Buffer Overflow Management in QoS Switches

Alexander Kesselman*§ alx@math.tau.ac.il
Zvi Lotker†§ zvilo@eng.tau.ac.il
Yishay Mansour*§ mansour@math.tau.ac.il
Boaz Patt-Shamir†§ boaz@eng.tau.ac.il
Baruch Schieber‡ sbar@us.ibm.com
Maxim Sviridenko‡ sviri@us.ibm.com

October 31, 2000

Abstract

We consider two types of buffering policies that are used in network switches supporting QoS (Quality of Service). In the *FIFO* type, packets must be released in the order they arrive; the difficulty 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 released, or otherwise it is lost. We study the cases where the incoming streams overload the buffers, resulting in packet loss. In our model, each packet has an intrinsic value; the goal is to maximize the total value of packets transmitted

Our main contribution is a thorough investigation of the natural greedy algorithms in various models. For the FIFO model we prove tight bounds on the competitive ratio of the greedy algorithm that discards the packets with the lowest value. 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 standard tail-drop policy, that drops the latest packets.

In the bounded delay model we show that the competitive ratio of any online 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.

*Dept. of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel.
†Dept. of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel.
‡IBM T.J. Watson Research Center, P.O. Box 218 Yorktown Heights, NY 10598, USA.
§Research partly supported by a grant from Israel Ministry of Science and Technology.

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 has been a few proposals that address the integration of QoS in the IP framework, and our models are based on some of these proposals. As a matter of fact, our work falls under the general area of providing *differentiated network services* in an IP environment. With the realization that "not all traffic is equal" comes the quest to provide network service differentiation. For example, different customers may get different levels of service, which might depend on the price they pay for the service.

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 always, regardless of all other connections. This conservative policy (implemented in the specification of CBR traffic in ATM networks [20]) 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). 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 has not arrived after packet p'. The main difficulty 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 is allowed to re-order the 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: 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

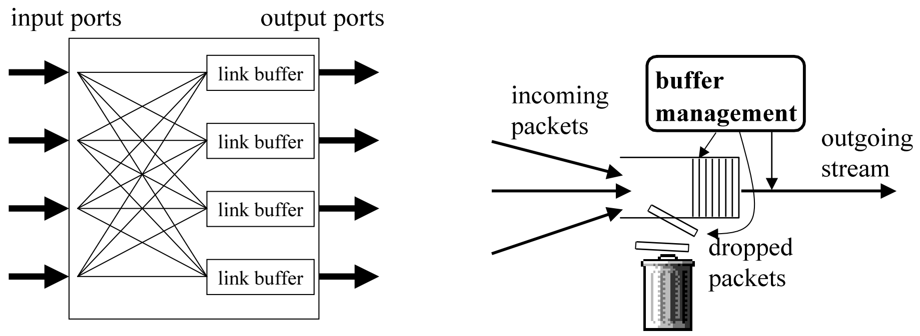

Figure 1: *Schematic representation of the model. Left: general switch structure. Right: output port structure. Packets are placed in the buffer, and the buffer management algorithm controls which packet will be discarded and which will be transmitted.*

broken arbitrarily. We prove a tight bound of $2 - \frac{1}{B+1}$ on the competitive factor of this algorithm, where B is the buffer size. 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. We remark that the proof of the upper bound is quite involved. We then consider the different variants of the greedy algorithms, since the greedy policy does not specify which packet to drop in case there are more then one 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 of the head-drop greedy policy to the value of the tail-drop policy can be as high as $3/2$ in some cases.

For the bounded delay model we have the following results. First, we show that the competitive ratio of any online algorithm for a uniform bounded delay buffer is bounded away from 1, independent of the delay size. This holds even if all packets has 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 at most 2, namely, a packet is either sent when it arrives, or in the next time slot, or else it is lost. 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 yields 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 offline algorithms.

Related work. There is a myriad of research papers about packet drop policies in communication networks—see, for example, the survey of [13] and references therein. Some of the drop mechanisms, such as RED [9], 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 [7, 8] and ATM [20]. The bounded delay model is an abstraction of the model described in [10].

There has been work on analyzing various aspects of the model using classical queuing theory, and assuming Poisson arrivals [18]. The Poisson arrival assumptions has been seriously undermined by the discovery of the heavy tail nature of traffic [15] and the chaotic nature of TCP [21]. In this work we use competitive analysis, which makes no probabilistic assumptions.

The work of [1] concentrated on the case where one cannot discard a packet already in the buffer. They give tight bounds on the competitive factor of various algorithms for the special case where there are only two different weights. In [17], 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 [11] studies the competitive analysis of the lost value rather then the throughput value. Translating the loss bound to to the throughput bound considered here, for the variant where only two packet weights are possible, one can show that the competitive ratio approaches one as the ratio of the two weights tends to infinity. A similar result is presented in this paper for the bounded delay model. The work of [2] 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 buffer. The goal is to maximize the throughput, that is, the weight of the jobs that terminate by their deadline. In our case we have also the additional constraint that all jobs have the same processing time. The off-line variant of this scheduling problem denoted $P|r_i; p_i = p| \sum w_i(1 - U_i)$ 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 [4, 3, 19]. The more general on-line problem where the processing time is not fixed was considered in [16]. In this case the results are substantially worse than our case. Slightly better results can be achieved for the case where processing time is not fixed if preemption is allowed. This case was considered in [5, 6, 12]. 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 in case all processing times are the same), [5] showed a tight competitive factor of 4. For the weighted version the bound is $\sqrt{1 + k^2}$ where k is the ratio of the maximum value density to the minimum value density of a job [12].

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. Due to lack of space, most proofs are presented in the appendices.

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 $v(p)$. At time t, a set of packets $A(t)$ arrives. A set of packets from $Q_A(t) \cup A(t)$, denoted $S_A(t)$, is *transmitted*. A subset of $Q_A(t) \cup A(t) \setminus S_A(t)$, denoted $D_A(t)$ is *dropped*. The set of packets in the buffer at time $t + 1$ is $Q_A(t + 1) =$

$Q(t) \cup A(t) \setminus (D_A(t) \cup S_A(t))$. 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 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 $v(P)$ 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, that the number of packets in the buffer is bounded by the *buffer size* parameter, denoted B. Formally, the constraint is that for all times t, $|Q(t)| \leq B$.[1]

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 $sl(p)$. The requirement is that for all packets $p \in A(t)$, if $p \in Q(t + sl(p))$ then it must be the case that $p \in S(t + sl(p)) \cup D(t + sl(p))$. In words, each packet $p \in A(t)$ must be either transmitted or dropped by time $t + sl(p)$. The time $t + sl(p)$ is also called the *deadline* of p, denoted $dl(p)$. We emphasize that in this model there is no explicit bound on the size of the buffer, and that packets may be re-ordered.

Uniform and variable bounded-delay buffers. One case of special interest of bounded-delay buffers 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 $t' > t$. In case future packet arrival is known the algorithm is called *off-line*. The *competitive ratio* (or *competitive factor*) of an algorithm A is an upper bound, over all input sequences, on the ratio of the maximal value that can be transmitted by *any* off-line algorithm to the value that is transmitted by A. Note that since we deal with a maximization problem this ratio will always be at least 1.

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 they arrive. Each packet has a value associated with it and the goal is to maximize the value of the transmitted packets.

[1]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.

3.1 Tight Analysis of the Greedy Algorithm

We consider the greedy algorithm. In this algorithm no packets are dropped in time steps t in which $|Q(t)| + |A(t)| \leq B + 1$. If $Q(t) \cup A(t) \neq \emptyset$, then the earliest packet is transmitted and the rest are stored in the buffer. In case $|Q(t)| + |A(t)| = B + k > B + 1$, the $k - 1$ lowest value packets are dropped, with ties broken arbitrarily. Among the remaining $B + 1$ packets the earliest one is transmitted and the rest are stored in the 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.

Theorem 3.1 *The competitive ratio of the greedy algorithm is at most* 2.

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

We construct the mapping iteratively. At each iteration we consider a packet from $DROP$ and map it to a packet in $GREEDY$. The packets in $DROP$ are considered in the order of their dropping time. Suppose that the current packet to be mapped is $p \in DROP_t$. This means that all the packets in $DROP_s$, for $s < t$, and maybe some of the packets in $DROP_t$ have been mapped already. Given the partially defined mapping, define a set of "available" packets in $GREEDY$ as follows. A packet in $q \in GREEDY \cap OPT$ is available if so far no packet is mapped to q. A packet in $q \in GREEDY \setminus OPT$ is available if so far no more than 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.

It is not difficult to see that the mapping defined above satisfies the second and third properties. It remains to be shown is that it satisfies the first property as well. We show this by induction following the order in which the dropped packets are mapped. For the basis of the induction note that if $p \in DROP_t$ 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 $v(p)$.

Suppose that the first property holds for all packets mapped before $p \in DROP_t$. We show that it holds also for p. 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 $r(t) = 1$ (the first time unit). 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. Observe that since the buffer is full at time t, $s(t)$ is always defined.

Claim 3.2 *The value of the packets transmitted by the greedy at times* $[t..s(t) + B]$ *is no less than* $v(p)$.

Proof: Since p is dropped at time t, the buffer at time t is full and the minimum value in the buffer at this time is at least $v(p)$. This implies that the value of each of the packets transmitted at times $[t..t + B]$ is at least $v(p)$. If $s(t) = t$, we are done. Otherwise, let $t' \in [t + 1..t + B]$ be the earliest time in which a packet that was in the buffer at time t is dropped. Since the dropped packet has value at least $v(p)$, the values of each of the packets transmitted at times $t', \ldots, t' + B$ is at least $v(p)$. Again, if $s(t) = t'$, we are done. Otherwise continue in the same manner. ∎

To prove that the first property holds for p we show that it is mapped to one of the packets transmitted by time $s(t) + B$. For this we need to show that at least one of the packets transmitted at times $[t..s(t) + B]$ 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 $[t...s(t) + B]$. Specifically, every available packet (at the time p is mapped) among the packets transmitted at times $[t...s(t) + B]$ that is also in OPT contributes 1 to the number of availabilities. Every available packet (at the time p is mapped) among the packets transmitted at times $[t...s(t) + B]$ that is not in OPT contributes 2 to the number of availabilities, except of at most one such packet that contributes 1, in case one packet has been mapped to it already.

Claim 3.3 *The number of availabilities before any of the packets dropped at time t are mapped is at least $\sum_{i=t}^{s(t)} |DROP_i|$.*

Proof: We first show that the number of availabilities before any of the packets dropped at time $r(t)$ and later are mapped is at least $\sum_{i=r(t)}^{s(t)} |DROP_i|$. Consider all the packets transmitted by the optimal algorithm that are considered by the greedy algorithm (and either transmitted or dropped) at times $[r(t)..s(t)]$. The maximum number of these packets is bounded by $s(t) - r(t) + 2B + 1$. This is since the earliest time these packets could be transmitted by the optimal algorithm is $r(t) - B$ and the latest time these packets could be transmitted by the optimal algorithm is $s(t) + B$. Let x be the number of these packets that are transmitted by the greedy algorithm at times $[r(t)..s(t) + B]$. Observe that no packets from OPT that arrive at times $[s(t)..s(t) + B]$ are transmitted by the greedy algorithm at times $[s(t)..s(t) + B]$. This follows from the definition of $s(t)$, since the transmission of any such packet at times $[s(t)..s(t) + B]$ would result in dropping at least one packet that has been in the buffer at time $s(t)$. It follows that exactly x packets from OPT are transmitted by the greedy algorithm at times $[r(t)..s(t) + B]$. This implies that the number of availabilities before any of the packets dropped at time $r(t)$ and later are mapped is at least $2(s(t) + B - r(t) + 1) - x$. This is since none of the packets that are dropped before time $r(t)$ are mapped to packets transmitted at times $[r(t)..s(t) + B]$. It follows that

$$\sum_{i=r(t)}^{s(t)} |DROP_i| \le s(t) - r(t) + 2B + 1 - x \le 2(s(t) + B - r(t) + 1) - x \ .$$

By the definition of $r(t)$ the maximum number of packets that can be mapped to the packets transmitted at times $[r(t)...t - 1]$ is strictly less than $\sum_{i=r(t)}^{t-1} |DROP_i|$. Since the maximum number of packets that can be mapped to the packets transmitted at times $[r(t)...s(t) + B]$ is at least $\sum_{i=r(t)}^{s(t)} |DROP_i|$. We conclude that the number of availabilities before the packets dropped at t are mapped is at least $\sum_{i=t}^{s(t)} |DROP_i|$. ∎

The theorem follows directly from the claim. ∎

We now refine the analysis of the previous proof to get a tighter bounds. Let α be the ratio of the largest value of a packet to the smallest value of a packet, and let B be the buffer size. In Appendix A we prove the following results.

Theorem 3.4 *The competitive ratio of the greedy algorithm is at most $2(1 - \frac{1}{\alpha+1})$, where $\alpha \stackrel{\text{def}}{=} \frac{\max_p\{v_p\}}{\min_p\{v_p\}}$.*

Theorem 3.5 *The competitive ratio of the greedy algorithm is at most $2 - \frac{1}{B+1}$, where B is the buffer size.*

Finally, we observe that the last two bounds are tight. Consider the following scenario. At time 1 $B + 1$ packets of value ϵ arrive. One is transmitted and the rest are kept in the buffer. Then, at each time $t \in [2..B + 1]$

a packet of value M arrives and is kept in the buffer. At time $B+2$ the buffer contains B packets of value M and then another $B+1$ packets of value M arrive. The greedy algorithm transmits only $B+1$ of these packets while the optimal algorithm transmits $2B+1$ of value M and only one of value ϵ. Let $\alpha = \frac{M}{\epsilon}$. We get that the competitive ratio is

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

Letting α go to infinity we get the ratio $2 - \frac{1}{B+1}$, 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 not completely defined: when there are several packets with the same low value, it is not specified which of them is discarded by the greedy algorithm. 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 common practice of *tail-drop*, where the newest packets are discarded. In fact, tail-drop is the worst variant of the greedy algorithm: 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 [14] enjoys additional advantages in the TCP/IP environment, namely, it helps the congestion avoidance mechanism.

The following theorem proves that GH is superior to any other greedy algorithm.

Theorem 3.6 *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.*

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 3.7 *There exists input sequences for which the value transmitted by GH is $3/2$ times the value transmitted by GT.*

The proofs are presented in Appendix A. We remark that the proof of Theorem 3.6 uses a somewhat different technique than the one used in the proof of Theorem 3.1.

4 Bounded Delay Buffers

In this section we consider the case of bounded delay buffers. We first show that even if the allowed delay is arbitrary large, any online algorithm is inferior to the offline algorithm. We show that for the general model, the greedy algorithm is exactly 2-competitive; we also prove that a better competitive factor can be attained when there are only two possible packet values. At the end we provide detailed analysis of the special case where the delay bound is 2. All the proofs appear in Appendix B.

We begin with a negative result whose motivation is as follows. It seems reasonable to hope that as the delay bound grows, the competitive factor of on-line algorithms might tend to 1: after all, an infinite delay bound seems like the off-line case! This intuition is false, as proved in the following theorem. In fact, even if all slack times are equal, the competitive factor of any on-line algorithm is bounded away from 1.

Theorem 4.1 *For all delay bounds δ, the competitive ratio of any on-line algorithm is bounded away from* 1. *For δ-uniform-delay buffers the competitive ratio tends to* 1.17 *as δ tends to infinity.*

Next, we consider the greedy algorithm for the general case where the allowed delays may be different and the link bandwidth is a parameter 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 we have the following relatively strong property.

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

In some cases, the values assigned to packets are not very refined. In the extreme case, there may be just "cheap" and "expensive" packets. 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 α, for any output link bandwidth.*

In some cases, where the end-to-end delay is important and the physical propagation delay is already high due to distance, QoS requirements dictate that a packet may never be delayed in a buffer, or may be delayed at most one step. In our terms, we have a bounded-delay buffer with $\delta = 2$. We study this case in detail in this section. We use the following algorithms.

The Greedy EDF Algorithm. At each time step t the online algorithm computes the optimal schedule S, starting at time t, assuming no new packet will arrive. (If the optimal schedule has a choice it behaves greedily, i.e., it chooses the higher weight packets earliest.) The Greedy EDF sends at time t the W packets which are sent at time t in the schedule S.

The β-Ratio EDF Algorithm. The β-Ratio EDF Algorithm first computes the Greedy EDF schedule, S. Let S' be the set of packets schedule to be sent at time t in schedule S. Then until the maximal value among the packets in $S \setminus S'$ is less than β times the minimal value among the packets in S' it swaps a packet with minimal value from S' and a packet with maximal value from $S \setminus S'$. The β-Ratio EDF Algorithm schedules S' at time t.

Another way to view this is the following. We sort both S' and $S \setminus S'$. Let i be the maximal index such that the value of the ith packet in S' is at least the value of the p_{W-i+1} packet in $S \setminus S'$ divided by β. The β-Ratio EDF Algorithm schedules the i highest value packets from S' and the $W - i$ highest packets from $S \setminus S'$.

Note that 1-Ratio EDF gives the greedy algorithm, which sends the W packets with the highest value. Also, 0-Ratio EDF is simply the Greedy EDF algorithm.

We first consider the general case, where the link bandwidth is a parameter. We prove that the β-Ratio EDF algorithm achieves competitive ratio of ϕ for $\beta = \phi$, where $\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$ is the Golden Ratio.

Theorem 4.4 *The ϕ-Ratio EDF algorithm is at most ϕ-competitive in the 2-variable bounded-delay buffer model, for any output link bandwidth W.*

The next theorem establishes an impossibility result for the uniform and variable bounded-delay models with arbitrary bandwidth.

Theorem 4.5 *The competitive ratio of any online scheduling algorithm for W-bandwidth 2-uniform bounded-delay model is at least $10/9$. Moreover, the competitive ratio of any online scheduling algorithm for W-bandwidth 2-variable bounded-delay model is at least 1.17.*

We consider the case of uniform bounded-delay buffers with link bandwidth 1, which is an important case for getting a good insight into the problem. In this case we can improve the previous bound.

Theorem 4.6 *The* $(1.5 + \sqrt{13}/2)$*-Ratio EDF algorithm is at most* 1.43*-competitive for* 2*-uniform bounded-delay buffer model with bandwidth* 1*. Moreover, the Greedy EDF algorithm is at most* 1.5*-competitive for* 2*-uniform bounded-delay buffer model with bandwidth* 1*.*

The next theorem establishes impossibility results for the uniform and variable models with bandwidth 1.

Theorem 4.7 *The competitive ratio of any online scheduling algorithm for* 2*-uniform and* 2*-variable bounded-delay model with bandwidth* 1 *is at least* 1.25 *and* $\sqrt{2}$*, respectively.*

5 The Off-line Case

In this section we show that both the FIFO and the bounded delay models of buffering have matroid structure. As a result, optimal off-line solutions can be found in polynomial time. At the end, we 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 be simply rejected when it arrives.

Let $\mathcal{C}$ be the class of all FIFO work-conserving schedules, defined as follows. A schedule A is said to be *work conserving*, denoted $A \in \mathcal{C}$, if for all t we have that $S_A(t) = \emptyset$ if and only if $Q_C(t) \cup A(t) \setminus D_C(t) = \emptyset$. In words, a schedule is work conserving is always sends a packet if some packet is available. Note that work conserving algorithms may reject packets arbitrarily: the only thing disallowed is not to send any packet at some time step even though some packets are retained in the buffer for later transmission.

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

We now prove a similar result for the bounded delay case. Let $\mathcal{E}$ be the class of work conserving schedules.

Theorem 5.2 *For a bounded delay schedule* A*, let* S_A *be the set of all packets served by* A*. Then* $\mathfrak{I} \stackrel{\text{def}}{=} \{S_A \,:\, A \in \mathcal{E}\}$ *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 $O(n \log n)$ time, and in $O(n^2)$ time for the variable bounded delay model. We omit details here.

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* $\delta = B$*.*

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* $C_D(B) \leq C_F(B)$*.*

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

References

[1] W. Aiello, Y. Mansour, S. Rajagpopalan, and A. Rosen. Competitive queue policies for diffrentiated services. In *INFOCOM*, 2000.

[2] B. Awerbuch, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. In *Proc. of the 23rd ACM Symp. on Theory of Computing*, pages 623 – 631, 1993.

[3] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. In *Proc. 32nd ACM Symp. on Theory of Computing (STOC)*, pages 735–744, 2000.

[4] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Approximating the throughput of multiple machines under real-time scheduling. In *Proc. 31st ACM Symp. on Theory of Computing (STOC)*, pages 622–631, 1999.

[5] 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 *Proc. 12th IEEE Symp. on Real Time Systems*, pages 106–115, 1991.

[6] S. Baruah, G. Koren, B. Mishra, A. Raghunathan, L. Rosier, and D. Shasha. Online scheduling in the presence of overload. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 101–110, 1991.

[7] D. Black, S. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. Internet RFC 2475, December 1998.

[8] D. Clark and J. Wroclawski. An approach to service allocation in the internet. Internet draft, 1997. Available from `diffserv.lcs.mit.edu`.

[9] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. on Networking*, 1(4):397–413, 1993.

[10] V. Jacobson, K. Nicholas, and K. Poduri. An expedited forwarding PHB. Internet RFC 2598, June 1999.

[11] A. Kesselman and Y. Mansour. Loss-bounded analysis for differentiated services. In *SODA*, 2001.

[12] G. Koren and D. Shasha. D-over: An optimal on-line scheduling algorithm for overloaded real-time systems. *SIAM Journal on Computing*, 24:318–339, 1995.

[13] M. A. Labrador and S. Banerjee. Packet droping policies for ATM and IP networks. *IEEE Communications Surveys*, 2(3), 1999.

[14] T. V. Lakshman, A. Neidhardt, and T. Ott. The drop from front strategy in TCP and in TCP over ATM. In *INFOCOM*, pages 1242–1250, 1996.

[15] 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.

[16] R. Lipton and A. Tomkins. On-line interval scheduling. In *Proc. 5th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 302–311, 1994.

[17] Y. Mansour, B. Patt-Shamir, and O. Lapid. Optimal smoothing schedules for real-time streams. In *PODC*, 2000.

[18] M. May, J.-C. Bolot, A. Jean-Marie, and C. Diot. Simple performance models of differentiated services for the internet. In *INFOCOMM*, 1998.

[19] C. Phillips, R. Uma, and J. Wein. Off-line admission control for general scheduling problems. In *Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 879–888, 2000.

[20] The ATM Forum Technical Committee. Traffic management specification version 4.0, Apr. 1996. Available from `www.atmforum.com`.

[21] A. Veres and M. Boda. The chaotic nature of TCP congestion control. In *INFOCOMM*, 2000.

APPENDICES

A Proofs from Section 3

Proof of Theorem 3.4: Given the mapping defined above, partition $GREEDY$ into three subsets: (i) The set G_1 of packets in $GREEDY$ that are not in OPT and that are not the target (in the mapping) of any packet in $DROP$. (ii) The set G_2 of packets in $GREEDY$ that are unavailable; that is, the packets in $GREEDY \cap OPT$ that are the image of one packet in $DROP$, and the packets in $GREEDY \setminus OPT$ that are the image of two packets in $DROP$. (iii) The set G_3 of the rest of the packets in $GREEDY$; that is, the packets in $GREEDY \cap OPT$ that are not the image of any packet in $DROP$, and the packets in $GREEDY \setminus OPT$ that are the image of one packet in $DROP$.

We associate two packets from OPT with each packet in $q \in G_2$ as follows. If $q \in G_2 \cap OPT$ then the two packets are q itself and the packet mapped to q. If $q \in G_2 \setminus OPT$ then the two packets are the two packets mapped to q. Similarly, we associate a packet from OPT with each packet in $q \in G_3$ as follows. If $q \in G_3 \cap OPT$ then the this packet is q. If $q \in G_1 \setminus OPT$ then this packet is the packet mapped to q. Note that this way we associated every packet in OPT with some packet in $GREEDY$. Note that always $v(q)$ is 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 $|GREEDY| \geq |OPT|$. 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

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

∎

Proof of Theorem 3.5: Modify the mapping defined in the proof of Theorem 3.1 as follows. For each time t if $DROP_t \neq \emptyset$ decrease by 1 the number of availabilities contributed by the packet p transmitted by the greedy algorithm at time t. Specifically, if $p \in OPT$ then no packet is mapped to p and if $p \notin OPT$ at most one packet is mapped to p. We need to show that Claim 3.3 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 are mapped is at least $\sum_{i=r(t)}^{s(t)} |DROP_i|$. Notice that the number of availabilities is decreased by at most $s(t) - r(t) + 1$. Claim 3.3 follows for the modified mapping since

$$\sum_{i=r(t)}^{s(t)} |DROP_i| \leq s(t) - r(t) + 2B + 1 - x \leq 2(s(t) + B - r(t) + 1) - x - (s(t) - r(t) + 1) .$$

Notice that the value of the packet transmitted at time t is at least the value of any packet in $DROP_t$. We "shift" $1/(|DROP_t| + 1)$ fraction of the value of each of the packets in $DROP_t$ from the packet to which it has been mapped originally and associate it with the packet transmitted at t. Since $|DROP_t| \leq B$, we get that each packet transmitted by the greedy is associated with at most $1 + \frac{B}{B+1} = 2 - \frac{1}{B+1}$ packets of at most the same value from OPT. ∎

Proof of Theorem 3.6: Consider an input sequence. We prove that the total value of packets transmitted by GH is at least the total value of packets transmitted by G. We start with the following simple property.

Lemma A.1 *For any time step t, $|Q_{GH}(t)| = |Q_G(t)|$.*

Proof: Follows from the fact that for all t, $|D_G(t)| = |D_{GH}(t)|$ and $|S_G(t)| = |S_{GH}(t)|$. ∎

For each time step t we define 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 imply 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 value order. The existence of the mappings imply that for each time step t, the value of 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 $\sum_t v(S(t))$ we get

$$\sum_t v(S_{GH}(t)) = \sum_t v(A(t)) - \sum_t v(D_{GH}(t)) \geq \sum_t v(A(t)) - \sum_t v(D_G(t)) = \sum_t v(S_G(t)) \,.$$

All left to be proven is the existence of the mappings. Fix a time step t, and consider $Q_G(t)$ and $Q_{GH}(t)$. We define the following 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 we use the following notation.

Definition A.1 *Let A be an algorithm. For $p \in Q_A(t)$ define the* height *of p, denoted $h_A(p,t)$, to be 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. A packet $p \in Q_A(t)$ is said to be* below (above) *a packet $p' \in Q_A(t)$ if $h_A(p,t) < h_A(p',t)$ ($h_A(p,t) > h_A(p',t)$).*

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 so far. 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 A.2 *Let $p \in Q_{GH}(t) \setminus Q_G(t)$ for some time t. Then $v(p) \leq v(q)$, for all $q \in \{Q_G(t) : h_G(q,t) \leq h_{GH}(p,t)\}$.*

Proof: Let t_0 be the time in which G dropped p. We prove the lemma by induction on $t - t_0$. For the base case $t = t_0$, we have that $v(p) \leq \min\{v(p') : p' \in Q_G(t_0)\}$ by the greedy rule. For the inductive step, let $t > t_0$, and consider a packet $p' \in Q_G(t)$ with $h_G(p',t) \leq h_{GH}(p,t)$. If $h_G(p',t-1) \leq h_{GH}(p,t-1)$, we are done by induction. Otherwise, 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 A.3 *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 the packet with the minimal height such that $h_G(p,t) < h_{GH}(p,t)$. Let p' be the packet directly below p in $Q_{GH}(t)$. 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 have also $h_G(p',t) < h_{GH}(p',t)$, 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 $v(q) \geq v(p')$. In case $q = p'$ this is trivial. Otherwise, we have that $p' \notin Q_G(t-1)$

and $h_{GH}(p', t-1) \geq h_G(q, t-1)$, then by Lemma A.2 $v(p') \leq v(q)$. Since $Q_G(t)$ has more packets above p than $Q_{GH}(t)$, 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 are in $Q_{GH}(t')$, and $h_{GH}(q', t') > h_{GH}(p', t')$. ∎

We now go back to the existence proof 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 show that $h_{GH}(p, t) \geq h_G(q, t)$. By Lemma A.2 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 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 A.3. ∎

Proof of Theorem 3.7: Consider the following sequence. At time 0, $B/2 + 2$ packets of value ϵ arrive, for some negligible ϵ. At time 1, $B/2$ packets of value 1 arrive. Thus, at time 2, the head half of the buffer is filled with light packets, and the tail half of the buffer is filled with heavy packets. At time 2, $B/2 + 1$ more ϵ-value packets arrive, and at time $2 + B/2$, $B + 1$ packets of value 1 arrive. TD drops at time 2 the last $B/2$ light packets, and transmits a light packet in steps $2, 3, \ldots, 1 + B/2$. At time $2 + B/2$, TD drops $B/2$ heavy packets, and the total value eventually transmitted is $B + 1$. HD drops at time 2 the first $B/2$ light packets, and then drops at time $2 + B/2$ the other $B/2$ light packets, for a total transmitted value of $3B/2 + 1$. ∎

B Proofs from Section 4

Proof of Theorem 4.2: We first show that the greedy algorithm is at most 2 competitive and then show that it is at least 2 competitive, which establishes the theorem. Fix the input sequence, and let W be the output link bandwidth. Consider any optimal algorithm for the sequence. Let S_O and S_G denote the set of packets sent by an optimal algorithm and the greedy algorithm, respectively. Consider all the packets in $S_O \setminus S_G$. These are packets that were transmitted by the optimal algorithm but dropped by the greedy algorithm. Consider all packets in this set in turn. Let $p \in S_O \setminus S_G$ and suppose that packet p was transmitted by the optimal algorithm at time t. We match p with a packet p' transmitted by the greedy algorithm at time t and not yet matched. Note that there is always a such a packet $p' \in S_G(t)$ since $|S_O(t)| \leq W$ and $|S_G(t)| = W$. This is because by the greedy rule if $|S_G(t)| < W$ then the greedy algorithm should have transmitted p as well at time t. Moreover, we claim that $v(p) \leq v(p')$, since otherwise, the greedy algorithm would have transmitted p at time t instead of p'. It follows that $v(S_O \setminus S_G) \leq v(S_G)$, and hence

$$v(S_O) = v(S_O \cap S_G) + v(S_O \setminus S_G) \leq 2v(S_G) .$$

This proves that the greedy algorithm is at most 2 competitive.

We now establish that the greedy algorithm is at least 2 competitive. Let $\epsilon > 0$. We present a scenario in which the value transmitted by an optimal algorithm is $2 - \epsilon$ times the value transmitted by the greedy algorithm. The scenario is as follows. Let $n \geq 1$ be any integer. At the first time step, $2n$ packets arrive, partitioned into two subsets A and B, containing n packets each. All packets $p \in A$ have $v(p) = 1$ and $sl(p) = 2n$, and all packets $p' \in B$ have $v(p') = 1 - \epsilon$ and $sl(p') = n$. No more packets arrive in the first $2n$ steps. The greedy algorithm transmits all packets in A in the first n steps (since their value is greater), thus loosing all packets in B and getting total transmitted value of n. The optimal algorithm in this case transmits in the first n steps all packets in B, and in the following n steps all packets in A, for a total of $(2 - \epsilon)n$ transmitted value. This scenario can be repeated any number of times. ∎

Proof of Theorem 4.1: Let A be any on-line algorithm. Consider the following scenario. At time $t = 1$ the buffer is empty and δ packets of value 1 arrive. During each of the first δ time units, $(t = 1, \ldots, \delta)$ a single packet of value $\alpha > 1$ arrives. Let $\delta - x$ be the number of value 1 packets transmitted by A by (and including) time δ. (We assume that $x \geq 1$.) 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 $\delta + 1$. In the former case, the value of A is at most $\delta - x + \delta\alpha$, while the optimal value is $\delta - 1 + \delta\alpha$. In the latter case, the value of A is $\delta - x + \alpha(\delta + x)$, while the optimal value is $2\delta\alpha$. Consider the two possible ratios. To get the lower bound our goal is to fix α so that the value of the maximum of the two ratios for any value of x is maximized. This is because the online 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 the online algorithm is the one where the two ratios are equal. First, it is easy to see that for any $\alpha > 1$ the two ratios are always bounded away from 1. When δ tends to infinity, define $x' = x/\delta$, and we get that the two ratios tend to $\frac{\alpha+1}{\alpha+1-x'}$ and $\frac{2\alpha}{(1+x')\alpha+1-x'}$. Consider the point where the two ratios are equal and maximize for α. We get that at this point $x' = \frac{\alpha^2-1}{\alpha^2+2\alpha-1}$. Substituting for x' we get that the ratio is $1 + \frac{\alpha-1}{\alpha(\alpha+1)}$. The maximum of this ratio is given when $\alpha = 1 + \sqrt{2}$, and the ratio is approximately $1 + \frac{1}{(1+\sqrt{2})^2} \approx 1.17$. ∎

Proof of Theorem 4.3: Fix the input sequence, and let W be the output link bandwidth. Consider any optimal algorithm for the sequence. Let S_O^{val} and S_G^{val} denote the set of packets of value val sent by an optimal algorithm and the greedy algorithm, respectively. 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. Since each high value packet may cause loss of at most one low value packet with earlier deadline we obtain that $v(S_O^1) - v(S_G^1) \leq v(S_G^\alpha)/\alpha$. The theorem follows. ∎

Proof Sketch of Theorem 4.4: Without loss of generality we may assume that the optimal algorithm schedules packets in order of non-decreasing deadline. We consider the schedule of the ϕ-Ratio EDF algorithm at time t. At time t let A be the set of packets with deadline t or later, which where not yet transmitted, and let S_1 and S_2 be the set of packets, not yet transmitted, with deadline time t and $t+1$, respectively. Clearly, when the ϕ-Ratio EDF selects a packet from S_1 which is among the W most valuable packets in A, its decision is optimal. The ϕ-Ratio EDF algorithm may err in two cases.

1. The ϕ-Ratio EDF schedules at time t a packet $p \in S_1$ which is not among the W most valuable packets in A. In this case OPT can schedule instead of p a packet $p' \in S_2$ whose value is at most ϕ times greater than the value of p.

2. The ϕ-Ratio EDF schedules a packet $p \in S_2$ which is among the W most valuable packets in A. In this case OPT can schedule in addition to p a packet $p' \in S_1$ whose value is at most the value of p divided by ϕ.

In the former case the competitive ratio is bounded by ϕ while in the latter case the competitive ratio is bounded by $(\phi + 1)/\phi$. The theorem follows since $\phi = (\phi + 1)/\phi$. ∎

Proof of Theorem 4.5: 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 $x \leq 1$ 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 $x \leq 1$ 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 the bound is $1+\frac{1}{(1+\sqrt{2})^2} \approx 1.17$. ▮

Proof of Theorem 4.6: We prove the bound for the $(1.5+\sqrt{13}/2)$-Ratio EDF algorithm, the proof for the Greedy EDF algorithm is similar.

Without loss of generality we may assume that the optimal algorithm schedules packets in order of non-decreasing deadline. We consider the schedule of the β-Ratio EDF algorithm.

At time t let S_1^t and S_2^t be the set of packets, not yet transmitted, with deadline time t and $t+1$, respectively. When clear from the context we drop the superscript t. (Note that since $W = 1$ we have that S_1^t may include at most one packet and S_2^t may include at most two packets.)

The outline of the proof is as follows. We consider the schedules of optimal algorithm and the online algorithm. We divide the time into intervals of two forms: (i) intervals in which both the optimal algorithm and the online send the same packets, i.e., they are identical in the interval, and (ii) intervals in which the optimal algorithm and the online algorithm do not send the same packets. Clearly, in the intervals of the first type the online is optimal, therefore we need only to bound the ratio of the values for intervals of the second type. From now on we will consider only intervals of the second type.

Let $S_G(t)$ and $S_O(t)$ be the packets that the online and optimal algorithms send at time t. We identify the intervals of the second type by their start point.

The first possibility is $p_2 = S_G(t) \in S_2$ and $p_1 = S_O(t) \in S_1$. Note that $v(p_2)/v(p_1) \geq \beta$. We consider the maximum length interval until we reach a time t' in which one of the three conditions occur: (1) both the online algorithm and the offline algorithm send the same packet; (2) the online algorithm sends a packet at its deadline; and (3) the online algorithm has no packet to send. Until time t' the online algorithm sends packets of maximal value one time unit before their deadline (i.e., at the time they arrive) and optimal algorithm sends the same packets at their deadline. In this interval the ratio of the value of the optimal to the online algorithms is at most $(1+\beta)/\beta$.

The second possibility is $p_1 = S_G(t) \in S_1$ and $p_2 = S_O(t) \in S_2$. Note that $v(p_2)/v(p_1) < \beta$. Consider the interval until time t' in which one of the two conditions occur: (1) the optimal algorithm sends a packet at its deadline; (2) the online algorithm sends a packet before its deadline. Observe that at any time slot i in the interval it holds that the value of $S_O(i)$ is at most the value of $S_G(i+1)$. If condition (2) is satisfied before condition (1), then $S_G(t') = S_O(t') \geq S_O(t'-1)/\beta$. In this case the ratio is bounded again by $(1+\beta)/\beta$. In case condition (1) is satisfied we first note that the length on the interval is at least three. To see this suppose that the interval has length 2. This means that the optimal algorithm sends at time $t+1$ a packet that arrived at time t. However, this packet was available also to the online algorithm at time t. The fact that it was not sent by the online Greedy EDF algorithm implies that the value of the optimal algorithm can be improved by sending the packet sent by the online algorithm instead of the packet $S_O(t+1)$; a contradiction to the optimality.

Consider the last three packets sent by the optimal algorithm in the interval $S_O(t'-2)$, $S_O(t'-1)$ and $S_O(t')$. Note that the deadline of $S_O(t'-2)$ is $t'-1$ and the deadlines of $S_O(t'-1)$ and $S_O(t')$ is t'. Note also that these three packets were available to the online algorithm at time $t'-1$.

We now distinguish between two subcases. In the first, the online algorithm transmits at t' a packet at its deadline (i.e., either $S_O(t'-1)$ or $S_O(t')$). In this subcase by the definition of the Greedy EDF algorithm the packets $S_G(t'-1)$ and $S_G(t')$ are the two packets with the largest value among $\{S_O(t'-2), S_O(t'-1), S_O(t')\}$. Finally, $v(S_G(t'-2)) \geq v(S_G(t'-1))/\beta$, since otherwise the online algorithm would have dropped it. Without loss of generality assume that $v(S_G t'-2) = 1$, and let X be the value of the packets $S_G(t)$ to $S_G(t'-2)$. This implies that the value of the optimal algorithm is at most $X+3\beta$, and the value of the online is at least $X+2\beta$. Since $X \geq 1$ we have that the ratio is bounded by $(1+3\beta)/(1+2\beta)$.

In the second subcase the online algorithm transmits at t' a packet with deadline $t'+1$. This happens only in case the value of $S_G(t')$ is β times larger than the value of the packet that would have been transmitted by the online algorithm at time t' had the first subcase held. In this subcase we extend the interval until the first time after t' in which (1) both the online algorithm and the offline algorithm send the same packet; (2) the online algorithm sends a packet at its deadline; and (3) the online algorithm has no packet to send. Following arguments similar to the previous subcase we get that in this subcase the ratio is bounded by $(1+3\beta+\beta^2)/(1+\beta+\beta^2)$.

We have three bounds $(1+\beta)/\beta$, $(1+3\beta)/(1+2\beta)$ and $(1+3\beta+\beta^2)/(1+\beta+\beta^2)$. We need to find β that minimizes the maximum of the three bound. Consider first the last two bounds. We get that the optimal β for them is $1.5+\sqrt{13}/2 \approx 3.3$. It is easy to see that for this value of β the first bound is less than the rest, and we get that the competitive ratio is at most 1.43. ∎

Proof of Theorem 4.7: We first establish the bound in the uniform model and then in the variable model. Fix an on-line algorithm A, and consider the following scenario. Initially, at time 0, the buffer is empty and two packets of value 1 arrive. At the next time a packet of value $\alpha > 1$ arrives. If A drops one of the low value packets then its competitive ratio is bounded from below by $\frac{\alpha+2}{\alpha+1}$ since there exists a feasible schedule of all three packets. If both low value packets are scheduled then at time 2 two additional packets of value α arrive. Thus, at least one packet of value α is necessarily lost by A. In this case the competitive ratio of A is bounded from below by $\frac{3\alpha+1}{2\alpha+2}$.

Therefore, the competitive ratio of A is at least $\min\{\frac{\alpha+2}{\alpha+1}, \frac{3\alpha+1}{2\alpha+2}\}$. To optimize the lower bound, we solve the quadratic equation $\frac{\alpha+2}{\alpha+1} = \frac{3\alpha+1}{2\alpha+2}$. We get that $\alpha = 3$ and the competitive ratio of A is at least 1.25.

We now establish the bound in the variable model. Fix an on-line algorithm A, and consider the following scenario. Initially, 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. If A drops the low value packet then its competitive ratio is bounded from below by $\frac{\alpha+1}{\alpha}$ since there exists a feasible schedule of both packets. If the low value packet is scheduled then at time 1 an additional packet of value α with delay 1 (i.e., zero slack time) arrives. Thus, at least one high value packet is necessarily lost by A. In this case the competitive ratio of A is bounded from below by $\frac{2\alpha}{\alpha+1}$.

Therefore, the competitive ratio of A is at least $\min\{\frac{\alpha+1}{\alpha}, \frac{2\alpha}{\alpha+1}\}$. To optimize the lower bound, we solve the quadratic equation $\frac{\alpha+1}{\alpha} = \frac{2\alpha}{\alpha+1}$. We get that $\alpha = 1 + \sqrt{2}$ and the competitive ratio of A is at least $\sqrt{2}$. ∎

C Proofs from Section 5

Proof of Theorem 5.1: 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 $|S_A(t_0)| > |S_B(t_0)|$. It follows that $|Q_A(t_0-1)| > |Q_B(t_0-1)|$. Let $t_1 < t_0$ be the maximal time such that $Q_A(t_1-1)| \leq |Q_B(t_1-1)|$. For this choice of t_1, we have that $Q_B(t)$ is not full for all $t_1 \leq t \leq t_0$, and that $|D_B(t_1)| > |D_A(t_1)$. It follows that there exists a packet $p \in D_B(t_1) \setminus D_A(t_1)$, and that packet can be added to S_B while keeping the schedule feasible. ∎

Proof of Theorem 5.2: As above, the first two properties are trivial. So Let A and B be such that $|S_A| \geq |S_B|$. We need to find a packet $p \in S_A \setminus S_B$ such that $S_A \cup \{p\} \in \mathfrak{I}$. Let t_0 be the first time such that $|S_B(t_0))| < |S_A(t_0))|$. Let $p_0 \in S_B(t_0) \setminus S_A(t_0)$. We now define an inductive process for $i \geq 0$ as follows. If $p_i \notin S_B$, the process stops. Otherwise, we define t_{i+1} to be the time such that $p_i \in S_B(t_{i+1})$. Since t_0 was the first time that $|S_B(t_0))| > |S_A(t_0))|$, there must exist a packet $p_{i+1} \in S_A(t_{i+1}) \setminus S_B(t_{i+1})$. With these t_{i+1} and p_{i+1} we continue the process. Note that since $t_{i+1} < t_i$ for all i and since time is bounded from below, the process must stop, say with $i = j$ for some j. We now claim that $S_B \cup \{p_j\} \in \mathfrak{I}$. To see this, note that starting from the feasible schedule B, we can re-schedule p_i to be transmitted at time t_i for all $0 \leq i \leq j$: none of the deadlines is violated, since these are the time these packets are transmitted by A. ∎

Proof of Theorem 5.4: 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 $\delta = B$. First, note that any 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 $OPT_D \leq OPT_F$. For the other direction, consider any schedule in the uniform bounded-delay model. Since in this model, a set of packets is feasible if and only iff the EDF schedule 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 $OPT_F \leq OPT_D$. ∎