SIAM J. COMPUT.
Vol. 33, No. 3, pp. 563–583

BUFFER OVERFLOW MANAGEMENT IN QoS SWITCHES*

ALEXANDER KESSELMAN†, ZVI LOTKER‡, YISHAY MANSOUR†,
BOAZ PATT-SHAMIR‡, BARUCH SCHIEBER§, AND MAXIM SVIRIDENKO§

Abstract. We consider two types of buffering policies that are used in network switches supporting Quality of Service (QoS). In the *FIFO* type, packets must be transmitted in the order in which they arrive; the constraint in this case is the limited buffer space. In the *bounded-delay* type, each packet has a maximum delay time by which it must be transmitted, or otherwise it is lost. We study the case of overloads resulting in packet loss. In our model, each packet has an intrinsic value, and the goal is to maximize the total value of transmitted packets.

Our main contribution is a thorough investigation of some natural greedy algorithms in various models. For the FIFO model we prove tight bounds on the competitive ratio of the greedy algorithm that discards packets with the lowest value when an overflow occurs. We also prove that the greedy algorithm that drops the earliest packets among all low-value packets is the best greedy algorithm. This algorithm can be as much as 1.5 times better than the tail-drop greedy policy, which drops the latest lowest-value packets.

In the bounded-delay model we show that the competitive ratio of any on-line algorithm for a uniform bounded-delay buffer is bounded away from 1, independent of the delay size. We analyze the greedy algorithm in the general case and in three special cases: delay bound 2, link bandwidth 1, and only two possible packet values.

Finally, we consider the off-line scenario. We give efficient optimal algorithms and study the relation between the bounded-delay and FIFO models in this case.

Key words. buffer overflows, competitive analysis, Quality of Service, FIFO scheduling, deadline scheduling

AMS subject classifications. 90B35, 68M20, 68Q25, 68W01

DOI. 10.1137/S0097539701399666

1. Introduction.

Unlike the "best effort" service provided by the Internet today, next-generation networks will support guaranteed Quality of Service (QoS) features. In order for the network to support QoS each network switch must be able to guarantee a certain level of QoS in some predetermined parameters of interest, including packet loss probability, queuing delay, jitter, and others.

In this work, we consider models based on the IP environment. Implementing QoS in an IP environment is receiving growing attention, since it is widely recognized that future networks would most likely be IP based. There have been a few proposals that address the integration of QoS in the IP framework, and our models are based on some of these proposals, specifically in the general area of providing *differentiated network services* in an IP environment. For example, different customers may get different levels of service, which might depend on the price they pay for the service.

*Received by the editors December 12, 2001; accepted for publication (in revised form) December 11, 2003; published electronically March 23, 2004. An extended abstract of this paper appeared in *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing* (STOC '01), Crete, Greece, 2001.

http://www.siam.org/journals/sicomp/33-3/39966.html

†Department of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel (alx@math.tau.ac.il, mansour@math.tau.ac.il). The research of these authors was partially supported by a grant from the Israel Ministry of Science and Technology.

‡Department of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel (zvilo@eng.tau.ac.il, boaz@eng.tau.ac.il). The research of these authors was partially supported by a grant from the Israel Ministry of Science and Technology.

§IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598 (sbar@us.ibm.com, sviri@us.ibm.com).

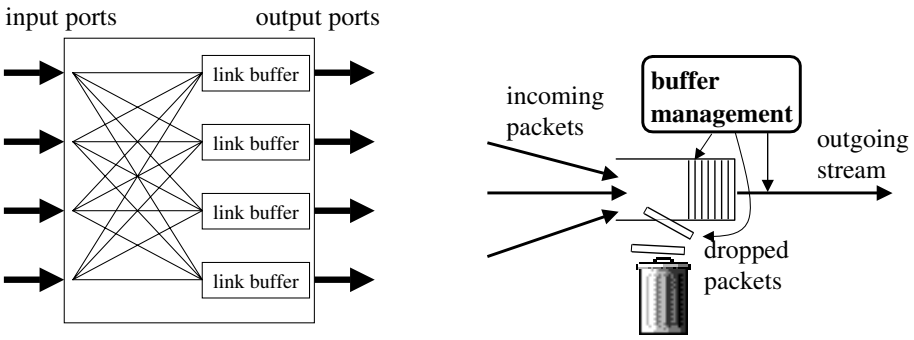

FIG. 1.1. *Schematic representation of the model. Left: general switch structure. Right: output port structure. Packets are placed in the buffer, and the buffer management algorithm controls which packet will be discarded and which will be transmitted.*

One way of guaranteeing QoS is by committing resources to each admitted connection, so that each connection has its dedicated resource set that will guarantee its required level of service regardless of all other connections. This conservative policy (implemented in the specification of CBR traffic in asynchronous transfer mode (ATM) networks [23]) might be extremely wasteful since network traffic tends to be bursty. Specifically, this policy does not take into consideration the fact that *usually*, the worst-case resource requirements of different connections do not occur simultaneously. Recognizing this phenomenon, most modern QoS networks allow some "overbooking," employing the policy popularly known as *statistical multiplexing*. While statistical multiplexing tends to be very cost-effective, it requires satisfactory solutions to the unavoidable events of overload. In this paper we consider such scenarios in the context of *buffering*. The basic situation we consider is an output port of a network switch with the following activities (see Figure 1.1). At each time step, an arbitrary set of packets arrives, but only a fixed number of packets can be transmitted. The buffer management algorithm controls which packets are admitted to the buffer, which are discarded, and which are transmitted at each step.

We consider two types of buffer models. In the *FIFO model*, packets can never be sent out of order: formally, for any two packets p, p' sent at times t, t', respectively, we have that if $t' > t$, then packet p did not arrive after packet p'. The main constraint in this classical model is that the buffer size is fixed, so when too many packets arrive, *buffer overflow* occurs and some packets must be discarded. In most implementations the discard policy is the natural *tail-drop* policy, in which the latest packets are discarded.

The second model we consider is the *bounded delay model*. This model is relatively new, and is warranted by networks that guarantee the QoS parameter of end-to-end delay. Specifically, in the bounded-delay model each packet arrives with a prescribed *allowed delay* time. A packet must be transmitted within its allowed delay time or else it is lost. In this model, the buffer management policy can reorder packets. We consider two variants of the model. In the *uniform* bounded delay model, the switch has a single fixed bound on the delay of all packets, and in the *variable* bounded delay model, the switch may have a different delay bound for each packet.

The focus of our paper is the following simple refinement of the models described above. Each packet arrives with its intrinsic *value*, and the goal of the buffer management algorithm is to discard packets so as to maximize the total value of packets

transmitted. All we assume about the value of the packets is that it is additive; that is, the value of a set of packets is the sum of the values of packets in the set.

In this paper we present a thorough investigation of the natural greedy algorithms in the various models. In the FIFO model, the greedy algorithm discards the lowest-value packets whenever an overflow occurs, with ties broken arbitrarily. We prove a tight bound of $2 - \frac{W}{B+W}$ on the competitive factor of this algorithm, where B is the buffer size and W is the link bandwidth. For the case where the ratio of the maximum to minimum value is bounded by some $\alpha \geq 1$, we prove a tight bound of $2 - \frac{2}{\alpha+1}$ on the competitive factor. The proof of the upper bound is quite involved. We then consider different variants of the greedy algorithms, since the greedy policy does not specify which packet to drop in case there is more than one packet with the lowest value. Specifically, we consider the head-drop greedy policy, which drops the earliest lowest-value packets. We show that for *any* input sequence the head-drop greedy policy achieves equal or better value than any other greedy policy. This is somewhat surprising, since most implementations use the tail-drop policy. Furthermore, we show that the ratio of the value served by the head-drop greedy policy to the value served by the tail-drop policy can be as high as 3/2 in some cases. We also prove a lower bound on the competitive ratio of any on-line algorithm in the FIFO model.

For the bounded delay model we have the following results. First, we show that the competitive ratio of any on-line algorithm for a uniform bounded delay buffer is bounded away from 1, independent of the delay size. This holds even if all packets have an arbitrarily long allowed delay. Next, we consider the simple greedy algorithm, which in this model always sends the packet with the highest value. We prove that the competitive ratio of this algorithm is exactly 2. In the common case when there are only two possible values of packets (i.e., "cheap" and "expensive" packets) the competitive factor of the greedy algorithm is exactly $1 + 1/\alpha$, where $\alpha \geq 1$ is the ratio of the expensive value to the cheap value. We then consider the special case where the delay is less than 2, namely, a packet, if not dropped, is sent when it arrives or in the next time step. We show that in this case, the bound of 2 can be improved: we give algorithms that achieve a competitive ratio of 1.618 for the variable delay model. We also prove lower bounds of 1.11 on the competitive ratio for the uniform delay model and 1.17 for the bounded delay model. Better bounds are presented for the case where the bandwidth link is 1. We show that in this variant slight modifications of the greedy algorithm guarantee better performance.

Lastly, we consider the off-line case. We prove that the overflow management problem has matroid structure in both buffer models, and hence admits efficient optimal off-line algorithms.

Related work. There is a myriad of research papers about packet drop policies in communication networks; see, for example, the survey of [16] and references therein. Some of the drop mechanisms, such as random early detection (RED) [11], are designed to signal congestion to the sending end. The approach abstracted in our model, where each packet has an intrinsic value and the goal is to maximize the total throughput value, is implicit in the recent DiffServ model [8, 9] and ATM [23]. The bounded delay model is an abstraction of the model described in [13].

There has been work on analyzing various aspects of the model using classical queuing theory and assuming Poisson arrivals [21]. The Poisson arrival model has been seriously undermined by the discovery of the heavy tail nature of traffic [18] and the chaotic nature of TCP [24]. In this work we use competitive analysis, which makes no probabilistic assumptions.

The work of [2] concentrated on the case where one cannot discard a packet already in the buffer. The authors give tight bounds on the competitive factor of various algorithms for the special case where there are only two different weights. In [20], the question of video smoothing is studied. One of the results in that paper, which we improve here, is an upper bound of 4 on the competitive ratio of the greedy algorithm for the FIFO model. The work of [14] deals with the loss-competitive analysis rather than the throughput-competitive analysis. The authors show how to translate the loss guarantee to the throughput guarantee and obtain an almost tight upper bound for the case of two packet values. A similar result is presented in this paper for the bounded delay model. The work of [3] studies bandwidth allocation from the competitive analysis viewpoint, disregarding buffer overflows.

The bounded delay model can be viewed as a scheduling problem. In this problem we are given parallel machines (whose number is the link bandwidth) and jobs with release time and deadline that correspond to the packets arriving to the pool. The goal is to maximize the throughput, defined as the sum of the weights of the jobs that terminate by their deadline. In our case we also have the additional constraint that all jobs have the same processing time. The off-line variant of this scheduling problem denoted by $P \,|\, r_i; p_i = p \,|\, \sum w_i(1 - U_i)$ in the standard scheduling notation can be solved in polynomial time using maximum matching. To the best of our knowledge the on-line variant of this problem has not been considered elsewhere. The more general off-line problem, where the processing time is not fixed, is NP-hard. Recently, approximation algorithms for this problem were considered in [5, 4, 22]. The more general on-line problem, where the processing time is not fixed, was considered in [19], where the authors prove competitive ratios that are substantially larger than those in our case. Slightly better ratios can be achieved for the case where processing time is not fixed if preemption is allowed. This case was considered in [6, 7, 15]. For the unweighted version of this problem, i.e., when the goal is to maximize the number of jobs completed by their deadline (which is trivial if all processing times are the same), [6] showed a tight competitive factor of 4. For the weighted version the tight bound is $\sqrt{1 + k^2}$, where k is the ratio of the maximum value density to the minimum value density of a job [15].

Paper organization. The remainder of this paper is organized as follows. In section 2 we define the models and some notation. In section 3 we consider the FIFO model and in section 4 we consider the bounded delay models. Finally in section 5 we discuss off-line algorithms.

2. Model and notation. In this section we formalize the model and the notation we use. We consider two main models: the FIFO model and the bounded-delay switch model. First, we list the assumptions and define quantities that are common to both models.

We assume that time is discrete. Fix an algorithm A. At each time step t, there is a set of *packets* $Q_A(t)$ stored at the *buffer* (initially empty). Each packet p has a positive real *value* denoted by $v(p)$. At time t, a set of packets $A(t)$ arrives. A set of packets from $Q_A(t) \cup A(t)$, denoted by $S_A(t)$, is *transmitted*. We denote by $S_A = \cup_t S_A(t)$ the set of packets transmitted by the algorithm A and by $S_A^{val} = \{p : p \in S_A, v(p) = val\}$ the set of packets of value val sent by A. Note that we consider the so-called cut-through model in which a packet may arrive and be transmitted at the same step. A subset of $Q_A(t) \cup A(t) \setminus S_A(t)$, denoted by $D_A(t)$, is *dropped*. The set of packets in the buffer at time $t + 1$ is $Q_A(t + 1) = Q_A(t) \cup A(t) \setminus (D_A(t) \cup S_A(t))$. We omit subscripts when no confusion arises.

The input is the packet arrival function $A(\cdot)$ and the packet value function $v(\cdot)$. Packets may have other attributes as well, depending on the specific model. The buffer management algorithm has to decide at each step which of the packets to drop and which to transmit, satisfying some constraints specified below. For a given input, the value *served* by the algorithm is the sum of the values of all packets transmitted by the algorithm. For a set P of packets define $v(A)$ to be the total value of the packets in the set. In this notation the value served by algorithm A is $\sum_t v(S_A(t))$.

The sequence of packets transmitted by the algorithm must obey certain restrictions.

Output link bandwidth. We assume that there is an integer number W called the *link bandwidth* such that the algorithm cannot transmit more than W packets in a single time unit; i.e., $|S(t)| \leq W$ for all t. For simplicity, we usually assume that $W = 1$ unless stated otherwise.

FIFO buffers. In the FIFO model there are two additional constraints. First, the sequence of transmitted packets has to be a subsequence of the arriving packets. That is, if a packet p is transmitted after packet p', then p could not have arrived before p'. Second, the number of packets in the buffer is bounded by the *buffer size* parameter, denoted by B. Formally, the constraint is that for all times t, $|Q(t)| \leq B$. (Note that our model allows for a larger number of packets in the transient period of each step, starting with packets' arrival and ending with packets' drop and transmission.)

Bounded-delay buffers. In this model we assume that packets have another attribute: for each packet p there is the *slack time* of p, denoted by $sl(p)$. The requirement is that a packet $p \in A(t)$ must be either transmitted or dropped before time $t + sl(p)$ (i.e., in one of the steps $t, t+1, \ldots, t+sl(p)-1$). The time $t + sl(p)$ is also called the *deadline* of p, denoted by $dl(p)$. We emphasize that in this model there is no explicit bound on the size of the buffer, and that packets may be reordered.

Uniform and variable bounded-delay buffers. One case of special interest within the bounded-delay buffers model is when the slack of all packets is equal to some known parameter δ. We call this model *δ-uniform bounded-delay buffers.* If all packet slacks are only *bounded* by some number δ, we say that the buffer is *δ-variable bounded-delay.*

On-line and off-line algorithms. We call an algorithm *on-line* if for all time steps t, it has to decide which packets to transmit and which to drop at time t without any knowledge of the packets arriving at steps $t' > t$. If future packet arrival is known, the algorithm is called *off-line.* We denote the optimal policy by OPT and the set of packets sent by the optimal policy by S_O. The *competitive ratio* (or *competitive factor*) of an algorithm A is an upper bound, over all input sequences P, on the ratio of the maximal value that can be transmitted by OPT to the value that is transmitted by A, that is, $\max_P(\frac{v(OPT)}{v(A)})$. Note that since we deal with a maximization problem this ratio will always be at least 1. In what follows, we denote by S_O and S_G the sets of packets transmitted by OPT and by the on-line algorithm considered, respectively, if it is not explicitly stated otherwise.

3. The FIFO model. In this section we consider the FIFO model. Recall that in this model a buffer of size B is used to store the incoming packets. Packets have to be transmitted in the order in which they arrive. Each packet has a value associated with it and the goal is to maximize the value of the transmitted packets.

First, we prove a lower bound on the competitive ratio of any on-line algorithm in the FIFO model. This proof improves the 1.25 bound proved in [20].

THEOREM 3.1. *The competitive ratio of any deterministic on-line algorithm in the FIFO model is at least* 1.281.

Proof. Assume that the link rate is 1 packet per time unit. Fix an on-line algorithm A, and let C_A denote its competitive ratio. We consider two scenarios. In both scenarios, B packets of value 1 arrive at time $t = 0$. In each of the next time steps a single packet of value $\alpha > 1$ arrives. This continues until we reach time B or until A sends an α value packet (which means that A has dropped all the remaining packets of value 1 from the buffer). Let t be that time.

In the first scenario, the input stream ends at time t. The benefit of A is $1 \cdot t + \alpha \cdot t$, while the off-line benefit is $1 \cdot B + \alpha \cdot t$.

In the second scenario, at time $t+1$ a burst of B packets, each of value α, arrive. The off-line benefit in this case is $\alpha(B + t)$, while the benefit of A is $1 \cdot t + \alpha \cdot B$.

Thus, the competitive ratio of A is

$$C_A \geq \max\left(\frac{B + \alpha t}{t + \alpha t}, \frac{\alpha(B + t)}{t + \alpha B}\right). \tag{3.1}$$

In our model, the adversary first chooses α so as to maximize C_A, then the algorithm chooses t (possibly depending on α) so as to minimize C_A, and then the competitive ratio is computed from (3.1). We work backwards, assuming first that α is a parameter. It is easy to see that (3.1) is minimized when

$$\frac{B + \alpha t}{t + \alpha t} = \frac{\alpha(B + t)}{t + \alpha B}. \tag{3.2}$$

Solving (3.2) for t, we get that its only nonnegative root is

$$t_0(B, \alpha) = B\frac{\sqrt{(\alpha - 1)^2 + 4\alpha^3} - \alpha + 1}{2\alpha^2}.$$

Define $t_0'(\alpha) = t_0(B, \alpha)/B$. With this notation, it follows from (3.1) that the competitive ratio of A satisfies

$$C_A \geq \frac{1 + \alpha t_0'(\alpha)}{t_0'(\alpha) + \alpha t_0'(\alpha)}. \tag{3.3}$$

Using numerical methods, we find that the right-hand side of (3.3), viewed as a function of α, is maximized when $\alpha \approx 4.01545$. Substituting in (3.1), we obtain that the competitive ratio of A cannot be better than 1.28197. □

3.1. Tight analysis of the greedy algorithm. We consider the greedy algorithm for output link bandwidth W, for any integer $W > 0$. In this algorithm no packets are dropped in time step t in which $|Q(t)|+|A(t)| \leq B+W$. If $|Q(t) \cup A(t)| = k > 0$, then the earliest $\min(W, k)$ packets are transmitted and the rest are stored in the buffer. If $|Q(t)| + |A(t)| = k > B + W$, the $k - B - W$ packets with the lowest value are dropped, ties broken arbitrarily. Among the remaining $B + W$ packets, the W earliest packets are transmitted and the rest are stored in the buffer.

For the sake of simplicity, we assume that B is a multiple of W.[1] Let $B_W = B/W$ denote the number of time steps that is needed to transmit the whole buffer. We first show that the competitive ratio of the greedy algorithm is no worse than 2. Later, we improve this bound and show that the improved bounds are tight.

[1] If W does not divide B, one can think of "fractions" of time steps in which sets of less than W packets are transmitted. This makes the analysis somewhat cumbersome, but the results of this section carry over to the general case as well.

THEOREM 3.2. *The competitive ratio of the greedy algorithm is at most* 2 *for any output link bandwidth* W.

Proof. Consider a sequence of packets. We need to show that $v(OPT) \leq 2v(Greedy)$. Let $D_G^O(t) = S_O \cap D_G(t)$ denote the set of packets that were dropped by the greedy algorithm at time t but were transmitted by the optimal algorithm. Let D_G^O be the union of all these sets. We show a mapping from the packets in D_G^O to packets in S_G with the following properties: (i) a packet from D_G^O is mapped to a packet in S_G with at least the same value, (ii) at most one packet from D_G^O is mapped to any packet in $S_G \cap S_O$, and (iii) at most two packets from D_G^O are mapped to any packet in $S_G \setminus S_O$. Clearly, the existence of the mapping implies the theorem.

We construct the mapping iteratively. At each iteration we consider a packet from D_G^O and map it to a packet in S_G. The packets in D_G^O are considered in the order of their drop time. Suppose that the current packet to be mapped is $p \in D_G^O(t)$. This means that all the packets in $D_G^O(s)$, for $s < t$, and maybe some of the packets in $D_G^O(t)$ have been mapped already. Given the partially defined mapping, define a set of "available" packets in S_G as follows. A packet in $q \in S_G \cap S_O$ is available if so far no packet is mapped to q. A packet in $q \in S_G \setminus S_O$ is available if so far at most one packet is mapped to q. The packet p is mapped to the earliest available packet that was transmitted by the greedy algorithm at or after time t.

Clearly, the mapping defined above satisfies the second and third properties. It remains to be shown is that it satisfies the first property as well. Consider the mapping process step by step, where in each step a single packet is mapped. We prove that the first property holds by induction on the steps of the mapping process. For the basis of the induction note that if $p \in D_G^O(t)$ is the first packet to be dropped, then it is mapped to the packet transmitted by the greedy algorithm at time t. Clearly, the value of this packet is at least $v(p)$.

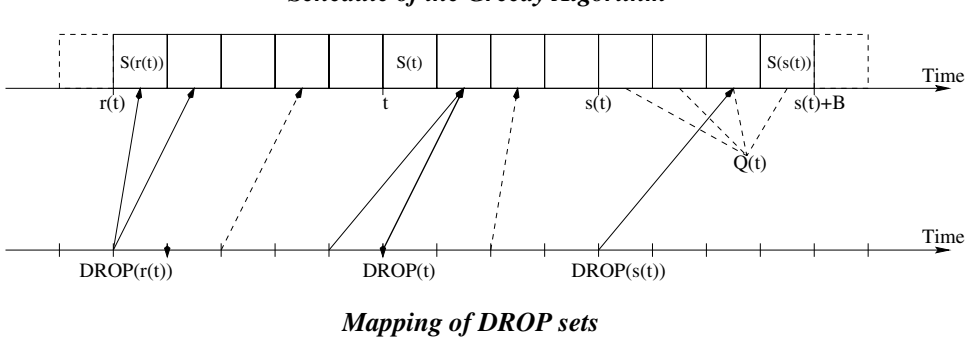

FIG. 3.1. *An example of the mapping from* D_G^O *to* S_G.

Suppose that the first property holds for all packets mapped before $p \in D_G^O(t)$. We show that it also holds for p. Let $s(t)$ be the earliest time at or after t such that the buffer maintained by the greedy algorithm at time $s(t)$ is full and all the packets in the buffer at time $s(t)$ are transmitted by the greedy algorithm (see Figure 3.1). Observe that since the buffer is full at time t, $s(t)$ is well defined.

CLAIM 1. *The value of each of the packets transmitted by Greedy at times* $t, \ldots, s(t) + B_W$ *is at least* $v(p)$.

Proof. Consider the greedy algorithm in the time interval $t, \ldots, s(t) + B_W$. Construct a sequence of time steps inductively as follows:

- $t_0 = t$.
- For $t_i < s(t)$, define t_{i+1} to be the first step after t_i in which a packet that was in the buffer at time t_i is dropped.

Let t_k be the last time step in this sequence. Note that $t_k = s(t_i)$ for all $0 \leq i \leq k$ by definition. We prove the claim by induction on k. For the case $k = 0$ we have that $s(t) = t$, and since all packets in the buffer at time $s(t)$ are transmitted in steps $s(t), \ldots, s(t) + B_W$, and all these packets have value at least $v(p)$ by the greedy rule, we are done. For the induction step, assume that the claim holds for k and consider $k + 1$. Let q be a packet that was in the buffer at time t_0 and dropped at time t_1. By the FIFO ordering, all packets transmitted in steps $t_0, \ldots, t_1$ were in the buffer at time t_0. By the greedy rule we have that (i) all packets transmitted at times $t_0, \ldots, t_1$ have value at least $v(p)$, and (ii) $v(q) \geq v(p)$. The claim follows by combining fact (i) with the induction hypothesis applied to q and time t_1. □

To prove that the first property holds for p, we show that it is mapped to one of the packets transmitted at times $t, \ldots, s(t) + B_W$. For this we need to show that at least one of the packets transmitted at times $t, \ldots, s(t) + B_W$ remains available at the time p is mapped. The "number of availabilities" at the time packet p is mapped is defined as the maximum number of packets that can still be mapped to the packets transmitted at times $t, \ldots, s(t) + B_W$. Specifically, every available packet (at the time p is mapped) among the packets transmitted at times $t, \ldots, s(t) + B_W$ that is also in S_O contributes 1 to the number of availabilities. Every available packet (at the time p is mapped) among the packets transmitted at times $t, \ldots, s(t) + B_W$ that is not in S_O contributes 2 to the number of availabilities if it is not mapped and contributes 1 if one packet has been mapped to it already.

Claim 2. *The number of availabilities before any of the packets dropped at time t is mapped is at least $\sum_{i=t}^{s(t)} |D_G^O(i)|$.*

Proof. Let $r(t)$ be the latest time before or at t such that no packets dropped by the greedy algorithm at time $r(t) - 1$ or earlier are mapped to packets transmitted at time $r(t)$ or later. If no such $r(t)$ exists, then define $r(t) = 0$ (the first step).

Intuitively, we start by showing that each interval $r(t), \ldots, s(t) + B_W$ is independent of the other intervals. Specifically, we claim that the number of availabilities before any of the packets dropped at time $r(t)$ and later is mapped is at least $\sum_{i=r(t)}^{s(t)} |D_G^O(i)|$. To see that, let $C(t)$ denote the set of packets available to the greedy algorithm at times $r(t), \ldots, s(t)$ that were transmitted by the optimal algorithm. In other words, $C(t)$ is the subset of packets transmitted by the optimal algorithm that were considered (stored at the beginning of the interval or arrived during the interval) by the greedy algorithm at times $r(t), \ldots, s(t)$. Note that $|C(t)| \leq W \cdot (s(t) - r(t) + 2B_W + 1)$, because packets in $C(t)$ cannot be transmitted by the optimal algorithm in time steps other than $r(t) - B_W, \ldots, s(t) + B_W$.

Next, let $x(t)$ be the number of packets of $C(t)$ that are transmitted by the greedy algorithm at times $r(t), \ldots, s(t) + B_W$. Observe that no packet from S_O that arrives at times $s(t), \ldots, s(t) + B_W$ is transmitted by the greedy algorithm at times $s(t), \ldots, s(t)+B_W$. This is because by definition, all packets stored by Greedy at time $s(t)$ are transmitted, a process that lasts B_W time units. It follows that exactly $x(t)$ packets from $C(t)$ are transmitted by the greedy algorithm at times $r(t), \ldots, s(t)+B_W$. This implies that the number of availabilities before any of the packets dropped at time $r(t)$ and later is mapped is at least $2W \cdot (s(t) + B_W - r(t) + 1) - x(t)$. This is true since by definition of $r(t)$, none of the packets that are dropped before time $r(t)$

is mapped to packets transmitted at times $r(t), \ldots, s(t) + B_W$. It follows that

$$\sum_{i=r(t)}^{s(t)} |D_G^O(i)| \leq W(s(t) - r(t) + 2B_W + 1) - x(t)$$

$$\leq 2W(s(t) + B_W - r(t) + 1) - x(t). \tag{3.4}$$

We can now prove the claim. Since by (3.4) the maximum number of packets that can be mapped to the packets transmitted at times $r(t), \ldots, s(t) + B_W$ is at least $\sum_{i=r(t)}^{s(t)} |D_G^O(i)|$, we have, by the definition of $r(t)$, that the maximum number of packets that can be mapped to the packets transmitted at times $r(t), \ldots, t-1$ is strictly less than $\sum_{i=r(t)}^{t-1} |D_G^O(i)|$. We conclude that the number of availabilities before the packets dropped at t are mapped is at least $\sum_{i=t}^{s(t)} |D_G^O(i)|$. □

The theorem follows directly from Claims 1 and 2. □

We now refine the analysis above to get tighter bounds. Let α be the ratio of the largest value of a packet to the smallest value of a packet (clearly we can assume without loss of generality that all packets have positive values).

THEOREM 3.3. *The competitive ratio of the greedy algorithm is at most* $2(1-\frac{1}{\alpha+1})$ *for any output link bandwidth* W*, where* $\alpha \stackrel{\text{def}}{=} \frac{\max_p\{v_p\}}{\min_p\{v_p\}}$.

Proof. Given the mapping defined above, partition S_G into three subsets as follows.

- G_1 is the set of packets in S_G that are not in S_O and that are not the image of (i.e., not mapped to by) any packet in D_G^O.
- G_2 is the set of packets in S_G that are unavailable after the mapping is done, namely, all packets in $S_G \cap S_O$ that are the image of one packet in D_G^O, and all packets in $S_G \setminus S_O$ that are the image of two packets in D_G^O.
- $G_3 = S_G \setminus (G_1 \cup G_2)$, that is, the packets in $S_G \cap S_O$ that are not the image of any packet in D_G^O, and the packets in $S_G \setminus S_O$ that are the image of one packet in D_G^O.

We associate two packets from S_O with each packet in $q \in G_2$ as follows. If $q \in G_2 \cap S_O$, then the two packets are q itself and the packet mapped to q. If $q \in G_2 \setminus S_O$, then the two packets are the two packets mapped to q. Similarly, we associate a packet from S_O with each packet in $q \in G_3$ as follows. If $q \in G_3 \cap S_O$, then this packet is q. If $q \in G_3 \setminus S_O$, then this packet is the packet mapped to q. Note that this way we associate every packet in S_O with some packet in S_G. Note that $v(q)$ is always at least the value of each of its associated packets, and the fact that no packet is associated with more than two packets implies the bound on the competitive ratio. We are able to improve the bound of 2 on this ratio using the fact that no packet is associated with packets in G_1.

Note that $|S_G| \geq |S_O|$. It follows that $|G_1| \geq |G_2|$, and thus we can match any packet $q \in G_2$ with a mate $p \in G_1$. We move a $\frac{1}{\alpha+1}$ "fraction" of the value of the packets associated with q and associate it with p. Note that after the move the total value associated with q is no more than $2(1-\frac{1}{\alpha+1})v(q)$. Since $v(q) \leq \alpha v(p)$, the total value associated with p is no more than

$$2\frac{1}{\alpha+1}v(q) \leq 2\frac{\alpha}{\alpha+1}v(p) = 2\left(1 - \frac{1}{\alpha+1}\right)v(p). \qquad \square$$

THEOREM 3.4. *The competitive ratio of the greedy algorithm is at most* $2-\frac{W}{B+W}$ *for any output link bandwidth* W.

Proof. Modify the mapping defined in the proof of Theorem 3.2 as follows. For each time t such that $D_G^O(t) \neq \emptyset$, decrease by W the number of availabilities contributed by the packets transmitted by the greedy algorithm at time t. Specifically, if $p \in S_O$, then no packet is mapped to p and if $p \notin S_O$, at most one packet is mapped to p.

We need to show that Claim 2 still holds. For this, it suffices to show that the number of availabilities before any of the packets dropped at time $r(t)$ and later is mapped is at least $\sum_{i=r(t)}^{s(t)} |D_G^O(i)|$. Notice that the number of availabilities is decreased by at most $W(s(t) - r(t) + 1)$. Claim 2 follows for the modified mapping since

$$\begin{aligned}\sum_{i=r(t)}^{s(t)} |D_G^O(i)| &\leq W(s(t) - r(t) + 2B_W + 1) - x(t) \\ &\leq 2W(s(t) + B_W - r(t) + 1) - x(t) - W(s(t) - r(t) + 1).\end{aligned}$$

Notice that the value of the packet transmitted at time t is at least the minimal value of the packets in $D_G^O(t)$. We "shift" a $W/(|D_G^O(t)| + W)$ fraction of the value of each of the packets in $D_G^O(t)$ from the packet to which it has been mapped originally and associate it to one of the packets transmitted by the greedy algorithm at time t. Since $|D_G^O(t)| \leq B$ and $|S_G(t)| = W$, we get that each packet transmitted by the greedy algorithm is associated with at most $1 + \frac{B}{B+W} = 2 - \frac{W}{B+W}$ packets of at most the same value from S_O. □

Finally, we observe that the last two bounds are tight. Consider the following scenario. At the first time step, $B + W$ packets of value 1 arrive; W are transmitted and the rest are kept in the buffer. Then, at each time $t \in \{1, \ldots, B_W\}$, W packets of value $\alpha \geq 1$ arrive and are kept in the buffer. At time $B_W + 1$ the buffer contains B packets of value α; at this time another $B + W$ packets of value α arrive. The greedy algorithm transmits only $B + W$ of these packets, while the optimal algorithm transmits $2B + W$ of value α and only W of value 1. We get that the competitive ratio for this scenario is

$$\frac{(2B + W)\alpha + W}{(B + W)(\alpha + 1)} = 2 - \frac{\alpha(W + 1) + 2B}{(B + W)(\alpha + 1)}.$$

Letting α go to infinity we get the ratio $2 - \frac{W}{B+W}$, and letting B go to infinity we get the ratio $2 - \frac{2}{\alpha+1}$.

3.2. The best greedy algorithm. The greedy algorithm is underspecified: when there are several packets with the same low value, it is not specified which of them is discarded by the greedy algorithm. It is easy to see that for any link bandwidth, the number of packets transmitted is the same for all variants of the greedy algorithm. However, is there a difference between the variants of the greedy algorithm in terms of weighted throughput? In this section we show that, perhaps surprisingly, it is always better to discard the earliest packets, i.e., the packets which spent the most time in the buffer. We call this policy *head-drop*, as the algorithm prefers to drop packets from the head of the buffer. Head-drop is in contrast to the common practice of *tail-drop*, where the newest packets are discarded. We show that there exist scenarios in which tail-drop results in significantly more losses than head-drop. We remark that the head-drop policy [17] enjoys additional advantages in the TCP/IP environment, namely, it helps the congestion avoidance mechanism.

The following theorem proves that the head-drop greedy algorithm is, in a certain sense, the best greedy algorithm.

THEOREM 3.5. *Let G be any greedy algorithm, and let GH be the greedy head-drop algorithm. For any input sequence, the total value transmitted by GH is at least the total value transmitted by G.*

Proof. Fix an input sequence. First, observe that the number of packets in the buffer of G and the number in the buffer of GH at time t are equal for all times t. (This follows from an easy induction on time, which shows that for any given arrival sequence and link bandwidth, the size of the queue in each step is the same for all algorithms that drop packets only when an overflow occurs.) To prove the theorem, we define, for each time step t, a 1-1 mapping from $Q_{GH}(t)$ to $Q_G(t)$ such that each packet $p \in Q_{GH}(t)$ is mapped to a packet $q \in Q_G(t)$ with at least the same value; i.e., $v(p) \leq v(q)$.

We claim that the existence of these mappings implies the theorem as follows. For a packet $p \in Q(t)$ define the *rank* of p to be its rank in the sequence of packets in $Q(t)$ ordered in ascending values. The existence of the mappings implies that for each time step t, the value of the packet ranked i in $Q_{GH}(t)$ is at most the value of the packet ranked i in $Q_G(t)$. Since $|D_{GH}(t)| = |D_G(t)|$, and since in any greedy algorithm $D(t)$ consists of the lowest ranked packets in $Q(t)$, it follows that $v(D_{GH}(t)) \leq v(D_G(t))$.

The value served by an algorithm is $\sum_t v(S(t))$; hence we get

$$\begin{aligned}\sum_t v(S_{GH}(t)) &= \sum_t v(A(t)) - \sum_t v(D_{GH}(t)) \\ &\geq \sum_t v(A(t)) - \sum_t v(D_G(t)) \\ &= \sum_t v(S_G(t)).\end{aligned}$$

All that remains is to prove the existence of the mappings. Fix a time step t and consider $Q_G(t)$ and $Q_{GH}(t)$. To construct the mapping, we use the natural concept of "height" of a packet in a buffer: For an algorithm A and a packet $p \in Q_A(t)$, the *height* of p, denoted by $h_A(p,t)$, is 1 plus the number of packets that have to be either transmitted or dropped before p can be transmitted. In other words, $h_A(p,t)$ is the rank of p in the sequence of packets in $Q_A(t)$ ordered by arrival time. We also say that a packet $p \in Q_A(t)$ is said to be *below* (or *above*) a packet $p' \in Q_A(t)$ if $h_A(p,t) < h_A(p',t)$ (or, respectively, $h_A(p,t) > h_A(p',t)$).

We now define the mapping from $Q_{GH}(t)$ to $Q_G(t)$. Each packet in $Q_{GH}(t) \cap Q_G(t)$ is mapped to itself. To complete the mapping, consider the packets in $Q_{GH}(t) \setminus Q_G(t)$ in ascending order of height. Map each such packet p to the packet with the lowest height in $Q_G(t) \setminus Q_{GH}(t)$ that has not been mapped yet. It is not difficult to see that this mapping is indeed 1-1. We need to show that each packet in $Q_{GH}(t) \setminus Q_G(t)$ is mapped to a packet in $Q_G(t) \setminus Q_{GH}(t)$ of at least the same value. For this we prove the following two lemmas.

LEMMA 3.6. *Let $p \in Q_{GH}(t) \setminus Q_G(t)$ for some time t. Then $v(p) \leq v(q)$, for all $q \in Q_G(t)$ satisfying $h_G(q,t) \leq h_{GH}(p,t)$.*

Proof. Let t_0 be the time in which G dropped p (and hence $p \notin Q_G(t_0+1)$). We prove the lemma by induction on $t - t_0$. For the base case $t = t_0 + 1$, we have by the greedy rule that $v(p) \leq v(p')$ for any $p' \in Q_G(t)$. For the inductive step, let $t > t_0+1$. First, note that p arrived before step $t-1$, since otherwise it must have been dropped at step $t-1$, contradicting the assumption that $t > t_0 + 1$. It follows that

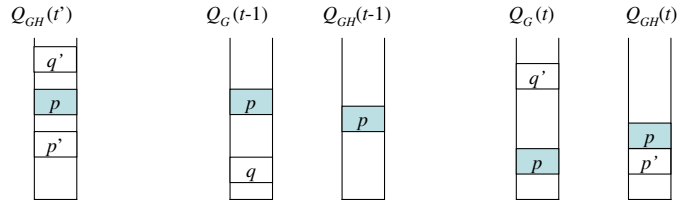

FIG. 3.2. *Scenario considered in the proof of Lemma* 3.7. $p \notin Q_G(t)$, $q' \notin Q_{GH}(t)$, *and* $q \notin D_G(t-1)$.

$h_{GH}(p, t-1)$ is well defined. Consider a packet $p' \in Q_G(t)$ with $h_G(p', t) \leq h_{GH}(p, t)$. If $h_G(p', t-1)$ is defined and $h_G(p', t-1) \leq h_{GH}(p, t-1)$, we are done by induction. If $p' \in A(t-1)$ or if $h_G(p', t-1) > h_{GH}(p, t-1)$, then it must be the case that some packet $p'' \in Q_G(t-1)$ with $h_G(p'', t-1) \leq h_{GH}(p, t-1)$ is dropped by G at time $t-1$. By the greedy rule $v(p') \geq v(p'')$. Since by induction $v(p'') \geq v(p)$, we are done in this case too. □

LEMMA 3.7. *Let t be any time step. If $p \in Q_{GH}(t) \cap Q_G(t)$, then $h_{GH}(p, t) \leq h_G(p, t)$.*

Proof. Suppose that the lemma does not hold. Let t be the first time it is violated, and let p be the packet with the minimal height such that $h_G(p, t) < h_{GH}(p, t)$. Let p' be the packet immediately below p in $Q_{GH}(t)$ (see Figure 3.2). Note that p' is well defined, since by assumption $h_{GH}(p, t) > h_G(p, t) \geq 1$. Also note that $p' \notin Q_G(t)$. This is because otherwise we would also have $h_G(p', t) < h_{GH}(p', t)$, contradicting the height minimality of p. Due to the minimality of t, $h_G(p, t-1) \geq h_{GH}(p, t-1)$. (Note that we cannot have the situation of $p \in A(t-1)$ and $h_G(p, t) < h_{GH}(p, t)$.) Thus, we must have that $D_G(t-1)$ contains at least one packet below p. Denote this packet by q. We claim that $v(q) \geq v(p')$. If $q = p'$, the claim is trivial. Otherwise, we have that $p' \notin Q_G(t-1)$ and $h_{GH}(p', t-1) \geq h_G(q, t-1)$, and hence, by Lemma 3.6, $v(p') \leq v(q)$. Since $Q_G(t)$ has more packets above p than $Q_{GH}(t)$ has, there must be a packet q' above p in $Q_G(t)$ that is not in $Q_{GH}(t)$. Since $q' \notin D_G(t-1)$ and $q \in D_G(t-1)$, it must be that $v(q') \geq v(q) \geq v(p')$. However, the packet q' is not in $Q_{GH}(t)$. Hence, it has been dropped by GH at some $t' < t$. This yields a contradiction to the head-drop rule, since $v(p') \leq v(q')$, both p' and q' are in $Q_{GH}(t')$, and $h_{GH}(q', t') > h_{GH}(p', t')$. □

We can now complete the proof of the theorem by proving the existence of the mapping. Suppose that $p \in Q_{GH}(t) \setminus Q_G(t)$ is mapped to $q \in Q_G(t) \setminus Q_{GH}(t)$. We now show that $h_{GH}(p, t) \geq h_G(q, t)$. By Lemma 3.6 this implies that $v(p) \leq v(q)$. Recall that the mapping of the packets in $Q_{GH}(t) \setminus Q_G(t)$ is done in ascending order of heights and that any such packet is mapped to the packet with the minimal height in $Q_G(t) \setminus Q_{GH}(t)$ that has not been mapped so far. To obtain a contradiction, suppose that p is the packet with the minimal height in $Q_{GH}(t) \setminus Q_G(t)$ that is mapped to $q \in Q_G(t) \setminus Q_{GH}(t)$, and $h_{GH}(p, t) < h_G(q, t)$. Consider the packet $p' \in Q_G(t)$ such that $h_G(p', t) = h_{GH}(p, t)$. It must be that $p' \in Q_G(t) \cap Q_{GH}(t)$. We must also have that the number of packets below p in $Q_{GH}(t) \setminus Q_G(t)$ is the same as the number of packets below p' in $Q_G(t) \setminus Q_{GH}(t)$. Thus, the set of packets below p in $Q_{GH}(t) \cap Q_G(t)$ is the same as the set of packets below p' in $Q_G(t) \cap Q_{GH}(t)$. It follows that $h_G(p', t) < h_{GH}(p', t)$, in contradiction to Lemma 3.7. This completes the proof of Theorem 3.5. □

The following theorem proves that GH can do much better than the greedy tail-

drop algorithm. Let GT denote the greedy tail-drop algorithm.

THEOREM 3.8. *There exist input sequences for which the value transmitted by GH is arbitrarily close to* $3/2$ *times the value transmitted by GT.*

Proof. We describe sequences for the case $W = 1$, using parameters α and B. At time 0, $B/2 + 2$ packets of value 1 arrive. At time 1, $B/2$ packets of value $\alpha \gg 1$ arrive. Thus, at time 2, the head half of the buffer is filled with cheap packets, and the tail half of the buffer is filled with expensive packets. At time 2, $B/2+1$ more 1-value packets arrive (resulting in overflow); finally, at time $2+B/2$, $B+1$ packets of value α arrive, resulting in an additional overflow. Let us now consider the performance of GT and GH. GT drops at time 2 the last $B/2$ cheap packets and transmits a cheap packet in steps $2, 3, \ldots, 1 + B/2$. At time $2 + B/2$, GT drops $B/2$ expensive packets, and the total value eventually transmitted is $\alpha(B+1)+1(\frac{B}{2}+1)$. On the other hand, GH drops at time 2 the first $B/2$ cheap packets and then drops at time $2 + B/2$ the other $B/2$ cheap packets, for a total transmitted value of $\alpha(\frac{3B}{2} + 1) + 1$. The result follows for large B and α values. □

4. Bounded delay buffers. In this section we consider the case of bounded delay buffers. General bounded delay buffers are studied in section 4.1. We prove a lower bound on the competitiveness of any on-line algorithm. The lower bound holds even in the uniform-delay model, independent of the allowed delay. We then show that for the general model, the greedy algorithm is exactly 2-competitive (the bound is refined in case there are exactly two packet values). Finally, in section 4.2 we provide a detailed analysis of the special case where the delay bound is 2.

4.1. General bounded delay buffers. In this section we consider the general case, where the slack times and values of the packets are arbitrary. We first present a lower bound on the competitiveness of all on-line algorithms and then we turn to analyze the simple greedy algorithm. We show that the greedy algorithm is exactly 2-competitive for the general delay-bounded case.

We begin with a negative result motivated by the following (false) intuition. It may seem reasonable to hope that as the delay bound grows, the competitive factor of on-line algorithms might tend to 1, since infinite delay bound seems like the off-line case. This is not true, as proved in the following theorem. The proof is similar in spirit to the proof in the FIFO case (Theorem 3.1).

THEOREM 4.1. *Let* α *be the ratio of the largest to the smallest packet value. Then for any delay bound* δ*, the competitive ratio of any on-line algorithm is at least* $1+\frac{\alpha-1}{\alpha(\alpha+1)}$*, even for uniform-delay buffers. Furthermore, if there are two packet values whose ratio is* $1 + \sqrt{2}$*, then the competitive ratio of any on-line algorithm is at least* $1 + \frac{1}{(1+\sqrt{2})^2} \approx 1.17$ *for any value of* δ*.*

Proof. Let A be any on-line algorithm, and let C_A be its competitive ratio. To bound C_A, consider the following scenario. All packets have the same slack value δ. At time $t = 0$ the buffer is empty and δ packets of value 1 arrive. During each of the following $\delta - 1$ steps ($t = 1, \ldots, \delta - 1$), a single packet of value $\alpha > 1$ arrives. Let x be the number of value 1 packets transmitted by A by time δ. We consider two possible continuations of the scenario. In the first case, no more packets arrive, and in the second case, δ packets of value α arrive at time δ. In the former case, the value served by A is at most $x + \delta\alpha$, while the optimal value is $\delta + \delta\alpha$. In the latter case, the value served by A is $x + \alpha(2\delta - x)$, while the optimal value is $2\delta\alpha$. It follows that the competitive ratio of A is at least

$$C_A \geq \max\left(\frac{\delta(1+\alpha)}{x+\delta\alpha}, \frac{2\delta\alpha}{x+\alpha(2\delta-x)}\right).$$

Consider the two possible ratios. To get the lower bound our goal is to fix α so that the maximum of the two ratios for any value of x is minimized. This is because the on-line algorithm fixes x given the value of α. For any value of α, it is not difficult to see that the best value of x that can be chosen by A is the one where the two ratios are equal. Solving for x as a function of α, we get that the maximum of the ratios is minimized when

$$x(\alpha) = \frac{2\delta\alpha}{\alpha^2+2\alpha-1},$$

in which case the competitive ratio of the algorithm is

$$C_A \geq 1 + \frac{\alpha-1}{\alpha(\alpha+1)}.$$

This proves the first part of the theorem. To prove the second part of the theorem, we find the worst case α by elementary calculus. It turns out that the competitive ratio is maximized (for $x(\alpha)$, i.e., when the algorithm makes the optimal choice) when $\alpha = 1+\sqrt{2}$. In this case we get that $C_A \geq 1 + \frac{1}{(1+\sqrt{2})^2}$, as desired. □

Next, consider the greedy algorithm for the general case where the allowed delays may be different and the link bandwidth is W. The greedy algorithm is extremely simple: at each time step t, transmit the W packets with the highest value whose deadlines have not expired yet. Ties are broken arbitrarily. Note that effectively, the greedy algorithm views each value as a priority class, in the sense that high-priority packets are always transmitted before low-priority ones. For this simplistic strategy, the following relatively strong property was already known [1, 12, 4] in slightly different models.

THEOREM 4.2. *The greedy algorithm is exactly* 2*-competitive in the bounded-delay buffer model, for any output link bandwidth.*

The lower bound for the greedy algorithm holds even if all jobs have the same weight. We note that for the unslotted model (where a packet may arrive during the transmission of another, and preemption is disallowed), [12] proves a lower bound of 2 on the competitive ratio of any deterministic algorithm, and 4/3 for the expected competitive ratio of any randomized algorithms.

In some practical cases, the values assigned to packets are not very refined. In the extreme case, there may be just "cheap" and "expensive" packets, for example, in ATM's Cell Loss Priority bit [23]. We can formalize this model by assigning only two possible values to packets: 1 for "cheap" and $\alpha > 1$ for "expensive." In the following theorem we prove that in this case, the bound guaranteed by Theorem 4.2 can be sharpened to $1 + 1/\alpha$. Notice that the competitive ratio approaches 1 when α tends to infinity.

THEOREM 4.3. *The greedy algorithm is at most* $1 + 1/\alpha$*-competitive in the bounded-delay buffer model with two packet values of* 1 *and* $\alpha > 1$*, for any output link bandwidth.*

Proof. Fix the input sequence, and let W be the output link bandwidth. Consider any optimal algorithm for the sequence. Clearly, $v(S_O^\alpha) \leq v(S_G^\alpha)$. In addition, the greedy algorithm schedules all packets from S_O^1, except the packets that were lost due to the decision of the greedy algorithm to transmit high-value packets with later deadlines. Since each high-value packet may cause loss of at most one low-value packet, we obtain that $v(S_O^1) - v(S_G^1) \leq v(S_G^\alpha)/\alpha$. The theorem follows. □

4.2. The case of $\delta = 2$. Improving on the greedy algorithm in the bounded-delay model turns out to be a challenging task. In this subsection we present a candidate algorithm. However, we are able to analyze its behavior only for the special case of $\delta = 2$, i.e., under the assumption that each packet must be sent either when it arrives or in the following time step. We also present improved lower bounds for this case.

Before we start, let us recall the following well-known fact. A schedule for a set of packets is called *earliest deadline first schedule* (or EDF for short) if the order in which packets are sent is the order of their deadlines.

LEMMA 4.4. *A set of packets with given arrival times and deadlines can be scheduled with link bandwidth W if and only if it can be scheduled by an EDF schedule.*

In addition, packets with the same deadline in a feasible schedule can be further ordered according to their values.

LEMMA 4.5. *A feasible EDF schedule may be transformed into another feasible EDF schedule in which packets with the same deadline are sent in order of nonincreasing value.*

Thus, without loss of generality, we may assume that the optimal algorithm schedules packets in order of nondecreasing deadlines, and packets with the same deadline in order of nonincreasing value. Ties are broken by the arrival order; i.e., packets arriving first are scheduled first.

We use the following algorithms, stated for general delay bound δ and link bandwidth W. The local-EDF algorithm is presented in Figure 4.1. We show in section 5 how the computation of the optimal schedule can be done efficiently. The β-EDF algorithm is defined by a parameter $0 \le \beta \le 1$ and appears in Figure 4.2.

Note that 1-EDF is the greedy algorithm, which sends the W packets with the highest value, and that 0-EDF is the local-EDF algorithm.

THEOREM 4.6. *The $1/\phi$-EDF algorithm is at most ϕ-competitive in the 2-variable bounded-delay buffer model, for any output link bandwidth W.*

Proof. Fix an arrival sequence. We compare the schedule generated by ϕ-EDF with a specific optimal schedule OPT. Specifically, by Lemmas 4.4 and 4.5, we may assume that OPT is EDF, and that if two packets with the same deadline and different values are sent at different times, then the more valuable packet is sent before the less valuable one. We now prove a series of simple properties that follow directly from the definition of the algorithms.

LEMMA 4.7. *If p is transmitted before p' by β-EDF, then p is not transmitted after p' by OPT.*

Proof. Suppose, for contradiction, that there exist packets p, p' such that $p \in S_G(t) \cap S_O(t+1)$ and $p' \in S_O(t) \cap S_G(t+1)$. Clearly both p and p' have deadline $t+1$. The lemma follows from the fact that both β-EDF and OPT send packets with the same deadline in order of nonincreasing value. ∎

LEMMA 4.8. *If $p \in S_G(t)$ performed a push-out at time t, then $v(q') \le v(p)/\phi$ for all packets q' pushed out at time t.*

Proof. It follows from the fact that at Step 3 of the β-EDF, we consider packets pushing out and packets to be pushed out in order of nonincreasing and nondecreasing value, respectively. ∎

LEMMA 4.9. *Suppose that $p, p' \in S_G(t)$ are packets with deadline $t+1$ such that p performed push-out at time t and p' did not perform a push-out at time t. Then $v(p) \le v(p')$.*

At each time step t do the following:

1. Compute the optimal schedule $\hat{S}$ for all packets not yet sent or expired (i.e., implicitly assuming no new packet will arrive).
2. Send the W packets which are sent at time t in $\hat{S}$ (we may assume, without loss of generality, that $\hat{S}$ is EDF).

FIG. 4.1. *The local-EDF algorithm.*

At each time step t do the following:

1. Compute the optimal EDF schedule $\hat{S}$ for all packets not yet sent or expired. Let S be the set of all packets scheduled in $\hat{S}$, and let S' and S'' be the sets of packets scheduled in $\hat{S}$ to be sent at times t and $t+1$, respectively.
2. Let $p \in S'$ be the packet of minimal value in S', and let $q \in S''$ be the packet of maximal value in S''.
3. If $v(p) < \beta \cdot v(q)$, then
   (a) **Push-out step:** Let $S' \leftarrow S' \cup \{q\} \setminus \{p\}$, and $S'' \leftarrow S'' \setminus \{q\}$. The packet q is said to have *pushed out* packet p.
   (b) Go to Step 2.
4. Otherwise (i.e., $v(p) \geq \beta \cdot v(q)$), terminate the algorithm and transmit S'.

FIG. 4.2. *The β-EDF algorithm.*

Proof. By definition, β-EDF sends packets with the same deadline in nondecreasing value order. □

LEMMA 4.10. *Consider the sets S' constructed at Step 1 at time t by the β-EDF algorithm. If some packet $p \in S_O(t)$ with deadline t is not included in S_G, then $(S' \cap A(t)) \subset S_O(t)$.*

Proof. Let $p' \in S' \cap A(t)$. Notice that $|S'| = W$ and $v(p') > v(p)$ because $p \notin S_G$. We argue that $p' \in S_O(t)$. First, note that $p' \in S_O$, since otherwise OPT could have been improved by swapping p and p'. Hence $p \in S_O(t) \cup S_O(t+1)$. To see that $p' \in S_O(t)$, suppose for contradiction that $p' \in S_O(t+1)$. Note that since p is not included in S_G, we must have that either S' contains no packet with deadline $t+1$, or $|S| = 2W$. In the former case, we get a contradiction to our assumption that $p' \in S'$ because the deadline of p' is $t+1$. In the latter case, we get that since $p' \in S' \cap S_O(t+1)$ and $|S| = 2W$, there exists a packet $p'' \in A(t) \cap S$ with deadline $t+1$ such that $p'' \notin S_O$. Moreover, $v(p'') > v(p)$ by construction of the β-EDF schedule. In this case, OPT can be improved by replacing p with p''. The lemma follows. □

For the remainder of the proof, we define a mapping $m : S_O \rightarrow S_G$ iteratively as follows.

1. If $p \in S_O(t) \cap (S_G(t-1) \cup S_G(t))$ for some time step t, then $m(p) = p$.
2. For each time step t, map any unmapped packet $p \in S_O(t)$ to $p' \in S_G(t)$ such that either
   (a) p' is unmapped, or
   (b) $p' \in S_O(t+1)$ and $|m^{-1}(p')| = 1$.

To prove the theorem, it suffices to show that (i) all packets in S_O are mapped, and that (ii) $v(m^{-1}(p)) \leq \phi \cdot v(p)$ for all $p \in S_G$.

LEMMA 4.11. *If at Step 2 a packet $p \in S_O(t)$ has to be mapped, then there always exists a packet $p' \in S_G$ eligible for either Step 2(a) or Step 2(b).*

Proof. Consider an unmapped packet $p \in S_O(t)$ that is processed in the course of Step 2. Since p has not been mapped at Step 1, either $p \in S_G(t+1)$ or $p \in S_O \setminus S_G$. Thus, since p is available to ϕ-EDF at time t and it is not transmitted at that time, we have that $|S_G(t)| = W$. The lemma follows by construction of the mapping. ∎

LEMMA 4.12. *If a packet $p \in S_O(t)$ is mapped to a packet p', then $v(p) \le v(p')/\phi$.*

Proof. Suppose first that p is mapped to a packet p' during Step 2(a). Then since p is available to ϕ-EDF at time t and it is not transmitted, the value of any packet that is scheduled at that time is at least $v(p) \cdot \phi$. So for the remainder of the proof, assume that p is mapped to p' during Step 2(b). In this case, since p has not been mapped at Step 1, either $p \in S_G(t+1)$ or $p \in S_O \setminus S_G$. By Lemma 4.7, $p \notin S_G(t+1)$. Also, p cannot have deadline $t+1$, since otherwise, $v(p) > v(p')$ and ϕ-EDF would have replaced p' by p. Hence $p \in (S_O \setminus S_G)$, and the deadline of p is t. It follows that it must be the case that either (1) p is pushed out at Step 3(a) of β-EDF, or (2) $p \notin S$, where S is the set computed at Step 1 of β-EDF. Let us analyze these cases.

(1) If p' performed a push-out, then we are done by Lemma 4.8. Otherwise, let q be the packet that pushed out p. According to Lemma 4.9 we have that $v(q) \le v(p')$, and consequently, $v(p) \le v(p')/\phi$.

(2) If $p \notin S$, then by Lemma 4.10 we have that $p' \notin S' \cap A(t)$ because $p' \notin S_O(t) \cap A(t)$ (recall that $p' \in S_O(t+1)$). Thus, if $p' \in S_G(t)$, then it should have pushed out some packet q such that $v(q) \le v(p)/\phi$. By construction of the optimal ϕ-EDF schedule we have that $v(q) \ge v(p)$. Hence, $v(p) \le v(p')/\phi$. ∎

Lemmas 4.11 and 4.12 conclude the proof of Theorem 4.6. ∎

Uniform delay buffers. We remark that for uniform-delay buffers with $\delta = 2$, better upper bounds can be obtained. For example, Corollary 5.5 (in conjunction with Theorem 3.4) says that a competitive ratio of 1.5 is achieved by the FIFO-greedy algorithm in this model. Moreover, careful case analysis shows that the β-EDF algorithm achieves a ratio of about 1.43 when $\beta = \frac{3+\sqrt{13}}{2} \approx 3.3$. We omit the details.

We now prove lower bounds for the case $\delta = 2$. First, we consider the case of arbitrary bandwidth.

THEOREM 4.13. *The competitive ratio of any on-line algorithm for a W-bandwidth 2-uniform bounded-delay model is at least* 10/9. *Moreover, the competitive ratio of any on-line algorithm for a W-bandwidth 2-variable bounded-delay model is at least* 1.17.

Proof. We first show the bound for the uniform model. Fix an on-line algorithm A and consider the following scenarios. Initially, the buffer is empty and $2W$ packets of value 1 arrive. At the next step W packets of value α arrive. Suppose that A drops a fraction $x \le 1$ of the 1-value packets. We consider two possible scenarios. In the first no more packets arrive. Then the competitive ratio is bounded from below by $\frac{\alpha+2}{\alpha+2-x}$ since there exists a feasible schedule of the whole sequence. In the second scenario $2W$ packets of value α arrive at the following time step. In this case the competitive ratio of A is bounded from below by $\frac{3\alpha+1}{(2+x)\alpha+2-x}$. Similar to the proof of Theorem 4.1, the "best" value of x is the one that equates the two ratios. In this case we get $x = \frac{(\alpha+2)(\alpha-1)}{\alpha^2+4\alpha-1}$. Substituting and maximizing for α we get $\alpha = 3$ and $x = 1/2$ to yield the ratio 10/9.

We now establish the bound for the variable model. Fix an on-line algorithm A and consider the following scenarios. Initially, the buffer is empty and W packets of value 1 and delay 1 arrive. At the same step, W packets of value α and delay 2 arrive. Suppose that A drops a fraction $x \le 1$ of the 1-value packets. We consider two possible scenarios. In the first no more packets arrive. Then the competitive

ratio is bounded from below by $\frac{\alpha+1}{\alpha+1-x}$ since there exists a feasible schedule of the whole sequence. In the second scenario, W packets of value α and delay 1 arrive at the next time step. In this case the competitive ratio of A is bounded from below by $\frac{2\alpha}{(1+x)\alpha+1-x}$. These are the same ratios considered in the proof of Theorem 4.1, and hence we get the same bound of $1+\frac{1}{(1+\sqrt{2})^2} \approx 1.17$. □

Slightly better results can be proved for bandwidth 1.

THEOREM 4.14. *The competitive ratio of any deterministic on-line algorithm for a* 2*-uniform and a* 2*-variable bounded-delay model with bandwidth* 1 *is at least* 1.25 *and* $\sqrt{2}$*, respectively.*

Proof. Consider the uniform model first. Fix an on-line algorithm A and consider the following scenario. At time 0, the buffer is empty and two packets of value 1 arrive, and at time 1, a packet of value $\alpha > 1$ arrives. There are two possible continuations. In one, no more packets arrive, and in the other, at time 2 two additional packets of value α arrive. Now, if A drops one of the low-value packets, then its competitive ratio is at least $\frac{\alpha+2}{\alpha+1}$ since there exists a feasible schedule of all three packets of the first continuation. Otherwise, at least one packet of value α is lost by A in the second continuation, and hence the competitive ratio of A is at least $\frac{3\alpha+1}{2\alpha+2}$. Setting $\frac{\alpha+2}{\alpha+1} = \frac{3\alpha+1}{2\alpha+2}$, we get that for $\alpha = 3$, the competitive ratio of A is at least 1.25.

The bad example for the variable delay model is even simpler. Let A be an on-line algorithm, and consider the following scenario. At time 0, the buffer is empty and a packet having value 1 and delay 1 arrives together with a packet of value $\alpha > 1$ and delay of 2. The two possible continuations are (i) no more arrivals and (ii) at time 1 an additional packet of value α with delay 1 (i.e., zero slack time) arrives. If A drops the low-value packet, then its competitive ratio for continuation (i) is $\frac{\alpha+1}{\alpha}$ since there exists a feasible schedule of both packets. If the low-value packet is scheduled, then for continuation (ii), A loses at least one high-value packet, showing that its competitive ratio is at least $\frac{2\alpha}{\alpha+1}$. Solving $\frac{\alpha+1}{\alpha} = \frac{2\alpha}{\alpha+1}$, we get that for $\alpha = 1+\sqrt{2}$, the competitive ratio of A is at least $\sqrt{2}$. □

5. The off-line case. In this section we show that the FIFO model has matroid structure in the off-line setting. As a result, optimal off-line solutions can be found in polynomial time. We also study the connection between the FIFO model and the bounded-delay model.

We first consider the FIFO model. Fix the input sequence for the remainder of this section. We also assume, without loss of generality, that all packets admitted to the buffer are later sent: since we are dealing with the off-line case, a packet that will be dropped can simply be rejected when it arrives.

Let $\mathcal{C}$ be the class of all work-conserving schedules, defined as follows. A schedule A is said to be *work conserving*, denoted by $A \in \mathcal{C}$, if

$$|S_A(t)| = \min(W,\ |Q_A(t) \cup A(t) \setminus D_A(t)|) \qquad \text{for all time steps } t,$$

where W is the link bandwidth. In words, a schedule is work-conserving if a packet may be delayed only when the full bandwidth is used by other packets. Note that work-conserving algorithms may still reject packets arbitrarily.

THEOREM 5.1. *For a FIFO schedule* A*, let* S_A *be the set of all packets served by* A*. Then* $\mathfrak{I}_{FIFO} \stackrel{\text{def}}{=} \{S_A : A \in \mathcal{C}\}$ *is a matroid.*

Proof. There are three properties to verify. The first two are trivial: $\emptyset \in \mathfrak{I}$, and for $S_A \subset S_B$ with $S_B \in \mathfrak{I}$, we clearly have that $S_A \in \mathfrak{I}$ by dropping the packets in $B \setminus A$. It remains to verify the following property: If $|S_A| > |S_B|$, then there exists $p \in S_A \setminus S_B$

such that $S_B \cup \{p\} \in \mathfrak{I}$. This can be seen as follows. Let t_0 be the first time in which $|S_A(t_0)| > |S_B(t_0)|$: t_0 exists by the assumption that $|S_A| > |S_B|$. Let $t_1 \le t_0$ be the last step before t_0 where $|D_A(t_1)| < |D_B(t_1)|$: t_1 exists by the assumption that B is work conserving. Then there exists a packet $p \in D_B(t_1) \setminus D_A(t_1)$. Moreover, since by our choice, for any $t \in [t_1, t_0]$ we have that $|S_B(t)| \ge |S_A(t)$ and $|D_B(t)| \le |D_A(t)|$, and since A is work conserving by assumption, we also have that $|Q_B(t)| < |Q_A(t)|$ for all $t \in [t_1, t_0]$. Hence the packet p can be added to S_B while keeping the schedule feasible. □

A similar result for the bounded delay case is well known [10, Theorem 17.12]. We state it here for completeness.

THEOREM 5.2. *For a bounded delay schedule A, let S_A be the set of all packets served by A. Then $\mathfrak{I}_{BD} \stackrel{\text{def}}{=} \{S_A : A \in \mathcal{C}\}$ is a matroid.*

We remark that for the FIFO model (and hence for the uniform bounded-delay model as well; see Theorem 5.4 below), an optimal solution can be found in $O(n \log B)$ time, and in $O(n^2)$ time for the variable bounded delay model, where n is the number of packets in the input sequence and B is the buffer size.

COROLLARY 5.3. *An optimal schedule for the FIFO and bounded delay models can be found in polynomial time.*

The following theorem shows a transformation from the FIFO model to the uniform bounded-delay model.

THEOREM 5.4. *For any input sequence, the optimal value served by a FIFO schedule with buffer size B is equal to the optimal value served by a uniform bounded-delay schedule with $\delta = B + 1$.*

Proof. Let OPT_F be the optimal value served by a FIFO schedule with buffer size B, and let OPT_D be the optimal value served by a uniform bounded-delay schedule with $\delta = B + 1$. First, note that any work-conserving schedule in the FIFO model is also a schedule in the bounded delay model, since no packet in the FIFO model is served more than B time units after its arrival, and hence $OPT_F \le OPT_D$. For the other direction, consider any schedule in the uniform bounded-delay model. Since in this model, a set of packets can be served if and only if the EDF schedule of this set is feasible, we may assume without loss of generality that the schedule is EDF. Also note that the number of packets in the bounded delay buffer is never more than the maximal delay bound (recall that only packets that are eventually transmitted enter the buffer). The result now follows from the fact that an EDF schedule in the uniform bounded delay model is exactly the FIFO order, and hence $OPT_D \le OPT_F$. □

A nice feature of the FIFO to bounded-delay transformation in the proof of Theorem 5.4 is that it does not require off-line information. We therefore have the following corollary.

COROLLARY 5.5. *Let $C_F(B)$ be the best competitive factor of on-line FIFO algorithms with buffer size B, and let $C_D(\delta)$ be the best competitive factor of on-line uniform bounded-delay algorithms with maximal delay δ. Then $C_D(B+1) \le C_F(B)$.*

We remark that the converse cannot be proved by our transformation, since it requires knowledge of the future.

6. Conclusion. In this work we studied competitive overflow management. We sharpened the results of [20] for the FIFO model, and initiated a study in the bounded-delay model. In particular, we have proved the following facts:

- The greedy algorithm is 2-competitive for FIFO buffers.
- Among all the greedy algorithms, head-drop is the best.
- No on-line algorithm can be optimal for the bounded-delay case.

- The greedy algorithm is $(1+\frac{1}{\alpha})$-competitive in the bounded delay model when the set of possible values is 1 and $\alpha > 1$.

Many important questions remain open:

- In the FIFO model, can one substantially improve on the 2-competitiveness of the greedy algorithm in the general case? The best known result [14] is for the case of two packet values 1 and $\alpha > 4$, with competitive ratio of $\frac{\sqrt{\alpha}+1}{\sqrt{\alpha}}$. Combining this with our results for the greedy algorithm, one can get a competitive ratio better than 2 for this case only. There is no real improvement for the general case of packet values.
- In the bounded delay model, we know very little in the general case. Even for the uniform bounded delay case, we know how to improve the greedy algorithm only for the special case of $\delta = 2$.

REFERENCES

[1] M. Adler, A. L. Rosenberg, R. K. Sitaraman, and W. Unger, *Scheduling time-constrained communication in linear networks*, in Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures, 1998, pp. 269–278.

[2] W. Aiello, Y. Mansour, S. Rajagopolan, and A. Rosen, *Competitive queue policies for differentiated services*, in Proc. IEEE INFOCOM, 2000, pp. 431–440.

[3] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts, *On-line routing of virtual circuits with applications to load balancing and machine scheduling*, J. ACM, 44 (1997), pp. 486–504.

[4] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber, *A unified approach to approximating resource allocation and scheduling*, J. ACM, 48 (2001), pp. 1069–1090.

[5] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber, *Approximating the throughput of multiple machines in real-time scheduling*, SIAM J. Comput., 31 (2001), pp. 331–352.

[6] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, and F. Wang, *On the competitiveness of online real-time task scheduling*, in Proceedings of the 32nd IEEE Symposium on Real-Time Systems, 1992, pp. 125–144.

[7] S. Baruah, G. Koren, B. Mishra, A. Raghunathan, L. Rosier, and D. Shasha, *Online scheduling in the presence of overload*, in Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science, 1991, pp. 101–110.

[8] D. Black, S. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss, *An Architecture for Differentiated Services*, Internet draft, RFC 2475, Internet Engineering Task Force, 1998.

[9] D. Clark and J. Wroclawski, *An Approach to Service Allocation in the Internet*, unpublished, 1997.

[10] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.

[11] S. Floyd and V. Jacobson, *Random early detection gateways for congestion avoidance*, IEEE/ACM Trans. on Networking, 1 (1993), pp. 397–413.

[12] S. Goldman, J. Parwatikar, and S. Suri, *On-line scheduling with hard deadlines*, J. Algorithms, 34 (2000), pp. 370–389.

[13] V. Jacobson, K. Nicholas, and K. Poduri, *An Expedited Forwarding PHB*, Internet draft, RFC 2598, Internet Engineering Task Force, 1999.

[14] A. Kesselman and Y. Mansour, *Loss-bounded analysis for differentiated services*, in Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 2001, pp. 591–600.

[15] G. Koren and D. Shasha, *D^{over}: An optimal on-line scheduling algorithm for overloaded uniprocessor real-time systems*, SIAM J. Comput., 24 (1995), pp. 318–339.

[16] M. A. Labrador and S. Banerjee, *Packet dropping policies for ATM and IP networks*, IEEE Communications Surveys, 2 (1999).

[17] T. V. Lakshman, A. Neidhardt, and T. Ott, *The drop from front strategy in TCP and in TCP over ATM*, in Proc. IEEE INFOCOM, 1996, pp. 1242–1250.

[18] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, *On the self-similar nature of ethernet traffic (extended version)*, IEEE/ACM Trans. on Networking, 2 (1994), pp. 1–15.

[19] R. Lipton and A. Tomkins, *Online interval scheduling*, in Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 1994, pp. 302–311.

[20] Y. Mansour, B. Patt-Shamir, and O. Lapid, *Optimal smoothing schedules for real-time streams*, Distributed Computing, 17 (2004), pp. 77–89.

[21] M. May, J.-C. Bolot, A. Jean-Marie, and C. Diot, *Simple performance models of differentiated services for the Internet*, in Proc. IEEE INFOCOM, 1999, pp. 1385–1394.

[22] C. Phillips, R. Uma, and J. Wein, *Off-line admission control for general scheduling problems*, in Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 2000, pp. 879–888.

[23] The ATM Forum Technical Committee, *Traffic Management Specification Version* 4.0, ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0056.000.pdf (April 1996).

[24] A. Veres and M. Boda, *The chaotic nature of TCP congestion control*, in Proc. IEEE INFOCOM, 2000, pp. 1715–1723.