pseudo-random hash functions,

mapping superpacket IDs to priorities (we only need to ensure that all algorithm sites use the same hash function and the same random seed). From the theoretical viewpoint, this rule (which is inspired by Luby's maximal independent set algorithm [2]) was showed to be effective in the case of online set packing [3], i.e., without buffers or redundancy.

In this paper we extend the results of [3] as follows. First, we consider redundancy: we show that if superpackets are encoded so that they can be recovered even if a fraction β of their constituent packets are lost, then the PRIORITY algorithm is very effective (in [3] $\beta = 0$). Second, we analyze systems with buffers, and derive a bound on the minimum size buffer that guarantees, for any given $\epsilon > 0$, $(1-\epsilon)(1-\beta)$ goodput utilization, when bursts are of roughly the same size. We also consider the case where burst sizes may vary. The idea is that by adding a little redundancy to the superpackets, we may overcome the variability of the burst sizes.

Our results are stated mostly in terms of competitive analysis, comparing the goodput of the algorithm in any given scenario to the best possible goodput of *any* schedule. Specifically, for the case where no buffer space is provided, namely the case in which an arriving packet arrives is either served or dropped, we show that PRIORITY guarantees expected goodput of $\Omega(|\text{OPT}|/(k\sqrt{\sigma_{\max}(1-\beta)}))$, where OPT is the best possible schedule and $\sigma_{\max}$ denotes the maximum burst size. In fact, it turns out that the competitive ratio (the worst-case ratio between the expected performance of the algorithm and OPT), depends on the variability of the burst sizes. For example, if all bursts have the same size, then the competitive ratio improves to $O(k)$. (We note that even offline, it is NP-hard to obtain an $O(k/\log k)$-approximation for the case of $\beta = 0$.) For the case where buffers are provided, and assuming that the burst size is fixed, we show that for any given $\epsilon > 0$, if the buffer size is $\Omega(\epsilon^{-2} k \log(k/\epsilon))$, then the algorithm guarantees expected goodput of $(1-\epsilon)^3|\text{OPT}|$. We also define a notion of "relatively large burst" and provide an upper bound on the competitive ratio of PRIORITY for the case where such bursts are uncommon. Our theoretical study is augmented by some simulation results.

Related Work. Overflow management was studied quite extensively in the last decade from the competitive analysis viewpoint (starting with [4], [5]: see [6] for a recent survey). However, only relatively little is known about the superpacket model we study. The model was first introduced in the competitive analysis context in [7]. In that work, no redundancy is considered, and burst size may be unbounded. They show that in this case, no finite upper bound on the competitive ratio exists; they proceeded to consider a certain restriction on packet arrival order.[1]. Under this assumption, [7] prove an upper bound of $O(k^2)$ and a lower bound of $\Omega(k)$ on the competitive ratio for deterministic online algorithms. A different, possibly more realistic, ordering restriction is studied in [8].[2] In this model, [8] gives an exponential (in k) upper bound on the deterministic competitive ratio, and a linear lower bound. They also give simulation results in which they compared their algorithm to various versions of tail-drop. Both [7] and [8] assume a push-out FIFO buffer architecture.

A different angle was taken in [3], where instead of restricting the packet ordering in the input, the results are expressed in terms of the maximal burst size. In [3], this model is studied under the simplifying assumption that no buffers are available (and again, without redundancy). For this model, PRIORITY is introduced, and shown to have optimal competitive ratio of $O(k\sqrt{\sigma_{\max}})$.

If $\beta = 0$ and $b = 0$, the offline version of the problem reduces to Set Packing, where each superpacket corresponds to a set and each burst correspond to an element. Set Packing is as hard as Maximum Independent Set even when all elements are contained in at most two sets (i.e., $\sigma(t) \leq 2$, for every t), and therefore cannot be approximated to within $O(n^{1-\epsilon})$-factor, for any $\epsilon > 0$ [9]. In terms of the number of elements (time steps), Set Packing is $O(\sqrt{T})$-approximable, and hard to approximate within $T^{1/2-\epsilon}$ [10]. When set sizes is at most k, it is approximable within $k/2 + \epsilon$, for any $\epsilon > 0$ [11] and within $(k+1)/2$ in the weighted case [12], but known to be hard to approximate to within $O(k/\log k)$-factor [13].

Paper organization. We formally define our model in Section II. In Section III we present upper bounds on the competitive ratio of Algorithm PRIORITY for the case where $\beta > 0$ and $c = 0$. Our results for the case of large buffers are given in Section IV, and we consider the case where most bursts are not large in Section V. Section VI contains our simulation results.

II. THE MODEL

In this section we formalize the model we study.

Our basic concept is a *superpacket*, typically denoted S. Each superpacket consists of k unit size *packets*. The input to the system is a sequence of packets that arrive online. Each packet is associated with a superpacket, where a super-packet S_i comprises the packets denoted $p_i^1, p_i^2, \ldots, p_i^k$. The set of superpackets is denoted by $\mathcal{I}$ ($\mathcal{I}$ is unknown in advance). The system progresses in discrete time steps, where in each step an arbitrary set of packets arrive. The time horizon is denoted by T. The arrival time of a packet p is denoted by $\text{arr}(p)$. In each step t, a set of $\sigma(t)$ packets arrive, corresponding to a set $\mathcal{I}(t)$ of superpackets, i.e. $\mathcal{I}(t) = \{S \in \mathcal{I} : p \in S \text{ and } \text{arr}(p) = t\}$. We denote $\overline{\sigma} = \frac{1}{T}\sum_t \sigma(t)$, $\overline{\sigma^2} = \frac{1}{T}\sum_t \sigma^2(t)$ and $\sigma_{\max} = \max_t \sigma(t)$.

The packets arrive at a buffer denoted by B. The buffer can contain at most b packets, and has an output link that can transmit c packets per step. Specifically, an execution of an online algorithm ALG proceeds as follows. Initially, the buffer is empty. Each step consists of three substeps.

[1]The condition is that for any $1 \leq i, j \leq k$, the ith packet of superpacket S arrives before the ith packet of superpacket S' if and only if the jth packet of S arrives before the jth packet of S'.

[2]The assumption there is that there are some M sources, each generating a stream of packets sequentially, one superpacket after another. The link sees an arbitrary interleaving of these streams.

The first substep is the *arrival substep*, where a set of $\sigma(t)$ packets arrives at the system. In the *drop substep* some packets may be dropped at the discretion of the buffer management algorithm. More specifically, a buffer management algorithm may discard packets that arrived earlier and are currently in the buffer or packets that arrived in this step. The third substep is the *delivery substep*: at most c packets are transmitted on the link. A feasible system satisfies the following capacity constraint: the maximum number of packets in the buffer between consecutive time steps (i.e., after the delivery substep) must not exceed the given buffer size b. A buffer management algorithm may drop packets even if there is space available at the buffer. In the special case where $b = 0$ the algorithm acts as an admission control algorithm: at most c packets are delivered and the rest are dropped. A schedule produced by a buffer management algorithm is an assignment of packets to time steps, such that a packet assigned to time t, is transmitted (or equivalently, delivered) in time t. Given an arrival sequence, we sometimes identify an execution of an algorithm with the schedule of its transmitted packets.

The input also contains a *redundancy* factor β. The goal of the buffer management algorithm is to maximize the number of successfully delivered superpackets, where a superpacket is considered successfully delivered if at least $(1-\beta)k$ out of its k constituent packets are delivered. (Henceforth, we assume that $(1-\beta)k$ is integral.) In the special case where $\beta = 0$ a super-packet is considered successfully delivered if all of its constituent packets are delivered.

Given an algorithm ALG, we denote the set of completed superpackets by $\text{ALG}(\mathcal{I})$ (or simply by ALG). If the algorithm is randomized, the benefit for a given instance is a random variable, and we shall use its expected value. We measure the performance of algorithms using competitive analysis: The *competitive ratio* of an algorithm is the supremum, over all instances $\mathcal{I}$, of $|\text{OPT}(\mathcal{I})|/|\text{ALG}(\mathcal{I})|$, where $\text{OPT}(\mathcal{I})$ denotes the schedule with maximum number of super-packets that can be delivered.

In this paper we concentrate on the algorithm whose pseudocode is presented below.

Algorithm 1 : PRIORITY

1: For each superpacket $S \in \mathcal{I}$, pick a random priority $r(S)$ according to the uniform distribution in the range $[0,1]$, namely $r(S) \sim U[0,1]$.
2: **for all** time step t **do**
3: Receive the packets corresponding to $\mathcal{I}(t)$
4: Keep the $b+c$ packets with smallest priority from $\mathcal{B} \cup \mathcal{I}(t)$. All other packets are dropped.
5: Deliver the first c packets (according to FIFO order), and keep the remaining packets in $\mathcal{B}$. (The buffer $\mathcal{B}$ has at most b packets.)
6: **end for**

It may also make sense to consider a weighted version of the problem, where each superpacket S has a weight $w(S)$, and the goal is to maximize the total weight of delivered superpackets (as opposed to their total number in the unweighted version). This seems reasonable in an MPEG video stream, for example, where a slice of an I-frame is more important than a slice of a B-frame.

We note that Algorithm PRIORITY extends to the weighted case by changing step 1: just let $r(S)$ be the maximum of $w(S)$ uniformly distributed random numbers in the range $[0,1]$. We omit further details due to lack of space.

III. ADMISSION CONTROL WITH REDUNDANCY

In this section we assume that no buffers are available. In some sense, in this case the algorithm acts as an admission control algorithm: at most c packets are delivered and the rest are dropped.

We show that PRIORITY guarantees, for any packet arrival pattern, that its expected number of completed superpackets is $\Omega\left(|\text{OPT}|/\left(k\sqrt{\sigma_{\max}(1-\beta)/c}\right)\right)$, where OPT denotes the schedule with maximum possible number of completed superpackets on that arrival pattern (recall that c is the link capacity). We note that this result extends to the weighted setting (details omitted). We also show how the competitive ratio of PRIORITY depends on the variability of the burst sizes.

This section is organized as follows. First we analyze the case of unit capacity links. This gives the basic ideas of the analysis. We then extend the results links of capacity $c > 1$. Finally, we consider the influence of variable burst size on the competitive ratio.

Additional notation. Before presenting our analyses we need several definitions. We write $S \in \text{PRIORITY}$ to denote that S was successfully delivered by Algorithm PRIORITY. For every superpacket $S \in \mathcal{I}$, we denote $N[S] \stackrel{\text{def}}{=} \{S' \in \mathcal{I} : \exists t \text{ s.t. } S, S' \in \mathcal{I}(t)\}$ and $N(S) \stackrel{\text{def}}{=} N[S] \setminus \{S\}$.

A. Unit Capacity Links

In this section we analyze Algorithm PRIORITY for the case where $c = 1$. We first compute a lower bound on the survival probability of a superpacket S.

Lemma 1. *Let $S \in \mathcal{I}$. Then for any $S_{1-\beta} \subseteq S$ containing at least $(1-\beta)k$ packets we have*

$$\Pr[S \in \text{PRIORITY}] \geq \frac{1}{|N[S_{1-\beta}]|} .$$

Proof: Clearly, a superpacket S is successfully delivered if $r(S) = \min_{S' \in N[S_{1-\beta}]} r(S')$. Hence,

$$\begin{aligned}\Pr[S \in \text{PRIORITY}] &\geq \Pr\left[r(S) = \min_{S' \in N[S_{1-\beta}]} r(S')\right] \\ &= \frac{1}{|N[S_{1-\beta}]|} ,\end{aligned}$$

and we are done. ∎

We use the following technical lemma that is based on the Cauchy-Schwarz Inequality.

Lemma 2. *For any positive reals $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$, it holds that $\sum_i \frac{a_i^2}{b_i} \geq \frac{\left(\sum_i a_i\right)^2}{\sum_i b_i}$.*

Lemma 1 and Lemma 2 imply the following.

Lemma 3. *Let $\mathcal{I}' \subseteq \mathcal{I}$ be a collection of superpackets, and for every $S \in \mathcal{I}'$ let $S_{1-\beta} \subseteq S$ be a subset containing at least $(1-\beta)k$ packets. Then,*

$$\mathbb{E}[|\textsc{Priority}|] \geq \frac{|\mathcal{I}'|^2}{\sum_{S\in\mathcal{I}'} |N[S_{1-\beta}]|} \ .$$

Proof: By linearity of expectation we have

$$\begin{aligned}\mathbb{E}[|\textsc{Priority}|] &= \sum_{S\in\mathcal{I}} \Pr[S \in \textsc{Priority}] \\ &\geq \sum_{S\in\mathcal{I}'} \frac{1}{|N[S_{1-\beta}]|} \\ &\geq \frac{|\mathcal{I}'|^2}{\sum_{S\in\mathcal{I}'} |N[S_{1-\beta}]|} \ ,\end{aligned}$$

where the first inequality is by Lemma 1 and the second is by Lemma 2 with $a_i = 1$ and $b_i = |N[S_{1-\beta}]|$. ∎

We now apply Lemma 3 to two collections of superpackets. First, to the superpackets in an optimal solution.

Lemma 4. $\mathbb{E}[|\textsc{Priority}|] \geq \frac{|\textsc{opt}|^2}{k \cdot |\mathcal{I}|}$.

Proof: For each $S \in \textsc{opt}$ fix $S_{1-\beta}$ to be the subset of S which contains the packets delivered by OPT. Clearly $|S_{1-\beta}| \geq (1-\beta)k$ for every $S \in \textsc{opt}$. By Lemma 3 with $\mathcal{I}' = \textsc{opt}$ we have that

$$\mathbb{E}[|\textsc{Priority}|] \geq \frac{|\textsc{opt}|^2}{\sum_{S\in\textsc{opt}} |N[S_{1-\beta}]|} \ .$$

Now, observe that since for any $S', S'' \in \textsc{opt}$ and any $p'_i \in S'_{1-\beta}$ and $p''_j \in S''_{1-\beta}$ we have $\text{arr}(p'_i) \neq \text{arr}(p''_j)$, each superpacket $S \in \mathcal{I}$ intersects at most k such subsets, and hence $\sum_{S\in\textsc{opt}} |N[S_{1-\beta}]| \leq k \cdot |\mathcal{I}|$. ∎

Next, we apply Lemma 3 with the collection of all superpackets in the instance.

Lemma 5. $\mathbb{E}[|\textsc{Priority}|] \geq \frac{|\mathcal{I}|^2}{(1-\beta)|T| \cdot \overline{\sigma^2}}$.

Proof: Fix a superpacket S. Order the packets of S by increasing burst size, namely, let $S = \{p^1, p^2, \ldots, p^k\}$ and assume, w.l.o.g., that $\sigma(\text{arr}(p^j)) \leq \sigma(\text{arr}(p^{j+1}))$ for $1 \leq j \leq k$. Let $S_{1-\beta}$ contain the first $(1-\beta)k$ packets in such an ordering, for every $S \in \mathcal{I}$. Now by Lemma 3, we have that

$$\mathbb{E}[|\textsc{Priority}|] \geq \frac{|\mathcal{I}|^2}{\sum_{S\in\mathcal{I}} |N[S_{1-\beta}]|} \ .$$

Summing over the superpackets we get

$$\begin{aligned}\sum_{S\in\mathcal{I}} |N[S_{1-\beta}]| &\leq \sum_{S\in\mathcal{I}} \sum_{p\in S_{1-\beta}} \sigma(\text{arr}(p)) \qquad (1)\\ &\leq \sum_{S\in\mathcal{I}} (1-\beta) \sum_{p\in S} \sigma(\text{arr}(p)) \\ &= (1-\beta) \sum_t \sigma(t)^2 \\ &= (1-\beta) T \cdot \overline{\sigma^2} \ ,\end{aligned}$$

and therefore $\mathbb{E}[|\textsc{Priority}|] \geq \frac{|\mathcal{I}|^2}{(1-\beta)T\cdot\overline{\sigma^2}}$, as required. ∎

Theorem 1. *In the case of unit-capacity instances, the competitive ratio of* PRIORITY *is at most $k\sqrt{(1-\beta)\overline{\sigma^2}/\overline{\sigma}}$.*

Proof: Lemma 4 and Lemma 5 give us two lower bounds on $|\textsc{Priority}|$. The maximum of these bounds is minimized when $|\textsc{opt}| = \sqrt{\frac{|\mathcal{I}|^3 \cdot k}{(1-\beta)T\cdot\overline{\sigma^2}}}$, and therefore, for any instance

$$\mathbb{E}[|\textsc{Priority}|] \geq |\textsc{opt}| \cdot \sqrt{\frac{|\mathcal{I}|}{kT(1-\beta)\cdot\overline{\sigma^2}}} \ .$$

Finally, since

$$T \cdot \overline{\sigma} = \sum_t \sigma(t) = \sum_{S\in\mathcal{I}} k = k \cdot |\mathcal{I}| \ , \qquad (2)$$

it follows that

$$\mathbb{E}[|\textsc{Priority}|] \geq |\textsc{opt}| \cdot \sqrt{\frac{\overline{\sigma}}{(1-\beta)k^2\overline{\sigma^2}}} = \frac{|\textsc{opt}|}{k} \cdot \sqrt{\frac{\overline{\sigma}}{(1-\beta)\overline{\sigma^2}}} \ ,$$

and we are done. ∎

Corollary 2. *For any unit capacity instance,*

$$\mathbb{E}[|\textsc{Priority}|] \geq \frac{|\textsc{opt}|}{k\sqrt{(1-\beta)\sigma_{\max}}} \ .$$

Proof: Follows from the fact that

$$\sqrt{\frac{\overline{\sigma}}{\overline{\sigma^2}}} = \sqrt{\frac{\sum_t \sigma(t)/T}{\sum_t \sigma(t)^2/T}} \geq \sqrt{\frac{\sum_t \sigma(t)}{\sigma_{\max}\sum_t \sigma(t)}} = \frac{1}{\sqrt{\sigma_{\max}}} \ .$$

∎

B. Larger Capacity Links

We now consider the case of links whose capacity is larger than 1. As we show, it turns out that larger capacity may increase the best possible goodput by a disproportionate factor, and therefore a more careful analysis is required. However, we show that the competitive factor proved for the unit capacity case essentially holds, losing only a constant factor.

The effect on OPT. We start by pointing out the following interesting phenomenon, which shows that a slight increase in the capacity can dramatically increase the best possible goodput.

Theorem 3. *For any link capacity $c \geq 2$ and for any $n > c$ there exist an input instance $\mathcal{I}$ with n superpackets such that*

$$\frac{|\textsc{opt}_c(\mathcal{I})|}{|\textsc{opt}_{c-1}(\mathcal{I})|} = \frac{n}{c-1} = \frac{\Omega(k^{1/c-1})}{c-1} \ ,$$

where $\textsc{opt}_x$ is the optimal schedule with capacity x.

Proof: We construct the instance $\mathcal{I}$ as follows. Our instance consists of $\binom{n}{c}$ time steps, where for each step t, we choose a distinct subset $S_t \subset \{1,\ldots,n\}$ of superpackets, and let packets of S_t arrive at time t. We take all distinct $\binom{n}{c}$ subsets of size c, one for each time step. This implies that $k = \binom{n-1}{c-1}$. Consider now different capacities of the link: If the capacity is c, then all packets—and hence all superpackets—can be delivered. However, if the capacity is $c-1$ only $c-1$ superpackets can be delivered, because by construction, any c superpackets meet at some step from which at most $c-1$ superpackets can survive. ∎

Analysis of the competitive ratio. As in Section III-A we first bound the probability that a given superpacket S is successfully delivered by the algorithm.

Lemma 6. *For $S \in \mathcal{I}$ and fix any subset $S_{1-\beta}$ of S containing at least $(1-\beta)k$ packets. If $|N[S_{1-\beta}]| \leq c$ then $\Pr[S \in \textsc{Priority}] = 1$, and if $|N[S_{1-\beta}]| > c$ then*

$$\Pr[S \in \textsc{Priority}] \geq \frac{c}{2|N[S_{1-\beta}]|} \ .$$

Proof: Consider a superpacket $S \in \mathcal{I}$. Clearly, S is successfully delivered if all packets from $S_{1-\beta}$ are delivered. Furthermore, $S \in \textsc{Priority}$ if every packet in $S_{1-\beta}$ has one of the top c priorities in $\mathcal{I}(t)$. We shall analyze the probability that even a stronger condition holds, namely that S has one of the top c priorities from the priorities of all superpackets in $N[S_{1-\beta}]$.

Observe that if $|N[S_{1-\beta}]| \leq c$, then $\Pr[S \in \textsc{Priority}] = 1$, as required. Hence, we assume that $N[S_{1-\beta}] > c$. View the selection of the top c superpackets in $N[S_{1-\beta}]$ as a sequential process, where we choose superpackets at random from $N[S_{1-\beta}]$ without repetition. The probability that a superpacket S in one of the top c will only decrease if we allow repetition. Thus, recalling that by the proof of Lemma 1, we have that $\Pr[r(S') = \min_{T \in N[S_{1-\beta}]} r(T)] = \frac{1}{|N[S_{1-\beta}]|}$, for every superpacket $S' \in N[S_{1-\beta}]$, we may conclude that

$$\Pr[S \in \textsc{Priority}] \geq 1 - \left(1 - \frac{1}{|N[S_{1-\beta}]|}\right)^c \ .$$

By the binomial expansion we have

$$\begin{aligned}\Pr[S \in \textsc{Priority}] &\geq 1 - \left(1 - \frac{1}{|N[S_{1-\beta}]|}\right)^c \\ &\geq \frac{c}{|N[S_{1-\beta}]|} - \frac{1}{2}\left(\frac{c}{|N[S_{1-\beta}]|}\right)^2 \\ &\geq \frac{c}{2|N[S_{1-\beta}]|} \ .\end{aligned}$$

and we are done. ∎

Lemma 6 and Lemma 2 imply the following.

Lemma 7. *Let $\mathcal{I}' \subseteq \mathcal{I}$ be a collection of superpackets, and for every $S \in \mathcal{I}'$ let $S_{1-\beta} \subseteq S$ be a subset containing at least $(1-\beta)k$ packets. Then,*

$$\mathbb{E}[|\textsc{Priority}|] \geq \min\left\{\tfrac{|\mathcal{I}'|}{2} \ , \ \tfrac{c|\mathcal{I}'|^2}{8\sum_{S\in\mathcal{I}'}|N[S_{1-\beta}]|}\right\} \ .$$

Proof: Let $\mathcal{I}'' = \{S \in \mathcal{I}' : |N[S_{1-\beta}] > c\}$. If $|\mathcal{I}''| \leq \frac{1}{2}|\mathcal{I}'|$, we are done. Otherwise,

$$\begin{aligned}\mathbb{E}[|\textsc{Priority}|] &\geq \sum_{S\in\mathcal{I}''} \frac{c}{2|N[S_{1-\beta}]|} \\ &\geq \frac{c|\mathcal{I}''|^2}{2\sum_{S\in\mathcal{I}''}|N[S_{1-\beta}]|} \\ &\geq \frac{c|\mathcal{I}'|^2}{8\sum_{S\in\mathcal{I}'}|N[S_{1-\beta}]|} \ ,\end{aligned}$$

where the first inequality is by Lemma 1 and the second is by Lemma 2 with $a_i = 1$ and $b_i = |N[S_{1-\beta}]|$. ∎

The next two lemmas They are analogs of Lemmas 4 and 5.

Lemma 8. $\mathbb{E}[|\textsc{Priority}|] \geq \dfrac{|\textsc{opt}|^2}{8k \cdot |\mathcal{I}|}$.

Proof: For each $S \in \textsc{opt}$ let $S_{1-\beta}$ contain the packets from S that are delivered by OPT. Now, by Lemma 7 with $\mathcal{I}' = \textsc{opt}$ we have that either $\mathbb{E}[|\textsc{Priority}|] \geq \frac{1}{2}|\textsc{opt}|$ and we are done, or

$$\mathbb{E}[|\textsc{Priority}|] \geq \frac{c|\textsc{opt}|^2}{8\sum_{S\in\textsc{opt}}|N[S_{1-\beta}]|} \ .$$

Now, observe that $|\mathcal{I}(t) \cap \{S_{1-\beta} : S \in \textsc{opt}\}| \leq c$, and therefore each superpacket $S \in \mathcal{I}$ intersects at most kc subsets, and hence $\sum_{S\in\textsc{opt}}|N[S_{1-\beta}]| \leq kc \cdot |\mathcal{I}|$. The lemma follows. ∎

Lemma 9. $\mathbb{E}[|\textsc{Priority}|] \geq \min\left\{\frac{|\mathcal{I}|}{2} \ , \ \frac{c|\mathcal{I}|^2}{8T(1-\beta)\cdot\overline{\sigma^2}}\right\}$.

Proof: As in Lemma 5 let $p^1, p^2, \ldots, p^k$ be an ordering of the packets in a superpacket S such that $\sigma(\text{arr}(p^j)) \leq \sigma(\text{arr}(p^{j+1}))$ for every j. Let $S_{1-\beta}$ contain the first $(1-\beta)k$ packets in such an ordering, for every $S \in \mathcal{I}$. By Lemma 7 with $\mathcal{I}' = \textsc{opt}$ we have that either $\mathbb{E}[|\textsc{Priority}|] \geq \frac{1}{2}|\mathcal{I}|$ and we are done, or

$$\mathbb{E}[|\textsc{Priority}|] \geq \frac{c|\mathcal{I}|^2}{8\sum_{S\in\mathcal{I}}|N[S_{1-\beta}]|} \ .$$

By Eq. (1) we have $\sum_{S\in\mathcal{I}}|N[S_{1-\beta}]| = (1-\beta)T\overline{\sigma^2}$. ∎

Theorem 4. *The competitive ratio of Algorithm* PRIORITY *is*

$$O\left(k\sqrt{\frac{(1-\beta)\overline{\sigma^2}}{c\overline{\sigma}}}\right) \ .$$

Proof: First, observe that if $\mathbb{E}[|\textsc{Priority}|] \geq \frac{1}{2}|\mathcal{I}|$, then we are done. Otherwise, Considering the lower bounds provided by Lemmas 8 and 9, we conclude that the larger of the two bounds on $\mathbb{E}(|\textsc{Priority}|)$ is minimized when

$$|\textsc{opt}| = \sqrt{\frac{ck\cdot|\mathcal{I}|^3}{(1-\beta)T\cdot\overline{\sigma^2}}} \ .$$

Hence, we have

$$\begin{aligned}\mathbb{E}[|\textsc{Priority}|] &\geq \frac{|\textsc{opt}|}{8}\sqrt{\frac{c|\mathcal{I}|}{(1-\beta)kT\overline{\sigma^2}}} \\ &= \frac{|\textsc{opt}|}{8}\cdot\sqrt{\frac{c\overline{\sigma}}{(1-\beta)k^2\overline{\sigma^2}}} \\ &= \frac{|\textsc{opt}|}{8}\cdot\frac{1}{k}\sqrt{\frac{c\overline{\sigma}}{(1-\beta)\overline{\sigma^2}}}\ ,\end{aligned}$$

since $T\cdot\overline{\sigma} = k\cdot|\mathcal{I}|$ (cf. Eq. (2)). ∎

Similarly to Corollary 2 we have:

Corollary 5. $\mathbb{E}[|\textsc{Priority}|] \geq \dfrac{|\textsc{opt}|}{8k\sqrt{(1-\beta)\sigma_{\max}/c}}$.

C. The Effect of Variable Burst Size

In this section we provide sharper bounds for the case where all bursts are of the same size.

Theorem 6. $\mathbb{E}[|\textsc{Priority}|] \geq |\textsc{opt}|\cdot\overline{\sigma}^2/(8k\overline{\sigma^2})$. *Furthermore, if* $c=1$, *then* $\mathbb{E}[|\textsc{Priority}|] \geq |\textsc{opt}|\cdot\overline{\sigma}^2/(k\overline{\sigma^2})$.

Proof: It follows from Lemma 9 that

$$\mathbb{E}[|\textsc{Priority}|] \geq \min\left\{\tfrac{|\mathcal{I}|}{2}, \tfrac{c|\mathcal{I}|^2}{8T(1-\beta)\overline{\sigma^2}}\right\}\ .$$

If $\mathbb{E}[|\textsc{Priority}|] \geq \frac{1}{2}|\mathcal{I}|$, then we are done. Otherwise,

$$\begin{aligned}\mathbb{E}[|\textsc{Priority}|] &\geq \frac{cn^2}{8T(1-\beta)\cdot\overline{\sigma^2}} \\ &= \frac{cT^2\overline{\sigma}^2}{8T(1-\beta)k^2\overline{\sigma^2}} \geq \frac{\overline{\sigma}^2}{8k\overline{\sigma^2}}\cdot|\textsc{opt}|\ ,\end{aligned}$$

where the equality is due to the fact that $nk = T\overline{\sigma}$, and the last equality is due to the fact that $|\textsc{opt}| \leq Tc/(1-\beta)k$.

The result for $c=1$ is obtained by using Lemma 5. ∎

The following corollary is for the case of uniform σ.

Corollary 7. *If all bursts have the same size, then* $\mathbb{E}[|\textsc{Priority}|] \geq |\textsc{opt}|/(8k)$. *Furthermore, if* $c=1$ *then* $\mathbb{E}[|\textsc{Priority}|] \geq |\textsc{opt}|/k$.

IV. Buffer Management

In this section we consider the case where sufficient buffer space is available, assuming that all burst sizes are equal. We do it in two steps. First, we analyze Algorithm Priority for large capacity links. We show that if $c = \Omega(\frac{1}{\epsilon^2(1-\epsilon)}\log\frac{(1-\beta)k}{\epsilon})$ for some $\epsilon>0$, then the link utilization is at least $(1-\epsilon)^2(1-\beta)$. Then, we reduce the model with large buffers and arbitrary link capacity to the model with no buffers and large link capacity by a simple "dual buffer" algorithm. This reduction allows us to prove link utilization of at least $(1-\epsilon)^3(1-\beta)$, if all bursts have the same size σ and the buffer size satisfies $b \geq \frac{6k\log(k/\epsilon)}{\epsilon^2(1-\epsilon)^2}$.

A. Large Capacity

In this section we analyze Algorithm Priority for the case where $c = \Omega(\frac{1}{\epsilon^2(1-\epsilon)}\log\frac{(1-\beta)k}{\epsilon})$ and no buffers, i.e., $b=0$. Intuitively, the idea of the analysis in this case is that since in each time step the algorithm can choose relatively many packets to serve, the law of large numbers should apply, in the sense that if a superpacket has "sufficiently good" priority, it is very likely that all of its packets will be served. We now formalize this intuition.

For each superpacket S, define $\sigma(S) \stackrel{\text{def}}{=} \max_{p\in S}\sigma(\text{arr}(p))$, namely $\sigma(S)$ is the size of the largest burst of any packet of S.

Lemma 10. *Let* $\epsilon>0$. *For every* $S\in\mathcal{I}$,

$$\Pr[S\in\textsc{Priority}] \geq \frac{(1-\epsilon)c}{\sigma(S)}\left(1-(1-\beta)ke^{-\epsilon^2(1-\epsilon)c/3}\right)\ .$$

Proof: Let S be a superpacket and let t be a time step such that $S\in\mathcal{I}(t)$. If $\sigma(t)=1$, then S's packet is delivered. Hence, we assume that $\sigma(t)>1$.

Observe that

$$\Pr\left[r(S_j)\leq\frac{(1-\epsilon)c}{\sigma(t)-1}\right] = \frac{(1-\epsilon)c}{\sigma(t)-1}\ ,$$

for every $S_j\in\mathcal{I}(t)$.

Now, define the following random variable

$$X_{tj} = \begin{cases}1 & r(S_j)\leq\frac{(1-\epsilon)c}{\sigma(t)-1},\\ 0 & \text{otherwise,}\end{cases}$$

for every $S_j\in\mathcal{I}'(t)$, where $\mathcal{I}'(t)\stackrel{\text{def}}{=}\mathcal{I}(t)\setminus\{S\}$. Also, let $X_t = \sum_{S_j\in\mathcal{I}'(t)}X_{tj}$. It follows that

$$\mathbb{E}[X_t] = \sum_{S_j\in\mathcal{I}'(t)}\frac{(1-\epsilon)c}{\sigma(t)-1} = (1-\epsilon)c\ .$$

Using a relative Chernoff bound we have,

$$\begin{aligned}\Pr[X_t\geq c] &< \Pr[X_t > (1+\epsilon)E[X_t]] \\ &< e^{-E[X_t]\cdot\epsilon^2/3} \\ &\leq e^{-c\cdot\epsilon^2(1-\epsilon)/3}\ .\end{aligned}$$

S is successfully delivered if its packets are delivered in its first $(1-\beta)k$ bursts, by union bound it follows that

$$\Pr[X_t\leq c, \forall t \text{ s.t.} S\in\mathcal{I}(t)] \leq 1-(1-\beta)ke^{-c\cdot\epsilon^2(1-\epsilon)/3}\ ,$$

and therefore

$$\Pr[S\in\textsc{Priority}] \geq \frac{(1-\epsilon)c}{\sigma(S)}\left(1-(1-\beta)ke^{-c\cdot\epsilon^2(1-\epsilon/3}\right)$$

as required. ∎

Theorem 8. *If* $c\geq\frac{3}{\epsilon^2(1-\epsilon)}\log\frac{(1-\beta)k}{\epsilon}$, *then*

$$\Pr[S\in\textsc{Priority}] \geq c(1-\epsilon)^2/\sigma(S)\ ,$$

for every $S\in\mathcal{I}$.

Proof: By Lemma 10 we have that

$$\begin{aligned}\Pr[S \in \textsc{Priority}] &\geq \frac{(1-\epsilon)c}{\sigma(S)}\left(1-(1-\beta)ke^{-c\cdot\epsilon^2(1-\epsilon)/3}\right)\\ &\geq \frac{(1-\epsilon)c}{\sigma(S)}\left(1-(1-\beta)ke^{-\log\frac{k}{\epsilon}}\right)\\ &\geq \frac{(1-\epsilon)^2 c}{\sigma(S)}\ ,\end{aligned}$$

and we are done. ∎

Corollary 9. *If all bursts have the same size σ and the link capacity satisfies $c \geq \frac{3}{\epsilon^2(1-\epsilon)}\log\frac{(1-\beta)k}{\epsilon}$, then the goodput utilization is at least $(1-\epsilon)^2(1-\beta)$.*

Proof: Since in every time t we have σ packets, the number of superpackets is $T\sigma/k$. The expected number of superpackets that are delivered is

$$\sum_S (1-\epsilon)^2\frac{c}{\sigma} = (1-\epsilon)^2\frac{c}{\sigma}\frac{T\sigma}{k} = (1-\epsilon)^2\frac{cT}{k}$$

Clearly, one cannot hope to deliver more than cT packets, and hence $\textsc{opt} \leq \frac{cT}{k(1-\beta)}$ frames. ∎

B. Buffering

In this section we consider the case where sufficient buffer space is available. Specifically, we present an algorithm whose link utilization is at least $(1-\epsilon)^3$, if all bursts have the same size σ and the buffer space satisfies $b \geq \frac{6k\log(k/\epsilon)}{\epsilon^2(1-\epsilon)^2}$.

Let us assume, for convenience, that $b = \ell\cdot c$ where c is the link capacity, for some integer $\ell \geq 2$. Our algorithm, ALG, is a dual buffer algorithm, defined as follows. The buffer is divided into two parts, of size $b/2$ each. At any given time, one part is used to receive arriving packets, and the other part is used to send packets. Only the parts used for receiving may overflow, in which case it retains only the highest priority packets. Every $b/(2c)$ time steps, the role of the buffers is switched: the receiving part starts transmitting, and the transmitting part (by now empty) starts receiving. This concludes the description of the algorithm. We now analyze it.

Lemma 11. *Let $\epsilon \geq 0$. If $b \geq 2k/\epsilon$, then*

$$\Pr[S \in \textsc{alg}] \geq \frac{c(1-\epsilon)^2}{\sigma}\left(1-ke^{-b\epsilon^2(1-\epsilon)^2/(6k)}\right)\ .$$

Proof: $S \in \textsc{alg}$ if it survives every block ω of $b/(2c)$ time steps in which its packets appear. Intuitively, the above algorithm simulates a scheduler executing Algorithm Priority, where the bursts contain $\sigma' = \sigma b/(2c)$ packets and the capacity is $c' = b/2$. The main difference is that some of the packets in the burst of size σ' may belong to the same superpacket. However, observe that we still have at least $\sigma b/(2kc)$ different superpackets within each burst.

We follow the same logic as in Lemma 10. Fix a block ω of $b/(2c)$ time steps in which a packet from S appears, and let n_S be the number of packets from S that appear in ω. Also, let $\mathcal{B}_\omega$ be the set of superpackets that appear in ω. For $S_j \in \mathcal{B}_\omega \setminus \{S\}$ define the following random variable,

$$X_{\omega,j} = \begin{cases}\frac{n_{\omega,j}}{k} & r(S_j) \leq \frac{(1-\epsilon)(c'-k)}{\sigma'-n_S},\\ 0 & \text{otherwise,}\end{cases}$$

where $n_{\omega,j}$ is the number of packets from S_j that appear in ω. Observe that $\sum_j n_{\omega,j} = \sigma' - n_S$. Let $X_\omega = \sum_{S_j\in\mathcal{B}\setminus\{S\}} X_{\omega,j}$. Note that if $X_\omega \leq (b/2-k)/k$ and $r(S) \leq \frac{(1-\epsilon)(c'-k)}{\sigma'-n_S}$ then all packets of S in ω are delivered. We lower bound the probability of this event. First, note that

$$\mathbb{E}[X_\omega] = \sum_{S_j\in\mathcal{B}\setminus\{S\}} \frac{(1-\epsilon)(c'-k)}{\sigma'-n_S}\cdot\frac{n_{\omega,j}}{k} = \frac{(1-\epsilon)(b/2-k)}{k}\ .$$

Using a relative Chernoff bound we have,

$$\begin{aligned}\Pr\left[X_\omega > \frac{b/2-k}{k}\right] &< \Pr[X_\omega > (1+\epsilon)E[X_\omega]]\\ &< e^{-E[X_\omega]\cdot\epsilon^2/3}\\ &= e^{-\frac{(1-\epsilon)(b/2-k)}{k}\cdot\epsilon^2/3}\\ &\leq e^{-\epsilon^2(1-\epsilon)^2 b/(6k)}\ ,\end{aligned}$$

because $b/2-k \geq (b/2)(1-\epsilon)$. Hence

$$\Pr[X_\omega \leq b/2-k, \forall\omega \text{ s.t.} S\in\mathcal{B}_\omega] \leq 1-ke^{-b\cdot\epsilon^2(1-\epsilon)^2/(6k)}\ ,$$

by union bound, and therefore

$$\begin{aligned}\Pr[S\in\textsc{alg}] &\geq \frac{(1-\epsilon)(c'-k)}{\sigma'-n_S}\left(1-ke^{-b\cdot\epsilon^2(1-\epsilon)^2/(6k)}\right)\\ &\geq \frac{c(1-\epsilon)^2}{\sigma}\left(1-ke^{-b\cdot\epsilon^2(1-\epsilon)^2/(6k)}\right)\end{aligned}$$

as required. ∎

Similarly to Theorem 8 we have

Theorem 10. *If $b \geq \frac{6k\log(k/\epsilon)}{\epsilon^2(1-\epsilon)^2}$, then for every $S \in \mathcal{I}$:*

$$\Pr[S\in\textsc{alg}] \geq c(1-\epsilon)^3/\sigma\ .$$

Analogously to Corollary 9 also have

Corollary 11. *If all bursts have the same size σ and $b \geq \frac{6k\log(k/\epsilon)}{\epsilon^2(1-\epsilon)^2}$, then the link utilization is at least $(1-\epsilon)^3(1-\beta)$.*

V. Few Large Bursts

The analysis in Section IV assumed that burst sizes are fixed, and now we extend it to the case where burst sizes may vary. While arbitrary variability is hard to cope with, we believe that in many practical situations something can be done. The rationale is that in practical many situations, it appears reasonable to assume that overflow events occur infrequently, namely in most time steps $\sigma(t) \leq c$. The idea is that by having a little redundancy, we can overcome the damage incurred by the few large bursts. We formalize this idea below.

First, we define a notion of large burst. Given $\alpha > 1$, define an *α-large burst* (or just large burst, when α is clear by context) to be a burst satisfying $\sigma(t) > \alpha\cdot c$. We shall focus on

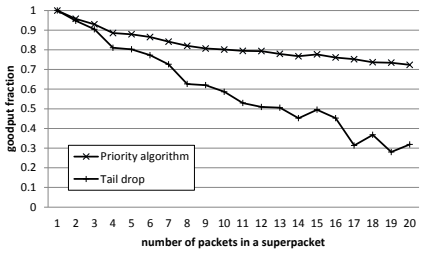

Fig. 1. The effect of k. $\overline{\sigma} = 4.67$ packet/time slot on average, link capacity is $c = 5$ packets per slot and buffer size is $b = 10$ packet slots.

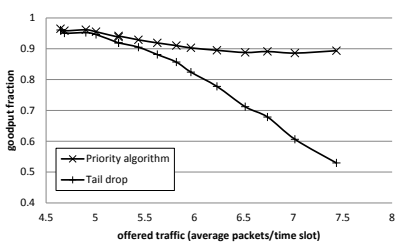

Fig. 2. The effect of varying the offered load. Here $k = 4$, link capacity is $c = 6$ packets per slot and buffer size is $b = 10$ buffer slots.

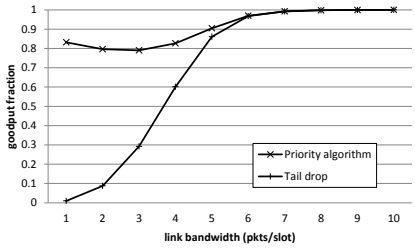

Fig. 3. The effect of varying the link capacity. Here $k = 4$, average offered packets/time slot is $\overline{\sigma} = 4.5$, and buffer size is $b = 10$ packet slots.

the "good" packets of a superpackets S, namely the packets whose corresponding bursts are not large. Formally, given a superpacket S, define S_α to be the packets p of S for which $\sigma(\text{arr}(p)) \leq \alpha \cdot c$, namely

$$S_\alpha = \{p \in S : \sigma(\text{arr}(p)) \leq \alpha \cdot c\} \ .$$

Our general idea is forfeit all packets in large bursts, and account only for the other packets, leveraging redundancy. We rely on the following argument. If for some $0 \leq \delta_S < 1$, a superpacket S participates in $\delta_S k$ large bursts (i.e., $|S_\alpha| = (1-\delta_S)k$), and the redundancy parameter satisfies $\beta > \delta_S$, then S is delivered if its packets are delivered in at least $(1-\beta)k$ non-large bursts. More formally, we prove the following.

Lemma 12. *Let $S \in \mathcal{I}$ be a superpacket, and let $\delta_S \stackrel{\text{def}}{=} 1 - \frac{|S_\alpha|}{k}$. If $\delta_S < \beta$ then $\Pr[S \in \text{PRIORITY}] > \frac{1}{4\alpha} \cdot \frac{\beta - \delta_S}{1 - \delta_S}$.*

The proof of the above lemma is omitted due to space constraints. Next, given $0 \leq \delta < \beta < 1$, define

$$\mathcal{I}_{1-\delta} = \{S \in \mathcal{I} : |S_\alpha| \geq (1-\delta)k\} \ ,$$

namely $\mathcal{I}_{1-\delta}$ contains all superpackets that have at most δk packets contained in large bursts. For $\mathcal{I}_{1-\delta}$ we prove the following result.

Theorem 12. *Algorithm* PRIORITY *completes at least* OPT $\cdot \frac{|\mathcal{I}_{1-\delta}|}{|\mathcal{I}|} \cdot \frac{\beta - \delta}{4\alpha(1-\delta)}$ *superpackets.*

Proof: Due to Lemma 12 we have that

$$\mathbb{E}[|\text{PRIORITY}|] \geq \sum_{S \in \mathcal{I}_{1-\delta}} \frac{1}{4\alpha} \cdot \frac{\beta - \delta_S}{1 - \delta_S} \geq \frac{|\mathcal{I}_{1-\delta}|}{4\alpha} \cdot \frac{\beta - \delta}{1 - \delta} \ ,$$

while $|\text{OPT}| \leq |\mathcal{I}|$. ∎

Setting $\delta = \beta/2$ and $\delta = 0$ in Theorem 12, we obtain

Corollary 13. *The competitive ratio of Algorithm* PRIORITY *is at most*

- $\frac{|\mathcal{I}|}{|\mathcal{I}_{1-\beta/2}|} \cdot \frac{4\alpha(2-\beta)}{\beta}$.
- $\frac{4\alpha}{\beta}$, *if $\sigma(t) = \alpha \cdot c$, for every t.*

Combining Corollary 9 and Theorem 12 yields:

Corollary 14. *If all bursts have the same size $\sigma = \alpha \cdot c$, for $\alpha > 1$, and $c \geq \frac{3}{\epsilon^2(1-\epsilon)} \log \frac{(1-\beta)k}{\epsilon}$, then the competitive ratio is at most $\frac{1+4\alpha(1-\epsilon)^2}{\alpha(1-\epsilon)^2}$.*

VI. SIMULATION

While most of our results are worst-case type of results with adversarial input, it seems also interesting to see how does the system perform when the input is generated by some stochastic process. We have implemented a simulation that allows us to observe the system behavior under a variety of conditions. In this section we present some of the simulation results. The buffer policy we employ here is a single buffer, rather than the dual buffer analyzed in section IV, because we believe it is much more likely to be used in practice.

A. Parameters

Our system implements the model as described in Section II. Our tunable parameters, set in each run, are

- k, the number of packets per superpacket,
- c, the link capacity in packets per step,
- b, the number of packet slots in the buffer, and
- β, the portion of the superpacket which is redundant.

The results presented below are for traffic generated by an aggregate of 10 on-off process, with a tunable parameter $\lambda \stackrel{\text{def}}{=} \lambda_{\text{on}}/\lambda_{\text{off}}$. The association of packets with superpackets is determined by a random permutation. Each datapoint represents the average of 10 runs.

B. Comparison Base

Our analytical bounds are stated in terms of the "competitive ratio," defined with respect to OPT, the optimal performance for each given sequence. Since computing the optimum is infeasible even off-line (it is NP-hard to approximate to within a factor $O(k/\log k)$, see [13]), in many cases we state the performance results of our algorithm as the fraction of *goodput*, defined to be the number of packets transmitted and used in completed frames divided by the total number of transmission slots available to the algorithm.

Also, to get a better perspective on the algorithm, we compare the performance of our algorithm with the performance of the simplistic tail-drop policy, which, in case of overflow, discards the last packets to arrive. We note that since the superpacket IDs are random, tail-drop in our simulation is equivalent to dropping packets at random upon overflow.

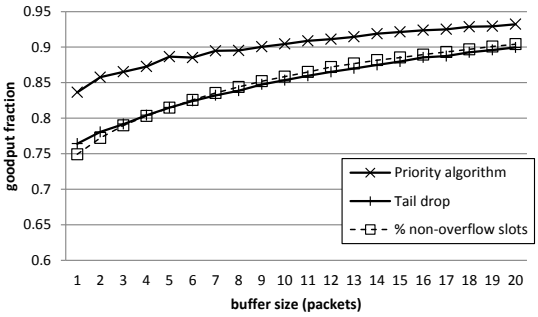

Fig. 4. The effect of varying the buffer size. Here $k = 4$, link capacity is $c = 5$ packets per slot and average number of offered packet/timeslot is $\overline{\sigma} = 4.66$.

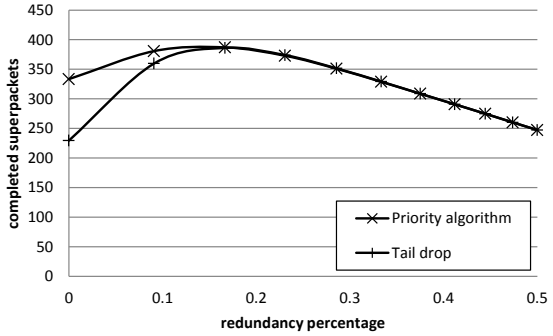

Fig. 5. The effect of adding redundancy to 10-packet superpackets. Adding 1 packet means $1/11$ redundancy, adding 2 packets means $2/12$ redundancy etc. The capacity is $c = 5$ packets per slot, average number of offered packet/timeslot is $\overline{\sigma} = 4.94$, and buffer size is $b = 10$.

C. Results

In Figure 1 we can see the effect of varying the number of packets per superpacket. While the goodput of tail-drop decreases almost linearly with k, the effect on PRIORITY is much less pronounced. These numbers were taken with moderately high load.

Next we measure the effect of offered load (Figure 2). Again, the fraction of goodput packets of tail-drop decreases linearly, while Priority manages to maintain close to 90% goodput even when the offered load is 124% capacity (with overflows occurring in 65% of the time slots).

Another view of the effect of overload is presented in Figure 3, where we vary the link capacity while holding offered load fixed. The effect on goodput is by far stronger for tail-drop (whose starting point is worse). We note that the non-monotonic plot for Priority is due to the normalization by available time-slots, which apparently provide a poor estimate of the optimal performance when the capacity is small. Nearly full goodput utilization is attained when the link bandwidth approaches twice the average offered load.

Our next figure gives a flavor of the effect of varying the buffer size (Figure 4). We also plot the percentage of overflow events from the set of time steps. Note that it matches quite closely the goodput percentage of tail-drop, which makes sense when we recall that in our set-up, tail-drop is equivalent to random packet drop.

Finally, we present results for variable redundancy (Figure 5). To be fair, we fix the number of packet arrivals per slot (and thus to total number of possible transmission slots), and observe the effect of adding redundancy packet to superpackets whose information spans 10 packets. This means that as we increase redundancy, the total number of offered superpackets decreases. As expected, adding too much redundancy hurts the goodput: adding just 1–2 redundancy packets yields the best goodput. Please note that in Figure 5 the y axis counts the total number of superpackets delivered (and not goodput fraction).

REFERENCES

[1] J. M. Boyce and R. D. Gaglianello, "Packet loss effects on mpeg video sent over the public internet," in *MULTIMEDIA '98: Proceedings of the sixth ACM international conference on Multimedia*. New York, NY, USA: ACM, 1998, pp. 181–190.

[2] M. Luby, "A simple parallel algorithm for the maximal independent set problem," in *17th Annual ACM Symposium on the Theory of Computing*, 1985, pp. 1–10.

[3] Y. Emek, M. M. Halldórsson, Y. Mansour, B. Patt-Shamir, J. Radhakrishnan, and D. Rawitz, "Online set packing and competitive scheduling of multi-part tasks," in *29th Annual ACM Symposium on Principles of Distributed Computing*, 2010.

[4] Y. Mansour, B. Patt-Shamir, and O. Lapid, "Optimal smoothing schedules for real-time streams," in *19th Annual ACM Symposium on Principles of Distributed Computing*, Portland, Oregon, July 2000, pp. 21–30.

[5] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko, "Buffer overflow management in QoS switches," in *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, July 2001, pp. 520–529.

[6] M. H. Goldwasser, "A survey of buffer management policies for packet switches," *SIGACT News*, vol. 41, no. 1, pp. 100–128, 2010.

[7] A. Kesselman, B. Patt-Shamir, and G. Scalosub, "Competitive buffer management with packet dependencies," in *23rd IPDPS*, 2009, pp. 1–12.

[8] J. L. Gabriel Scalosub and P. Marbach, "Buffer management for aggregated streaming data with packet dependencies," in *Proceedings of IEEE INFOCOM 2010 Mini-conference*, 2010.

[9] J. Håstad, "Clique is hard to approximate within $n^{1-\epsilon}$," *Acta Mathematica*, vol. 182, no. 1, pp. 105–142, 1999.

[10] M. M. Halldórsson, J. Kratochvíl, and J. A. Telle, "Independent sets with domination constraints," *Discrete Applied Mathematics*, vol. 99, no. 1–3, pp. 39–54, 2000.

[11] C. A. J. Hurkens and A. Schrijver, "On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems," *SIAM J. Discrete Math.*, vol. 2, no. 1, pp. 68–72, 1989.

[12] P. Berman, "A $d/2$ approximation for maximum weight independent set in d-claw free graphs," *Nord. J. Comput.*, vol. 7, no. 3, pp. 178–184, 2000.

[13] E. Hazan, S. Safra, and O. Schwartz, "On the complexity of approximating k-dimensional matching," in *6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, ser. LNCS, vol. 2764, 2003, pp. 83–97.