Buffer Overflows of Merging Streams

Alex Kesselman[1], Zvi Lotker[2], Yishay Mansour[3], and Boaz Patt-Shamir[4]

[1] School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel.
alx@cs.tau.ac.il
[2] Dept. of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel.
zvilo@eng.tau.ac.il
[3] School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel.
mansour@cs.tau.ac.il
[4] Cambridge Research Lab, Hewlett-Packard, One Cambridge Center, Cambridge, MA 02142.
On leave from Dept. of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel.
Boaz.PattShamir@HP.com

Abstract. We consider a network merging streams of packets with different quality of service (QoS) levels, where packets are transported from input links to output links via multiple merge stages. Each merge node is equipped with a finite buffer, and since the bandwidth of a link outgoing from a merge node is in general smaller than the sum of incoming bandwidths, overflows may occur. QoS is modeled by assigning a positive value to each packet, and the goal of the system is to maximize the total value of packets transmitted on the output links. We assume that each buffer runs an independent local scheduling policy, and analyze FIFO policies that must deliver packets in the order they were received. We show that a simple local on-line algorithm called Greedy does essentially as well as the combination of locally optimal (off-line) schedules. We introduce a concept we call the *weakness* of a link, defined as the ratio between the longest time a packet spends in the system before transmitted over the link, and the longest time a packet spends in that link's buffer. We prove that for any tree, the competitive factor of Greedy is at most the maximal link weakness.

1 Introduction

Consider an Internet service provider (ISP), or a corporate intranet, that connects a large number of users with the Internet backbone using an "uplink." Within such a system, consider the traffic oriented towards the uplink, namely the streams whose start points are the local users and whose destinations are outside the local domain. Then streams are merged by a network that consists of *merge nodes*, typically arranged in a tree topology whose root is directly connected to the uplink. Without loss of generality, we may assume that the bandwidth of the link emanating from a merge node is less than the sum of bandwidths of incoming links (otherwise, we can assume that the incoming links are connected directly to the next node up). Hence, when all users inject data at maximum local speed, packets will eventually be discarded. A very effective way to mitigate some of the losses due to temporary overloads is to equip the merge nodes with buffers, that can absorb transient bursts by storing incoming packets while the outgoing link is busy. The merge nodes are controlled by local on-line *buffer management algorithms* whose job is to decide which packets to forward and which to drop so as to minimize the damage in case of an overflow.

In this paper we study the performance of various buffer management algorithms in the context of a system of merging streams, under the assumption that the system is required to support different quality of service (QoS) levels. The different QoS levels are modeled by assuming that each packet has a positive value, and that the goal of the system is to maximize the total value of packets delivered.

Evaluating the performance of the system cannot be done in absolute terms, since the total value delivered depends on the actual streams that arrive. Instead, we measure the competitive ratio of the algorithm [18] by bounding, over all possible input sequences, the ratio between the value gained by the algorithm in question, and the best possible value that can be gained by any schedule.

Our model. To allow us to describe our results, let us give here a brief informal overview of the model (more details are provided in Section 2). Our model is essentially the model used by Adversarial Queuing Theory [5], with the following important differences: packet injection is unrestricted, buffers are finite, and each packet has a value. More specifically, the system is described by a communication graph, where each link e has a buffer Q_e in its ingress and a prescribed bandwidth $W(e)$. An execution of the system proceeds in synchronous steps. In each step, new packets may enter the system, where each packet has a value (in $\mathbb{R}^+$), and a completely specified route. Also in each step, packets may progress along edges, some packets may be dropped from the system, and some packets may be absorbed by their destinations. The basic limitation on these actions is that for each edge e, at most $W(e)$ packets may cross it in each step, and at most $size(Q_e)$ packets may be retained in the buffer from step to step. The task of the buffer management algorithm is to decide which packets to forward and which packets to drop subject to these restrictions. Given a system and an input sequence, the total value of a schedule for that input is the total value of the packets that reach their destinations.

In this paper, we consider a few special cases of the general model above, justified by practical engineering considerations. The possible restrictions are on the network topology, scheduling algorithms, and packet values. The variants are as follows. *Tree topology* assumes that the union of the paths of all packets is a directed tree, where all paths start from a leaf and end at the root of the tree. Regarding schedules, our results are for the class of *work-conserving* schedules, i.e., schedules that always forward a packet when the buffer is non-empty [9].[5] We consider the class of FIFO algorithms, i.e., algorithms that may not send a packet that arrives late before a packet that arrives early. This condition is natural for many network protocols (e.g., TCP).

Our results. We study the effect of different packet values, different buffer sizes and link bandwidths on the competitiveness of various local algorithms. We study very simple Greedy algorithm that drops the least valuable packets available when there is an overflow. We also consider the Locally Optimal schedule, which is the best possible schedule with respect to a single buffer. Roughly speaking, it turns out that in many

[5] Work conserving schedules are sometimes called “greedy” [16, 5]. In line with the networking community, we use the term “work conserving” here; we reserve the term “greedy” for a specific algorithm we specify later.

cases, the Greedy algorithm has performance which is asymptotically equivalent to the performance of a system defined by a composition of locally optimal schedules, and in some cases, its performance is proportional to the global optimum. More specifically, we obtain the following results.

First, we present simple scenarios that show that local algorithms cannot be too good: specifically, even allowing each node to run the locally optimal (offline) schedule may result in competitive ratio of $\Omega(h)$ on height-h trees with uniform buffer sizes and uniform link bandwidths. For bounded degree trees of height h, the competitive factor drops to $\Omega(h/\log h)$, and for trees of height h and $O(h)$ nodes, the lower bound drops further to $\Omega(\sqrt{h})$.

Next, we analyze the Greedy algorithm. By extending the analysis of the single buffer case, we show that for arbitrary topologies, the maximal ratio between the performance of Greedy and the performance of any work-conserving (off-line) schedule is $O(DR/B_{\min})$, where D is the length of the longest packet route (measured in time units), R is the maximal rate in which packets may reach their destinations, and $B_{\min}$ is the size of the smallest buffer in the system.

We then focus on tree topologies, where we present our most interesting result. We introduce the concept of *link weakness*, defined as follows. For any given link e, define the *delay* of e to be the longest time a packet can spend in the buffer of e (for work-conserving schedules, it's exactly the buffer size divided by the link bandwidth). Define further the *height* of e to be the maximal length of a path from an input leaf to the egress of e, where the length of a link is its delay. Finally, the *weakness* of e, denoted $\lambda(e)$, is the ratio between its height and its delay (we have that $\lambda(e) \geq 1$). Our main result is that the competitive factor of Greedy is proportional to the maximal link weakness in the system. Our proof is for the case where each packet has one of two possible values.

Related work. There is a myriad of research papers about packet drop policies in communication networks—see, e.g., the survey of [13] and references therein. Some of the drop mechanisms (most notably RED [7]) are designed to signal congestion to the sending end. The approach abstracted in our model is implicit in the recent DiffServ model [4, 6] and ATM [19].

There has been work on analyzing various aspects of this model using classical queuing theory, and assuming Poisson arrivals [17]. The Poisson arrival model has been seriously undermined by recent discoveries regarding the nature of traffic in computer networks (see, e.g., [14, 20]).

In this work we use competitive analysis, which studies the worst-case performance guarantees of an on-line algorithm relative to an off-line solution. This approach is used in Adversarial Queuing Theory [5], where packet injections are restricted, and the main measure of performance is the size of the buffers required to never drop any packet. In a recent paper, Aiello et al. [1] propose to study the *throughput* of a network with bounded buffers and packet drops. Their model is similar to ours, so let us point out the differences. The model of [1] assumes uniform buffer sizes, link bandwidths, and packet values, whereas we consider individual sizes, bandwidths and values. As we show in this paper, these factors have a decisive effect on the competitiveness of the system even in very simple cases. Another difference is that [1] compares on-line algorithms to any off-line schedule, including ones that are not work-conserving. Due to this approach, the performance guarantees they can prove are rather weak, and thus they are mainly interested in whether the competitive factor of a scheduling policy is finite or not. By contrast, we consider work-conserving off-line schedules, which allow us to derive quantitative results and gain more insights from the practical point of view.

Additional relevant references study the performance guarantees of a single buffer, where packets have different values. The works of [2, 12] study the case where one cannot preempt a packet already in the buffer. In [10], an upper bound of 2 is proven for the competitive factor of the greedy algorithm. The two-value single buffer case is further studied in [11, 15]. Overflows in a shared-memory switch are considered in [8].

A recent result of Azar and Richter [3] analyzes a scenario of stream merging in input-queued switches. Briefly, finite buffers are located at input ports; the output port has no buffer: it selects, at each step, one of the input buffers and transmits the packet in the head of that buffer. Their main result is a centralized algorithm that reduces this scenario of a single merge to the problem of managing a single buffer, while incurring only a constant blowup in the competitive factor.

Paper organization. Section 2 contains the model description. Lower and upper bounds for local schedules are considered in Section 3 and Section 4, respectively.

2 Model and Notation

We start with a description of the general model.

The system is defined by a directed graph $G = (V, E)$, where each link $e \in E$ has *bandwidth* (or *speed*) $W(e) \in \mathbb{N}$, and a *buffer* Q_e with storage capacity $size(Q_e) \in \mathbb{N} \cup \{0\}$. (The buffer resides at the link's ingress—see below.)

The input to the system is a sequence of *packet injections*, one for each time step. A packet injection is a set of packets, where each packet p is characterized by its route, denoted $route(p)$, and its value, denoted $\omega(p)$.[6] The first node on the route is called the packet's *source*, and the last node is called the packet's *destination*. To avoid trivialities, we assume that each packet route is a simple path that contains at least one link.

The *execution* (or *schedule*) of the system proceeds in synchronous steps as follows. The state of the system is defined by the current contents of each link's buffer Q_e, and by each link's *transit contents*, denoted $transit_e$ for a link e. Initially, all buffers and transit contents are empty sets. Each step consists of the following substeps.

(1) Packet injection: For each link e, an arbitrary set of new packets whose first link is e is added to Q_e.

(2) Packet delivery: For all links $e_1 = (u, v)$ and $e_2 = (v, w)$, all packets currently in $transit_{e_1}$ whose next route edge is e_2 are moved from $transit_{e_1}$ into Q_{e_2}. All packets whose destination is v are *absorbed*. After this substep, $transit_e = \emptyset$ for all $e \in E$.

(3) Packet drop: A subset of the packets currently stored in Q_e is removed from Q_e, for each $e \in E$.

(4) Packet send: For each link e, a subset of the packets currently stored in Q_e is moved from Q_e to $transit_e$.

[6] There may be many packets with the same route and value, so technically each packet injection is a multiset; we abuse notation slightly, and always refer to multisets when we say "sets."

We stress that packet injection rate is unrestricted (as opposed, e.g., to Adversarial Queuing Theory). Note also that we assume that all link latencies are one unit.

A *scheduling algorithm* determines which packets to drop (Substep 3) and which packets to send (Substep 4), so as to satisfy the following conditions after each step is completely done:

- For each link e, the number of packets stored in Q_e is at most $size(Q_e)$.[7]
- For each link e, the total number of packets stored in the transit contents of e is at most $W(e)$.

Given an input sequence $\mathcal{I}$ and an algorithm A for a system, the *value* obtained by A for $\mathcal{I}$, denoted $\omega_A(\mathcal{I})$, is the sum of values of all packets that have reached their destination.

Tree Topology. A system is said to have *tree topology* if the union of all packet routes used in the system is a tree, where packet sources are leaves and all packets are destined at the single root. In this case each node except the root has a single storage buffer (associated with its unique outgoing edge), sometimes referred to as the *node*'s buffer. It is convenient also to assume in the tree case that the leaves and root are links: this way, we have streams entering the system and a stream leaving the system. We say that a node v is *upstream* from u (or, equivalently, u is *downstream* from v), if there is a directed path from v to u.

FIFO Schedules. We consider FIFO schedules, which adhere to the rule that packets are sent over a link in the same order they enter the buffer at the tail of the link (packets may be arbitrarily dropped by the algorithm, but the packets that do get sent preserve their relative order). More precisely, for all packets p, q and every link e: If p is sent on e at time t and q is sent on e at time $t' > t$, then q did not enter Q_e before p.

Work-Conserving Schedules. A given schedule is called *work conserving* if for every step t and every link e we have that the number of packets sent over e at step t is the minimum between $W(e)$ and the number of packets in Q_e (at step t just before Substep 4). Intuitively, a work conserving schedule always forwards the maximal number of packets allowed by the local bandwidth restriction. (Note that packets may be dropped in a work-conserving schedule even if the buffer is not full.)

Algorithms and Their Evaluation. An algorithm is called *local on-line* if its action at time t at node v depends only on the sequence of packets arriving at v up to time t. An algorithm is called *local off-line* if its action at time t at node v depends only on the sequence of packets arriving at v, including packets that arrive at v after t. Given a sequence of packet arrivals and injections at node v, the local-offline schedule with the maximum output value of v for the given sequence is the Local Optimal schedule, denoted OptL_v. When the set of routes is acyclic, we define the schedule OptL to be the composition of Local Optimal schedules, constructed by applying OptL_v in topological order. A *global off-line* schedule has the whole input (at all nodes, at all times) available ahead of any decision. We denote by Opt the global off-line work-conserving schedule with the maximum value.

Given a system and an algorithm A for that system, the *competitive ratio* (or *competitive factor*) of A is the worst-case ratio, over all input sequences, between the value

[7] Note that the restriction applies only *between* steps: in our model, after Substeps 1,2 and before Substeps 3,4, more than $size(Q_e)$ packets may be stored in Q_e.

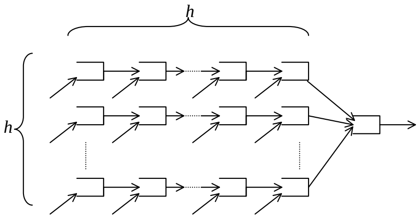

Fig. 1. *Topology used in the proof of Theorem 1, with parameter h. Diagonal arrows represent input links, and the rightmost arrow represents the output link.*

of Opt and the value of A. Formally:

$$\mathrm{cr}(A) = \sup \left\{ \frac{\omega_{\mathsf{Opt}}(\mathcal{I})}{\omega_A(\mathcal{I})} : \mathcal{I} \text{ is an input sequence} \right\} .$$

Since we deal with a maximization problem this ratio will always be at least 1.

3 Lower Bounds for Local Schedules

In this section we consider simple scenarios that establish lower bounds on local algorithms. We show that even if each node runs OptL – a locally optimal schedule (that may be computed off-line) – the performance cannot be very close to the globally optimal schedule.

As we are dealing with lower bounds, we will be interested in very simple settings. In the scenarios below, all buffers have the same size B and all links have bandwidth 1. Furthermore, we use only two packet values: low value of 1, and high value of $\alpha > 1$. (The bounds of Theorems 2 and 3 are tight for the two-value case; we omit details here.)

As an immediate corollary of Theorem 4, we have that the the lower bound of Theorem 1 is tight, as argued below.

Theorem 1. *The competitive ratio of* OptL *for a tree-topology system is* $\Omega(\min(h, \alpha))$*, where* h *is the depth of the tree.*

Proof: Consider a system with $h^2 + 1$ nodes, where h^2 "path nodes" have input links, and are arranged in h paths of length h each, and one "output node" has input from the h last path nodes, and has one output link (see Figure 1). Let B denote the size a buffer. The input sequence is as follows. The input for all nodes in the beginning of a path is B packets of value α followed by B packets of value 1 (at steps $0, \ldots, 2B-1$). The input for the i-th node on each path for $i > 1$ is B packets of value 1 at time $B(i-2)+i-1$.

Consider the schedule of OptL first. There are no overflows on the buffers of the path nodes, and hence it is easy to verify by induction that the output from the i-th node on any path contains $B \cdot i$ packets of value 1, followed by B packets of value α. Thus, the output node gets h packets of value 1 in each time step t for $t = h, \ldots, h \cdot B$, and h packets of value α in each time step t for $t = h \cdot B + 1, \ldots, (h+1) \cdot B + 1$. Clearly, the value of OptL in this case consists of $(h-1)B$ low value packets and $2B$ high value packets.

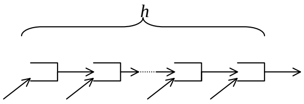

Fig. 2. *A line of depth h. Diagonal arrows represent input links, and the rightmost arrow represents the output link.*

On the other hand, the globally optimal schedule Opt is as follows. On the j-th path, the first $B(j-1)$ low value packets are dropped. Thus, the stream outcoming from the j-th path consists of $B(h-(j-1))$ low value packets followed by B high value packets, so that in each time step $t = h, \ldots, hB$ exactly one high value packet and $h-1$ low value packets enter the output node, and Opt obtains the total value of $hB\alpha+B$. It follows that the competitive ratio of OptL in this case is $\frac{h\alpha+1}{(h-1)+2\alpha} = \Omega\left(\frac{h\alpha}{h+\alpha}\right) = \Omega(\min(h,\alpha))$. ▮

If we insist on bounded-degree trees, the above lower bound changes slightly, as stated below. The proof is omitted from this extended abstract.

Theorem 2. *The competitive ratio of* OptL *for a binary tree with depth h is $\Theta(\min(\alpha, \frac{h}{\log h}))$.*

Further restricting attention to a line topology (see Figure 2), the lower bound for $\alpha \gg h$ decreases more significantly, as the following result shows. Proof is omitted.

Theorem 3. *The competitive ratio of* OptL *for a line of length h is $\Theta(\min(\alpha, \sqrt{h}))$.*

4 Upper Bounds for Local Schedules

In this section we study the competitive factor of local schedules. We first prove a simple upper bound for arbitrary topology, and then give our main result which is an upper bound for the tree topology.

4.1 An Upper Bound on Greedy Schedules for General Topology

We now turn to positive results, namely upper bounds on the competitive ratio of a natural on-line local algorithm [10].

Algorithm 1 Greedy*: Never discard packets if there is free storage space. When an overflow occurs, drop the packets of the least value.*

We now prove an upper bound on the competitiveness of Greedy in general topologies. We remark that all lower bounds proved in Section 3 for OptL hold also for Greedy as well (details omitted).

We start with the following basic definition.

Definition 1. *For a given link e in a given system, we define the* delay *of e, denoted $d(e)$, to be the ratio $\lceil size(Q_e)/W(e)\rceil$. The delay of a given path is the sum of the edge delays on that path. The* maximal delay *in a system, denoted D, is the maximal delay over all simple paths in the systems.*

Note that the delay of a buffer is the maximal number of time units a packet can be stored in it under any work-conserving schedule.

We also use the concept of drain rate, which is the maximal possible rate of packet absorption. Formally, it is defined as follows.

Definition 2. *Let Z be the set of all links leading to an output node in a given system. The* drain rate *of the system, denote R, is the sum $\sum_{e \in Z} W(e)$.*

With these notions, we can now state and prove the following general result. Note that the result is independent of node degrees.

Theorem 4. *For any system with maximal delay at most D, drain rate at most R, and buffers with size at least $B_{\min}$, the competitive ratio of* Greedy *is $O(DR/B_{\min})$.*

We remark that the proof given below holds also for OptL.

Proof: Fix an input sequence $\mathcal{I}$. Divide the schedule into time intervals $I_j = [jD, (j+1)D - 1]$ D time steps each. Consider a time interval I_j. Define S_j to be the set of $2DR$ most valuable packets that are injected into the system during I_j. Observe that in a work conserving schedule, any packet is either absorbed or dropped in D time units. It follows that among all packets that arrive in I_j, at most $2DR$ will be eventually absorbed by their destinations: DR may be absorbed during I_j, and DR during the next interval of D time units (i.e. I_{j+1}). Since this property holds for any work-conserving algorithm, summing over all intervals we obtain that for the given input sequence

$$\omega_{\mathsf{Opt}}(\mathcal{I}) \le \sum_j \omega(S_j) \ . \tag{1}$$

Consider now the schedule of Greedy. Let S'_j denote the set of $B_{\min}$ most valuable packets absorbed during I_j, let S''_j denote the $B_{\min}$ most valuable packets stored in one of the buffers in the system when the next interval I_{j+1} starts, and let S^*_j denote the $B_{\min}$ most valuable packets from $S'_j \cup S''_j$. Note that S^*_j is exactly the set of $B_{\min}$ most valuable packets that were in the system during I_j and were not dropped. We claim that

$$\omega(S^*_j) \ge \frac{B_{\min}}{2DR} \omega(S_j) \ . \tag{2}$$

To see that, note that a packet $p \in S_j$ is dropped from a buffer Q_e only if Q_e contains at least $size(Q_e) \ge B_{\min}$ packets with value greater than $\omega(p)$. To complete the proof of the theorem, observe that for all j we have that $\omega(S'_j) \ge \omega(S''_{j-1})$, i.e., the value absorbed in an interval is at least the total value of the $B_{\min}$ most valuable packets stored when the interval starts. Hence, using Eqs. (1,2), and since $S^*_j \subseteq S'_j \cup S''_j$, we get

$$\begin{aligned}
\omega_{\mathsf{Opt}}(\mathcal{I}) &\le \sum_j \omega(S_j) \le \frac{2DR}{B_{\min}} \sum_j \omega(S^*_j) \\
&\le \frac{2DR}{B_{\min}} \left(\sum_j \omega(S'_j) + \sum_j \omega(S''_j) \right) \\
&\le 4 \frac{DR}{B_{\min}} \sum_j \omega(S'_j) = \frac{4DR}{B_{\min}} \cdot \omega_{\mathsf{Greedy}}(\mathcal{I}) \ . \quad \blacksquare
\end{aligned}$$

One immediate corollary of Theorem 4 is that the lower bound of Theorem 1 is tight, as implied by the result below.

Corollary 1. *In a tree-topology system where all nodes have identical buffer size and all links have the same bandwidth, the competitive factor of* Greedy *is* $O(\min(h, \alpha))$*, where* h *is the depth of the tree and* α *is the ratio between the most and the least valuable packets in the input.*

Proof: For the given system, we have that $D = hB_{\min}/R$ since all buffers have size $B_{\min}$ and all links have bandwidth R. Therefore, by Theorem 4, the competitive factor is at most $O(h)$. To see that the competitive factor is at most $O(\alpha)$, observe that Greedy outputs the maximal possible number of packets. ▮

4.2 An Upper Bound for Greedy Schedules on Trees

We now prove our main result, which is an upper bound on the competitive ratio of Greedy for tree topologies with arbitrary buffer sizes and link bandwidths. The result holds under the assumption that all packet values are either 1 or $\alpha > 1$.

We introduce the following key concept. Recall that the delay of a link e, denoted $d(e)$, is the size of its buffer divided by its bandwidth, and the delay of a path is the sum of its links' delays.

Definition 3. *Let* $e = (v, u)$ *be any link in a given tree topology, and suppose that* v *has children* $v_1, \ldots, v_k$*. The* height *of* e*, denoted* $h(e)$*, is the maximum path delay, over all paths starting at a leaf and ending at* u*. The* weakness *of* e*, denoted* $\lambda(e)$*, is defined to be* $\lambda(e) = \frac{h(e)}{d(e)}$.

Intuitively, $h(e)$ is just an upper bound on the number of time units that a packet can spend in the system before being sent over e. The significance of the notion of weakness of a link is made explicit in the following theorem.

Theorem 5. *The competitive ratio of* Greedy *for any given tree topology* $G = (V, E)$ *and two packet values is* $O(\max\{\lambda(e) : e \in E\})$.

Proof: Fix the input sequence. Consider the schedule produced by Greedy. We construct a set of time intervals called *overload intervals*, where each interval is associated with a link. The construction proceeds from the root link inductively as follows. Consider a link e, and suppose that all overload intervals were already defined for all links e' downstream from e. The set of overload intervals at e is defined as follows. For each time point t^* in which a high-value packet is dropped from Q_e, we define an overload interval $I = [t_s, t_f]$ such that

(1) $t^* \in I$.

(2) In each time step $t \in I$, $W(e)$ high value packets are sent over e.

(3) For any overload interval $I' = [t'_s, t'_f]$ of a downstream link e', we have that either $t_s > t'_f$ or $t_f < t'_s - d(e, e')$, where $d(e, e')$ is the sum of link delays on the path that starts at the endpoint of e and ends at the endpoint of e'.

(4) I is maximal.

Note that if a high value packet is dropped from a buffer Q_e by Greedy at time t, then Q_e is full of high value packets at time t, and hence $W(e)$ high value packets will be sent over e in each time step $t, t+1, \ldots, t+d(e)$. However, the overload interval containing t may be shorter (possibly empty), due to condition (3).

We now define a couple of notions regarding overload intervals. The *dominance* relation between overload intervals is defined as follows. If for an overload interval $I = [t_s, t_f]$ that occurs at link e there exists an overload interval $I' = [t'_s, t'_f]$ that occurs at a downstream link e' such that $t'_s = t_f + d(e, e') + 1$, we say that I is *dominated* by I'. We also define the notion of *full intervals*: an overload interval I that occurs at link e is said to be *full* if $|I| \geq d(e)$. Note that some non-full intervals may be not dominated.

We now proceed with the proof. For the sake of simplicity, we do not attempt to get the tightest possible constant factors. We partition the set of overload intervals so that in each part there is exactly one full interval, by mapping each overload interval I to a full interval denoted $P(I)$. Given an overload interval I, the mapping is done inductively, by constructing a sequence $I_0, \ldots, I_\ell$ of overload intervals such that $I = I_0$, $P(I) = I_\ell$, and only interval I_ℓ is full. Let I be any overload interval, and suppose it occurs at link e. We set $I_0 = I$, and let $e_0 = e$. Suppose that we have defined I_j already. If I_j is full, the sequence is complete. Otherwise, by definition of overload intervals, there must exist another interval I_{j+1} at a link e_{j+1} downstream from e_j that dominates I_j. If there is more than one interval dominating I_j, let I_{j+1} be the one that occurs at the lowest level. Note that the sequence must terminate since for all j, e_{j+1} is strictly downstream from e_j.

Let $\mathcal{F}$ denote the set of all full intervals. Let I be a full interval that occurs at link e. Define the set $\mathcal{P}(I) = \{I' : P(I') = I\}$. This set consists of overload intervals that occur at links in the subtree rooted by e. Define the *coverage* of I, denoted $\mathcal{C}(I)$, to be the following time window:

$$\mathcal{C}(I) = \left[\min\left\{t : t \in I' \text{ for } I' \in \mathcal{P}(I)\right\} - h(e) \;\; , \;\; \max\left\{t : t \in I' \text{ for } I' \in \mathcal{P}(I)\right\} + h(e)\right]$$

In words, $\mathcal{C}(I)$ starts $h(e)$ time units before the first interval starts in $\mathcal{P}(I)$, and ends $h(e)$ time units after the last interval ends in $\mathcal{P}(I)$. The key arguments of the proof are stated in the following lemmas.

Lemma 1. *For any full interval I that occurs at any link e, $|\mathcal{C}(I)| < |I| + 4h(e)$.*

Proof: Let I_0 be the interval that starts first in $\mathcal{P}(I)$, and let $I_1, \ldots, I_\ell$ be the sequence of intervals in $\mathcal{P}(I)$ such that I_{j+1} dominates I_j for all $0 \leq j < \ell$, and such that $I_\ell = I$. For each j, let $I_j = [t_j, t'_j]$, and suppose that I_j occurs at e_j. Note that I_ℓ is also the interval that ends last in $\mathcal{P}(I)$. Since for all $j < \ell$ we have that I_j is not full, and using the definition of the dominance relation, we have that

$$\begin{aligned} |\mathcal{C}(I)| - 2h(e) = t'_\ell - t_0 &= \sum_{j=0}^{\ell}(t'_j - t_j) + \sum_{j=1}^{\ell}(t_j - t'_{j-1}) \\ &< |I| + \sum_{j=0}^{\ell-1} d(e_j) + \sum_{j=1}^{\ell} d(e_{j-1}, e_j) \leq |I| + 2h(e) \ . \quad \blacksquare \end{aligned}$$

Lemma 2. *For each full interval I that occurs at a link e, the total number of high value packets that are ever sent by* Opt *from e and were dropped by* Greedy *during $\mathcal{C}(I)$ is at most $W(e) \cdot (|I| + 6h(e))$.*

Proof: As mentioned above, a packet that is dropped from any buffer upstream from e at time t can never be sent by any schedule outside the time window $[t - h(e), t + h(e)]$. The result therefore follows from Lemma 1. ∎

Lemma 3. *For each high-value packet p dropped by* Greedy *from a link e' at time t, there exists a full overload interval I that occurs in a link downstream from e' (possibly e' itself) such that $t \in \mathcal{C}(I)$.*

Proof: We proceed by the case analysis.

If $t \in I'$ for some full overload interval I' of e', we are done since $t \in \mathcal{C}(I')$.

If $t \in I'$ for some non-full overload interval of e' dominated by another overload interval I, we have that $t \in \mathcal{C}(P(I))$.

If $t \in I'$ for some non-full overload interval $I' = [t'_s, t'_f]$ of e' that is not dominated by any other overload interval then there exists an overload interval I'' that occurs in a link e'' downstream from e' such that $t'_s = t''_f + 1$ and hence $t \in \mathcal{C}(P(I''))$ because $t''_f + d(e') \geq t'_f$.

If t is not in any overload interval of e' then by the construction for an overload interval I'' that occurs in a link e'' downstream from e' we have that $t''_s - d(e', e'') \leq t \leq t''_f$, which implies that $t \in \mathcal{C}(P(I''))$.

∎

Lemma 4. *For each overload interval I,* Greedy *sends at least $|I| \cdot W(e)$ high value packets from e, and these packets are never dropped.*

Proof: The number of packets sent follows from the fact that when a high-value packet is dropped by Greedy from Q_e, the buffer is full of high value packets. The definition of overload intervals ensures that no high value packet during an overload interval is ever dropped, since if a packet that is sent over e at time t is dropped from a downstream buffer e' at time t', then $t' \leq t + d(e, e')$. ∎

We now conclude the proof of Theorem 5. Consider the set of all packets sent by Opt. Since the total number of packets sent by Greedy in a tree topology is maximal, it is sufficient to consider only the high-value packets. By Lemma 3, it is sufficient to consider only the time intervals $\{\mathcal{C}(I) : I \in \mathcal{F}\}$ since outside these intervals Greedy does as well as Opt. For each $I \in \mathcal{F}$ that occurs at a link e, we have by Lemma 4 that Greedy sends at least $|I| \cdot W(e)$ high value packets, whereas by Lemma 2 Opt sends at most $W(e) \cdot (|I| + 6h(e))$ high value packets. The theorem follows. ∎

References

1. W. Aiello, E. Kushilevitz, R. Ostrovsky, and A. Rosén. Dynamic routing on networks with fixed-size buffers. In *Proc. of the 14th ann. ACM-SIAM Symposium on Discrete Algorithms*, pages 771–780, Jan. 2003.

2. W. Aiello, Y. Mansour, S. Rajagopolan, and A. Rosen. Competitive queue policies for diffrentiated services. In *Proc. IEEE INFOCOM*, 2000.
3. Y. Azar and Y. Richter. Management of multi-queue switches in QoS networks. In *Proc. 33rd ACM STOC*, June 2003. To appear.
4. D. Black, S. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. Internet RFC 2475, December 1998.
5. A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial queuing theory. *J. ACM*, 48(1):13–38, 2001.
6. D. Clark and J. Wroclawski. An approach to service allocation in the Internet. Internet draft, 1997. Available from `diffserv.lcs.mit.edu`.
7. S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. on Networking*, 1(4):397–413, 1993.
8. E. H. Hahne, A. Kesselman, and Y. Mansour. Competitive buffer management for shared-memory switches. In *Proc. of the 2001 ACM Symposium on Parallel Algorithms and Architecture*, pages 53–58, 2001.
9. S. Keshav. *An engineering approach to computer networking: ATM networks, the Internet, and the telephone network*. Addison-Wesley Longman Publishing Co., Inc., 1997.
10. A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko. Buffer overflow management in QoS switches. In *Proc. 33rd ACM STOC*, pages 520–529, July 2001.
11. A. Kesselman and Y. Mansour. Loss-bounded analysis for differentiated services. *Journal of Algorithms*, Vol. 46, Issue 1, pages 79–95, January 2003.
12. A. Kesselman and Y. Mansour. Harmonic buffer management policy for shared memory switches. In *Proc. IEEE INFOCOM*, 2002.
13. M. A. Labrador and S. Banerjee. Packet dropping policies for ATM and IP networks. *IEEE Communications Surveys*, 2(3), 1999.
14. W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.
15. Z. Lotker and B. Patt-Shamir. Nearly optimal FIFO buffer management for DiffServ. In *Proc. 21st Ann. ACM Symp. on Principles of Distributed Computing*, pages 134–143, 2002.
16. Y. Mansour and B. Patt-Shamir. Greedy packet scheduling on shortest paths. *J. of Algorithms*, 14:449–465, 1993. A preliminary version appears in the *Proc. of 10th Annual Symp. on Principles of Distributed Computing, 1991*.
17. M. May, J.-C. Bolot, A. Jean-Marie, and C. Diot. Simple performance models of differentiated services for the Internet. In *Proc. IEEE INFOCOM*, 1998.
18. D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Comm. ACM*, 28(2):202–208, 1985.
19. The ATM Forum Technical Committee. Traffic management specification version 4.0, Apr. 1996. Available from `www.atmforum.com`.
20. A. Veres and M. Boda. The chaotic nature of TCP congestion control. In *Proc. IEEE INFOCOM*, pages 1715–1723, 2000.