Comparison-based Buffer Management in QoS Switches*

Kamal Al-Bawani† Matthias Englert‡ Matthias Westermann§

Abstract

The following online problem arises in network devices, e.g., switches, with quality of service (QoS) guarantees. In each time step, an arbitrary number of packets arrive at a single buffer and only one packet can be transmitted. The differentiated service concept is implemented by attributing each packet with a non-negative value corresponding to its service level. The goal is to maximize the total value of transmitted packets.

We consider two models of this problem, the FIFO and the bounded-delay model. In the FIFO model, the buffer can store a limited number of packets and the sequence of transmitted packets has to be a subsequence of the arriving packets. In this model, a buffer management algorithm can reject arriving packets and preempt buffered packets. In the bounded-delay model, the buffer has unbounded capacity, but each packet has a deadline and packets can only be transmitted before their deadlines. Here, a buffer management algorithm determines the packet to be sent in each time step.

We study comparison-based buffer management algorithms, i.e., algorithms that make their decisions based solely on the relative order between packet values with no regard to the actual values. This kind of algorithm proves to be robust in the realm of QoS switches. Kesselman et al. (SIAM J. Comput., 2004) present two deterministic comparison-based online algorithms, one for each model, which are 2-competitive. For a long time, it has been an open problem, whether a comparison-based online algorithm exists, in either model, with a competitive ratio below 2.

In the FIFO model, we present a lower bound of $1 + 1/\sqrt{2} \approx 1.707$ on the competitive ratio of any deterministic comparison-based algorithm and give an algorithm that matches this lower bound in the case of monotonic sequences, i.e., packets arrive in a non-decreasing order according to their values. In the bounded-delay model, we show that no deterministic comparison-based algorithm exists with a competitive ratio below 2. In the special s-uniform case, where the difference between the arrival time and deadline of any packet equals s, we present a randomized comparison-based algorithm that is 5/3-competitive.

1 Introduction

We consider the following online problem which arises in network devices, e.g., switches, with quality of service (QoS) guarantees. In each time step, an arbitrary number of packets arrive at a single buffer. Each packet has a non-negative value attributing its service level (also known as class of service (CoS)). All packets have the same size and only one packet can be transmitted in each time step. The goal is to maximize the total value of transmitted packets.

We consider two models of this problem, the FIFO and the bounded-delay model. In the FIFO model, the buffer can store at most B packets and the sequence of transmitted packets has to be a subsequence of the arriving packets. In this model, a buffer management algorithm can reject arriving packets and preempt buffered packets. In the bounded-delay model, the buffer has unbounded capacity, but each packet has a deadline and packets can only be transmitted before their deadlines. Here, a buffer management algorithm determines the packet to be sent in each time step.

*The second and third author's work was supported by ERC Grant Agreement No. 307696. A preliminary version appeared in Proceedings of the 12th Latin American Theoretical Informatics Symposium (LATIN), pages 27–40, 2016.

†Department of Computer Science, RWTH Aachen University, Germany kbawani@cs.rwth-aachen.de

‡DIMAP and Department of Computer Science, University of Warwick, UK englert@dcs.warwick.ac.uk

§Department of Computer Science, TU Dortmund, Germany matthias.westermann@cs.tu-dortmund.de

In probabilistic analysis of network traffic, packet arrivals are often assumed to be Poisson processes. However, such processes are not considered to model network traffic accurately due to the fact that in reality packets have been observed to frequently arrive in bursts rather than in smooth Poisson-like flows (see, e.g., [23, 26]).

We do not make any prior assumptions about the arrival behavior of packets, and instead resort to the framework of competitive analysis [25], which is the typical worst-case analysis used to assess the performance of online algorithms. Online algorithms receive the input sequence piece-by-piece over time, and the decisions they make in each time step are irrevocable. In competitive analysis, the benefit of an online algorithm is compared to the benefit of an optimal algorithm OPT, which is assumed to know the entire input sequence in advance. An online algorithm ONL is called *c-competitive* if, for each input sequence σ, the benefit of OPT over σ is at most c times the benefit of ONL over σ. The value c is also called the *competitive ratio* of ONL.

Comparison-based buffer management. In QoS networks, packet values are only an implementation of the concept of differentiated service. A packet value stands for the packet's service level, i.e., the priority with which this packet is transmitted, and does not have any intrinsic meaning in itself. However, just slight changes to the packet values, even though the relative order of their corresponding service levels is preserved, can result in substantial changes in the outcome of current buffer management algorithms. We aim to design new buffer management algorithms whose behavior is independent of how the service levels are implemented in practice. Therefore, we study *comparison-based* buffer management algorithms, i.e., algorithms that make their decisions based solely on the relative order between values with no regard to the actual values. Such algorithms are robust to order-preserving changes of packet values.

Consider the following GREEDY algorithm in the FIFO model: Accept any arriving packet as long as the queue is not full. If a packet arrives while the queue is full, drop the packet with the smallest value. Clearly, GREEDY is comparison-based, and Kesselman et al. [20] show it is 2-competitive. In the bounded-delay model, they show that the comparison-based algorithm that sends in each time step the packet with the greatest value (among all packets that have not expired yet) is also 2-competitive. For a long time, it has been an open problem whether a comparison-based online algorithm exists, in either model, with a competitive ratio below 2.

1.1 Related work

Azar and Richter [7] introduce the 0/1 principle for the analysis of comparison-based algorithms in a variety of buffering models. They show the following theorem.

Theorem 1. *Let* ALG *be a comparison-based switching algorithm (deterministic or randomized).* ALG *is c-competitive if and only if* ALG *is c-competitive for all input sequences of packets with values 0 and 1, under every possible way of breaking ties between equal values.*

In the FIFO model, GREEDY is the only deterministic comparison-based algorithm that has been analyzed. Kesselman et al. [20] show that the exact competitive ratio of GREEDY is $2 - 1/B$, where B is the size of the buffer. Andelman [4] employs the 0/1 principle to give a randomized comparison-based algorithm with a competitive ratio of 1.75.

In a related model with multiple FIFO queues, Azar and Richter [7] give a comparison-based deterministic algorithm with a competitive ratio of 3. In another related model, where the buffer is not FIFO and packet values are not known for the online algorithm, Azar et al. [6] use the 0/1 principle to show a randomized algorithm with a competitive ratio of 1.69. This algorithm is modified to a 1.55-competitive randomized algorithm, and a lower bound of 1.5 on the competitive ratio of any randomized algorithm is shown for that model [5].

Apart from comparison-based algorithms, the FIFO model has been extensively studied. Kesselman, Mansour, and van Stee [19] give the state-of-the-art algorithm PG, and prove that PG is 1.983-competitive. Additionally, they give a lower bound of $(1+\sqrt{5})/2 \approx 1.618$ on the competitive ratio of PG and a lower bound of 1.419 on the competitive ratio of any deterministic algorithm. Algorithm PG adopts the same preemption

strategy of GREEDY and moreover, upon the arrival of a packet p, it proactively drops the first packet in the queue whose value is within a fraction of the value of p. This additional rule makes PG non-comparison-based. Bansal et al. [8] slightly modify PG and show that the modified algorithm is 1.75-competitive. Finally, Englert and Westermann [15] show that PG is 1.732-competitive and give a lower bound of $1 + (1/\sqrt{2}) \approx 1.707$ on its competitive ratio.

In the case where packets take on only two values, 1 and $\alpha > 1$, Kesselman et al. [20] give a lower bound of 1.282 on the competitive ratio of any deterministic algorithm. Englert and Westermann [15] give an algorithm that matches this lower bound. In the non-preemptive model of this case, Andelman et al. [3] optimally provide a deterministic algorithm which matches a lower bound of $2 - 1/\alpha$ given by Aiello et al. [1] on the competitive ratio of any deterministic algorithm.

In the general-value case of the non-preemptive model, Andelman et al. [3] show a lower bound of $1 + \ln(\alpha)$ for any deterministic algorithm, where α is the ratio between the maximum and minimum packet values. This bound is achieved by a deterministic algorithm given by Andelman and Mansour [2].

The previous work in the bounded-delay model is just as extensive. In the case of arbitrary deadlines, the greedy algorithm that always sends the packet with the greatest value is 2-competitive [20]. No better comparison-based algorithm is known for this model. All the following work is concerned with non-comparison-based algorithms only. Chrobak et al. [13] are the first to break the barrier of 2-competitiveness with a 1.939-competitive algorithm. Li, Sethuraman, and Stein [22] give a better algorithm with a competitive ratio of 1.854. The best algorithm to date is given by Englert and Westermann and has a competitive ratio of 1.828 [16]. The best known randomized policy is given by Chin et al. [11] and has a competitive ratio of $e - 1 \approx 1.582$ against an oblivious adversary. Bienkowski et al. [10] show the same competitive ratio against an adaptive adversary.

The bounded-delay model has further been considered in the following restricted cases of deadlines.

- The s-bounded case in which the span of any packet, i.e., the difference between its arrival time and its deadline, is at most s.
- The agreeable-deadline case in which, for any two packets p and q, if p arrives before q, then the deadline of p is smaller than or equal to that of q. Notice that the 2-bounded case is a special case of agreeable deadlines.
- The s-uniform case where all packets have the same span of s. Clearly, this case is also a special case of agreeable deadlines.

The 2-bounded case is completely settled. For deterministic algorithms, Chin and Fung [12] show a lower bound of 1.618. Kesselman et al. [20] give an algorithm that matches this lower bound. For randomized algorithms, Chin and Fung also show a lower bound of 1.25 against an oblivious adversary. This lower bound is matched by a randomized algorithm given by Chin et al. [11]. Moreover, Bienkowski et al. [9] show a lower bound of 1.33 against an adaptive adversary and give a randomized algorithm with a matching competitive ratio.

As 2-bounded is a special case of general and agreeable deadlines, the deterministic lower bound of 1.618 holds also for those cases. Chin et al. [11] show that 1.618 is the optimal deterministic competitive ratio for the 3-bounded case as well and they show a 1.732-competitiveness for the 4-bounded case. Furthermore, they give a new algorithm that has a competitive ratio of $2 - \frac{2}{s} + o(\frac{1}{s})$ in the general s-bounded case.

In the case of agreeable deadlines, Li, Sethuraman and Stein [21] give a deterministic algorithm that achieves the optimal competitiveness of 1.618. Jeż et al. [18] provide a new randomized policy for this case that is 1.33-competitive against an oblivious adversary. The randomized lower bounds known for this case are implied from the 2-bounded case.

Most results on the s-uniform case focus on $s = 2$. Chrobak et al. [13] show that optimal deterministic competitive ratio is 1.377. In the case of memoryless algorithms, i.e., algorithms that do not use any information other than the current buffer contents, the optimal competitiveness is shown to be 1.414 [27, 11]. In the randomized case, Bienkowski et al. [9] show a lower bound of 1.2 against an adaptive adversary. As for upper bounds, the 2-bounded competitive ratios of 1.25 and 1.33 against oblivious and adaptive adversaries, respectively, carry over to the 2-uniform case.

Relationship between FIFO and s-uniform bounded-delay. Finally, while not much is known in the more general s-uniform case, it is worth noting that any upper bound from the FIFO model for buffer size s carries over to the s-uniform bounded-delay model. As a consequence, the 7/4-competitive randomized comparison-based algorithm which Andelman [4] designed for the FIFO model can also be used in the s-uniform bounded delay model. Specifically, we can take an input for the s-uniform model and require that arriving packets are placed in a queue of capacity s. At each send event, the packet at the front of the queue is transmitted. Packets can be removed from the queue at any time and from any position within the queue, but once removed, a packet will be lost forever. Note that packets no longer have explicit deadlines associated with them. Instead, the fact that the buffer has size s implicitly ensures that no packet is transmitted more than s steps after it has arrived.

With this transformation, any online algorithm for the FIFO model can also be used for the s-uniform model. Notice that the benefit of an optimal offline algorithm OPT is the same in both models. Therefore, any upper bound on the competitive ratio of the former model carries over to the latter. In general, the converse is not necessarily true. However, in Section 3, we will discuss an online algorithms for the s-uniform bounded-delay model that has an equivalent version in the FIFO model.

The problem of online buffer management has also been studied under several other models. A comprehensive survey on this problem and most of its variants is given in [17].

1.2 Our results

In the FIFO model, we present a lower bound of $1+1/\sqrt{2} \approx 1.707$ on the competitive ratio of any deterministic comparison-based algorithm. This lower bound is significantly larger than the lower bound of 1.419 known for general deterministic algorithms. The lower bound construction only uses *monotonic sequences*, i.e., packets arrive in a non-decreasing order according to their values. We also give an algorithm, CPG, that matches this lower bound, for large B, in the case of monotonic sequences. Note that GREEDY remains 2-competitive in the case of monotonic sequences.

An intriguing question in this respect is whether a comparison-based algorithm with a competitive ratio close to $1 + 1/\sqrt{2} \approx 1.707$ could exist for more general packet sequences. If so, this would mean that we do not need to know the actual values of packets in order to compete with PG, the best non-comparison-based algorithm so far. If not, and in particular if 2 is the right lower bound for any comparison-based algorithm, the desired robustness of this kind of algorithm must come at a price, namely, a significantly degraded performance.

In the bounded-delay model, we first present a lower bound of 2 for any deterministic comparison-based algorithm in the 2-bounded case. Clearly, the 2-bounded case is a special case of the s-bounded case as well as the cases of agreeable and general deadlines. Thus, this lower bound holds in all these settings.

In the s-uniform case, we show that the greedy algorithm which always sends the packet with the greatest value and the algorithm which always sends the packet with the earliest deadline have both a competitive ratio of at least $2 - 1/s$. We then present a randomized comparison-based algorithm, RAND, that tosses a coin at the beginning, then sticks to one of these two algorithms until the end. This kind of algorithm is called *barely randomized* [24]. We show that RAND is 5/3-competitive.

1.3 Models and notations

We consider a single buffer that stores packets of unit size. Each packet p is associated with a non-negative value, denoted by $v(p)$, that corresponds to its level of service.

Time is discretized into slots of unit length. An arbitrary number of packets arrive at fractional (non-integral) times, while at most one packet is sent from the queue, i.e., transmitted, at every integral time, i.e., at the end of each time slot. We denote the arrival time of a packet p by $\mathrm{arr}(p)$. An arriving packet is either inserted into the queue or it is rejected, and an enqueued packet may be dropped from the queue before it is sent. The latter event is called preemption. Rejected and preempted packets are lost.

We denote the arrival of a new packet as an arrival event, and the sending of a packet as a send event. An input sequence σ consists of arrival and send events. The time that precedes the first arrival event of the sequence is denoted as time 0. We assume that the queue of any algorithm is empty at time 0.

The benefit that an algorithm ALG makes on an input sequence σ is denoted by $\text{ALG}(\sigma)$, and is defined as the total value of packets that ALG sends. We aim to maximize this benefit. We denote by OPT an optimal algorithm that sends packets offline.

In the FIFO model, the buffer, sometimes referred to as the queue, can store at most B packets at any time, and packets must be sent in the order of their arrival. In contrary to that, the capacity of the queue is not limited in the bounded-delay model and packets can be sent in any order. However, packets are restricted by deadlines in this model, i.e., each packet p has an integral deadline $d(p) \geq \text{arr}(p)$, such that p is lost if it is not sent before or at $d(p)$. The benefit of an online algorithm ALG is in this case the total value of packets that ALG sends before or at their deadlines.

2 The FIFO model

2.1 Lower bound

First, we present a lower bound of $1 + 1/\sqrt{2} \approx 1.707$ on the competitive ratio of any deterministic comparison-based algorithm. Recall that the best lower bound for general deterministic algorithms is 1.419.

Theorem 2. *The competitive ratio of any deterministic comparison-based algorithm is at least* $1 + 1/\sqrt{2} \approx 1.707$.

Proof. Fix an online algorithm ONL. The adversary constructs a sequence of packets with non-decreasing values over a number of iterations. The 0/1 values corresponding to the packets' real values are revealed only when the sequence stops. In each iteration, the adversary generates a burst of B packets in one time slot followed by a number of individual packets, each in one time slot. We call a slot with B arrivals a *bursty* slot, and a slot with one arrival a *light* slot. A construction routine is repeated by the adversary until the desired lower bound is obtained. For $i \geq 0$, let f_i denote the i-th bursty slot, and let t_i denote the number of time slots (i.e. send events) that ONL takes to send and preempt all packets that it has stored in the queue right before the end of slot f_i, i.e., right before the send event of slot f_i.

As initialization, the adversary generates B packets in the first time slot. Thus, the first slot is f_0. After that, the adversary generates t_0 light slots, i.e., one packet arrives in each slot. Now, starting with $i = 0$, the adversary constructs the rest of the sequence by the following routine which is repeated until $t_i \geq B/\sqrt{2}$.

1. Generate the bursty slot f_{i+1}.
2. If $t_i \geq B/\sqrt{2}$, stop the sequence. At this point, all packets that arrive between f_0 and f_i (inclusive) are revealed as 0-packets and all packets after that are revealed as 1-packets, i.e., the 1-packets are those which arrive in the t_i light slots and in the bursty slot f_{i+1}. Clearly, the optimal algorithm, denoted as OPT, will send all the 1-packets while ONL will gain only the B 1-packets which it has in slot f_{i+1}. Notice that ONL sends only 0-packets in the t_i light slots. Hence, provided that $t_i \geq B/\sqrt{2}$,
$$\begin{aligned} \frac{\text{OPT}}{\text{ONL}} &= \frac{t_i + B}{B} \\ &\geq \frac{B/\sqrt{2} + B}{B} \quad . \end{aligned}$$
3. If $t_i < B/\sqrt{2}$, continue the sequence after f_{i+1} by generating t_{i+1} light slots.
   (a) If $t_{i+1} \leq t_i$, stop the sequence. At this point, all packets that arrive between f_0 and f_i (inclusive) are revealed as 0-packets and all packets after that are revealed as 1-packets, i.e., the 1-packets are those which arrive in the t_i and t_{i+1} light slots and in the bursty slot f_{i+1}. Clearly, OPT will

send all the 1-packets while ONL will send only t_{i+1} packets of the B 1-packets which it has in slot f_{i+1} and also the t_{i+1} 1-packets which it collects after f_{i+1}. Hence, provided that $B > \sqrt{2} \cdot t_i$ and $t_i \geq t_{i+1}$,

$$\begin{aligned}\frac{\text{OPT}}{\text{ONL}} &= \frac{t_i + B + t_{i+1}}{t_{i+1} + t_{i+1}} \\ &\geq \frac{t_i + \sqrt{2} \cdot t_i + t_{i+1}}{2 \cdot t_{i+1}} \\ &\geq \frac{(1+\sqrt{2})t_{i+1} + t_{i+1}}{2 \cdot t_{i+1}} \quad .\end{aligned}$$

(b) If $t_{i+1} > t_i$, set $i = i + 1$ and repeat the routine.

Obviously, the above routine terminates eventually, because a new iteration is invoked only when $t_{i+1} > t_i$, and thus the amount of t_i is strictly increased in each iteration. Therefore, there must exist i such that $t_i \geq B/\sqrt{2}$. □

2.2 Algorithm CPG

We present a comparison-based preemptive greedy (CPG) algorithm. This algorithm can be seen as the comparison-based version of the well-studied algorithm PG [19] and also has a parameter β. It follows a similar rule of preemption as PG, but without addressing the actual values of packets: Roughly speaking, once you have a set S of β packets in the queue with a packet r in front of them, such that r is less valuable than each packet in S, preempt r.

CPG is described more precisely in Algorithm 1. To avoid using the same set of packets to preempt many other packets, it associates with each arriving packet p a non-negative *credit*, denoted by $c(p)$. For a set S of packets, $c(S)$ will also denote the total credit of all packets in S. We now describe the above preemption rule in more details.

First, we present the notations of *preemptable* packets and *preempting sets*. Assume that a packet p arrives at time t. Let $Q(t)$ be the set of packets in CPG's queue immediately before t. For any packet $r \in Q(t)$, if there exists a set $S \subseteq (Q(t) \cup \{p\}) \setminus \{r\}$ such that (i) $p \in S$, (ii) $c(S) \geq \beta$, and (iii) for each packet $q \in S$, $\text{arr}(q) \geq \text{arr}(r)$ and $v(q) \geq v(r)$, then we say that r is *preemptable* by p. Furthermore, we call S a *preempting* set of r.

When a packet p arrives, CPG preempts a packet r, such that r is the first packet in the queue (in the FIFO order) that is preemptable by p. After r is preempted, CPG invokes a subroutine CHARGE to deduct a total of β units from the credits of the preempting set of packets of r. This charging operation can be done arbitrarily, but subject to the non-negative constraint of credits, i.e., $c(p) \geq 0$, for any packet p. After that, the algorithm proceeds similarly to GREEDY: It inserts the arriving packet p into the queue if the queue is not full or p is more valuable than the packet with the least value in the queue. In the latter case, the packet with the least value is dropped. Otherwise, p is rejected. Finally, in send events, CPG simply sends the packet at the head of the queue.

Notice that CPG is a comparison-based algorithm. Hence, by Theorem 1, it is sufficient to show the competitive ratio of CPG for only 0/1 sequences. We denote a packet of value 0 as 0-packet, and of value 1 as 1-packet.

Lost packets. We distinguish between three types of packets lost by CPG:

1. Rejected packets: An arriving packet p is rejected if the queue is full and no packet in the queue is less valuable than p.

2. Evicted packets: An enqueued packet q is evicted by an arriving packet p if the queue is full and q is the least valuable among p and the packets in the queue.

Algorithm 1: CPG

arrival event. A packet p arrives at time t:

$c(p) \leftarrow 1$;

let r be the first packet in the queue such that r is *preemptable* by p;

if r *exists* **then**
  let S be a *preempting set* of r;
  drop r;
  CHARGE(S);

if *the queue is not full* **then**
  insert p;
else
  let q be the packet with the smallest value in the queue;
  if $v(q) < v(p)$ **then**
    drop q and insert p;
  else
    reject p;

3. Preempted packets: An enqueued packet r is preempted upon the arrival of another packet p if r is the first packet in the queue such that r is preemptable by p.

Notice that a 1-packet can only be evicted by a 1-packet. Also, if a 1-packet q is preempted, the preempting set of packets of q are all 1-packets.

In the following, we analyze the performance of CPG for monotonic input sequence. In a monotonic input sequence, packets arrive with non-decreasing values, i.e., for any two packets p and q, $v(p) \leq v(q)$ if p arrives before q. We observe that the 2-competitive greedy algorithm from [20] remains 2-competitive in this case.

Theorem 3. *Choosing $\beta = \sqrt{2}+1$, the competitive ratio of* CPG*, for $B \to \infty$, is at most $1 + 1/\sqrt{2} \approx 1.707$ for monotonic input sequences.*

Following the principle of Theorem 1, we restrict our analysis to *monotonic* 0/1 sequences. In particular, such a sequence has the property that whenever a comparison-based algorithm compares the value of a packet p with the value of a packet q, which arrived after p, packet q will be shown to have the larger value.

From now on, let $Q(t)$ (resp. $Q^*(t)$) denote the set of 1-packets in the queue of CPG (resp. OPT) at time t.

Assumptions on the optimal and the online algorithm. Notice that OPT, in contrast to CPG, can determine whether a packet of σ has value 0 or 1. Therefore, we can assume that OPT accepts arriving 1-packets as long as its queue is not full, and rejects all 0-packets. In send events, it sends 1-packets (in FIFO order) unless its queue is empty.

We further assume that no packets arrive after the first time in which the queue of CPG becomes empty. This assumption is also without loss of generality as we can partition σ into phases such that each phase satisfies this assumption and the queues of CPG and OPT are both empty at the start and the end of the phase. Then, it is sufficient to show the competitive ratio on any arbitrary phase. Consider for example the creation of the first phase. Let t be the first time in which the queue of CPG becomes empty. We postpone the packets arriving after t until OPT's queue is empty as well, say at time t', so that OPT and CPG are both empty at t'. This change can only increase the benefit of OPT. Clearly, t' defines the end of the first phase, and the next arrival event in σ defines the start of the second phase. The remaining of σ can be further partitioned in the same way.

Overflow time slot. We call a time slot in which CPG rejects or evicts at least one 1-packets an overflow time slot. Assume for the moment that at least one overflow time slot occurs in σ. For the rest of the analysis, we will use f to denote the last overflow slot, and t_f to denote the time immediately before this slot ends, i.e., after all arrival events of this slot take place and right before its send event. Obviously, rejection and eviction of 1-packets can happen only when the queue of CPG is full of 1-packets. Let t'_f be the point of time immediately before the first rejection or eviction in f takes place. Thus, the number of 1-packets in the queue at time t'_f is B. Thereafter, between t'_f and t_f, any 1-packet that is evicted or preempted is replaced by the 1-packet whose arrival invokes that eviction or preemption. Thus, the size of the queue does not change between t'_f and t_f, and hence the following observation.

Observation 4. $|Q(t_f)| = B$.

Furthermore, the following lemma shows that the B 1-packets in the queue at time t_f can be used to preempt at most one 1-packet in later arrival events.

Lemma 5. *Let t be the time immediately after an event e and let $D(t)$ denote the set of packets in the queue at time t except the head packet. Then, $c(D(t)) < \beta$.*

Proof. Let t' be a time right before the event e. For monotonic input sequences, the head packet is always the least valuable packet in the queue. Hence, if $c(D(t')) \geq \beta - 1$ and a packet arrives, the head packet of the queue is preempted. While the total amount of credit in the queue may increase by 1 due to the credit of the newly arriving packet, it will also decrease by $\beta > 1$ due to the preemption, resulting in an overall decrease. If $c(D(t')) < \beta - 1$ and a packet arrives, $c(D(t'))$ can increase by at most 1. Thus, we conclude $c(D(t)) < \beta$. Also, sending a packet can only decrease $c(D(t'))$. □

Before we proceed, we introduce further notations. Let $\textsc{arr}(t, t')$ denote the set of 1-packets that arrive in σ between time t and t', inclusive. Furthermore, let $\textsc{sent}(t, t')$ and $\textsc{lost}(t, t')$ denote the set of 1-packets that CPG sends and loses, respectively, between time t and t', inclusive. Similarly, we define $\textsc{sent}^*(t, t')$ and $\textsc{lost}^*(t, t')$ for OPT.

Lemma 6. *It holds that*

$$|\textsc{lost}(0, t_f)| - |\textsc{lost}^*(0, t_f)| + |Q(t_f)| - |Q^*(t_f)| = |\textsc{sent}^*(0, t_f)| - |\textsc{sent}(0, t_f)| \ .$$

Proof. The lemma follows from this simple observation:

$$\begin{aligned} &|Q(t_f)| + |\textsc{sent}(0, t_f)| + |\textsc{lost}(0, t_f)| \\ &\quad = |\textsc{arr}(0, t_f)| \\ &\quad = |Q^*(t_f)| + |\textsc{sent}^*(0, t_f)| + |\textsc{lost}^*(0, t_f)| \ . \end{aligned}$$ □

The following lemma is crucial for the analysis of CPG. It essentially upper-bounds the number of 1-packets that CPG loses between the start of the sequence and the end of the overflow slot.

Lemma 7. $|\textsc{lost}(0, t_f)| - |\textsc{lost}^*(0, t_f)| + |Q(t_f)| - |Q^*(t_f)| \leq \frac{\beta}{\beta+1} B$.

Proof. First, we present further notations. If an algorithm ALG does not send anything at integral time t, we say that ALG sends a $\emptyset$-packet (or *empty packet*) in t. We call a send event in which OPT sends an x-packet and CPG sends a y-packet an x/y send event, where x and y take on values from $\{0, 1, \emptyset\}$. Furthermore, we denote by $\delta_{x/y}(t, t')$ the number of x/y send events that occur between time t and time t', inclusive.

Now, observe that

$$\begin{aligned} |\textsc{sent}^*(0, t_f)| &= \delta_{1/0}(0, t_f) + \delta_{1/1}(0, t_f) + \delta_{1/\emptyset}(0, t_f) \ , \\ |\textsc{sent}(0, t_f)| &= \delta_{0/1}(0, t_f) + \delta_{1/1}(0, t_f) + \delta_{\emptyset/1}(0, t_f) \ . \end{aligned}$$

Recall that OPT does not send 0-packets and that, by assumption, the queue of CPG does not get empty before t_f. Thus, $\delta_{0/1}(0, t_f) = \delta_{1/\emptyset}(0, t_f) = 0$, and therefore

$$|\text{SENT}^*(0, t_f)| - |\text{SENT}(0, t_f)| = \delta_{1/0}(0, t_f) - \delta_{\emptyset/1}(0, t_f) \leq \delta_{1/0}(0, t_f) \ .$$

Hence, given Lemma 6, it suffices to show that $\delta_{1/0}(0, t_f) \leq \lfloor \frac{\beta}{\beta+1} B \rfloor$.

Assume for the sake of contradiction that $\delta_{1/0}(0, t_f) > \lfloor \frac{\beta}{\beta+1} B \rfloor$. Let M_1 (resp. M_0) be the set of 1-packets (resp. 0-packets) that OPT (resp. CPG) sends in these 1/0 send events. Thus,

$$|M_1| = |M_0| \geq \left\lfloor \frac{\beta}{\beta+1} B \right\rfloor + 1 > \frac{\beta}{\beta+1} B \ . \tag{1}$$

Let p (resp. q) denote the first arriving packet in M_1 (resp. M_0). Furthermore, let r be the last arriving packet in M_0 and denote the time in which it is sent by t_r. Recall that σ is monotonic. Thus, all the 1-packets of M_1 arrive after r. Moreover, since CPG's buffer is FIFO, none of these 1-packets is sent before t_r. Also, since r, which is a 0-packet, is before them in the queue and is eventually sent, CPG does not either reject, evict or preempt any 1-packet from M_1 before t_r. Therefore, all the 1-packets of M_1 must be in the queue of CPG at time t_r.

Let's now look closely at the queue of CPG immediately after the arrival of p. Let that time be denoted as t_p. Since q is sent with p in the same 1/0 send event and since r is between q and p (by the above argument), q and r must be in the queue as well at time t_p. Moreover, since r is the last arriving 0-packet in M_0, the remaining 0-packets of M_0 must also be in the queue at t_p. Hence, the queue of CPG contains all the packets of M_0 along with p at time t_p.

Next, notice that all the 1-packets of M_1 are inserted in CPG's queue after r. Since r (which is a 0-packet) is preemptable by many packets of M_1 but has not been preempted by any of them, the credits of the packets of M_1 must have been used to preempt some other packets that were in front of r when they got preempted. Let R be the set of these preempted packets. Recall that the total credit of the packets of M_1 equals $|M_1|$ and that each preemption requires an amount β of credit. Thus, the number of packets in R is given by

$$|R| \geq \left\lfloor \frac{|M_1|}{\beta} \right\rfloor > \frac{|M_1|}{\beta} - 1 \ . \tag{2}$$

Since the packets of R cannot be preempted before the arrivals of the packets of M_1, all of them must be before r in the queue at time t_p. Thus, the queue of CPG contains the packets of both M_0 and R along with p at time t_p. Clearly, M_0, R, and $\{p\}$ are mutually disjoint. Hence, given Inequalities 1 and 2, the size of CPG's queue at t_p is at least

$$|M_0| + |R| + 1 > |M_0| + |M_1|/\beta = \frac{\beta+1}{\beta} M_0 > \frac{\beta+1}{\beta} \frac{\beta}{\beta+1} B = B \ ,$$

which is strictly larger than B, and hence a contradiction. □

So far, our discussion has been focused on one half of the scene; namely, the one between the start of the sequence and the end of the last overflow slot. We shall now move our focus to the second half which extends from time t_f until the end of the sequence.

First, let t_0 be defined as follows: $t_0 = 0$ if no overflow slot occurs in σ, and $t_0 = t_f$ otherwise. Notice that in both cases, no 1-packet is rejected or evicted by CPG after t_0. Moreover, let T denote the first time by which the sequence stops and the queues of OPT and CPG are both empty. Thus, the benefits of OPT and CPG are given by $|\text{SENT}^*(0, T)|$ and $|\text{SENT}(0, T)|$, respectively.

The following lemma is the main ingredient of the proof of the competitive ratio.

Lemma 8.

$$\left(1 + \frac{\beta - 1}{B}\right) \cdot |\text{SENT}(0, T)| \geq (\beta - 1) \cdot (|\text{LOST}(0, T)| - |\text{LOST}^*(0, T)|)$$

Proof. Obviously,

$$\begin{aligned}|\textsc{sent}(0,T)| &= |\textsc{sent}(0,t_0)| + |Q(t_0)| + |\textsc{arr}(t_0,T)| - |\textsc{lost}(t_0,T)| \\ &\geq |Q(t_0)| + |\textsc{arr}(t_0,T)| - |\textsc{lost}(t_0,T)| \ .\end{aligned}$$

Due to the fact that no 1-packet is rejected or evicted by CPG after t_0, all packets in $\textsc{lost}(t_0,T)$ are lost by preemption. We consider the two cases of t_0.

Case $t_0 = 0$: Since preempting a packet requires a credit of β, preempting the packets of $\textsc{lost}(0,T)$ implies the arrival of at least $\beta\,|\textsc{lost}(0,T)|$ 1-packets. Hence,

$$\begin{aligned}\left(1+\frac{\beta-1}{B}\right)\cdot|\textsc{sent}(0,T)| \geq |\textsc{sent}(0,T)| &\geq |Q(0)| + |\textsc{arr}(0,T)| - |\textsc{lost}(0,T)| \\ &\geq \beta\cdot|\textsc{lost}(0,T)| - |\textsc{lost}(0,T)| \\ &= (\beta-1)\cdot|\textsc{lost}(0,T)| \\ &\geq (\beta-1)\cdot(|\textsc{lost}(0,T)| - |\textsc{lost}^*(0,T)|) \ .\end{aligned}$$

Case $t_0 = t_f$: Since preempting a packet requires a credit of β and due to Lemma 5, preempting the packets of $\textsc{lost}(t_f,T)$ implies the arrival of at least $\beta\cdot(|\textsc{lost}(t_f,T)|-1)+1$ 1-packets after t_f.

Due to Observation 4, $|Q(t_f)| = B$ and, as a consequence, $|\textsc{sent}(0,T)| \geq |\textsc{sent}(t_f,T)| \geq B$, since $|\textsc{arr}(t_f,T)| \geq \beta\cdot(|\textsc{lost}(t_f,T)|-1)+1 \geq |\textsc{lost}(t_f,T)|$. In this case, the lemma follows from

$$\begin{aligned}\left(1+\frac{\beta-1}{B}\right)\cdot|\textsc{sent}(0,T)| &\geq |\textsc{sent}(0,T)| + \beta - 1 \\ &\geq |Q(t_f)| + |\textsc{arr}(t_f,T)| - |\textsc{lost}(t_f,T)| + \beta - 1 \\ &\geq B + \beta\cdot(|\textsc{lost}(t_f,T)|-1) + 1 - |\textsc{lost}(t_f,T)| + \beta - 1 \\ &\geq \frac{\beta+1}{\beta}\cdot(|\textsc{lost}(0,t_f)| - |\textsc{lost}^*(0,t_f)| + |Q(t_f)| - |Q^*(t_f)|) \\ &\quad + (\beta-1)\cdot|\textsc{lost}(t_f,T)| \\ &\geq (\beta-1)\cdot(|\textsc{lost}(0,t_f)| - |\textsc{lost}^*(0,t_f)| + |Q(t_f)| - |Q^*(t_f)|) \\ &\quad + (\beta-1)\cdot(|\textsc{lost}(t_f,T)| - |\textsc{lost}^*(t_f,T)| - |Q(t_f)| + |Q^*(t_f)|) \\ &= (\beta-1)\cdot(|\textsc{lost}(0,T)| - |\textsc{lost}^*(0,T)|) \ ,\end{aligned}$$

where the third inequality follows from Observation 4, the fourth from Lemma 7, and the fifth from the fact that $\beta - 1 = (\beta+1)/\beta$, for $\beta = \sqrt{2}+1$. □

Lemma 8 now implies Theorem 3 as follows.

$$\begin{aligned}|\textsc{sent}^*(0,T)| &= |\textsc{arr}(0,T)| - |\textsc{lost}^*(0,T)| \\ &= |\textsc{sent}(0,T)| + |\textsc{lost}(0,T)| - |\textsc{lost}^*(0,T)| \\ &\leq |\textsc{sent}(0,T)| + \left(\frac{1}{\beta-1}+\frac{1}{B}\right)\cdot|\textsc{sent}(0,T)| \\ &= \left(\frac{\beta}{\beta-1}+\frac{1}{B}\right)\cdot|\textsc{sent}(0,T)| \ .\end{aligned}$$

3 Bounded-delay model

3.1 Lower bound

First, we present a lower bound of 2 for any deterministic comparison-based algorithm in the 2-bounded case.

Theorem 9. *The competitive ratio of any deterministic comparison-based algorithm in the 2-bounded case cannot be better than 2.*

Proof. First, fix a comparison-based algorithm ALG. We generate an input sequence for ALG as follows. In the first time slot, two packets arrive, p_1 with deadline 1 and p_2 with deadline 2, such that $v(p_1) \leq v(p_2)$. If ALG sends p_1 in the first slot, we continue the sequence by generating a new packet p_3 in the second time slot with deadline 2; otherwise, we stop the sequence. No more packets arrive after p_3 in the first case.

Now consider the first case. If ALG sends p_1 in the first slot, we reveal the values of packets as $v(p_1) = 0$ and $v(p_2) = v(p_3) = 1$. Thus, the optimal algorithm makes a benefit of 2 by sending p_2 in the first slot and p_3 in the second, while ALG benefits only from sending either p_2 or p_3 in the second slot as both packets have the same deadline.

If ALG sends p_2 in the second case, we let $v(p_1) = v(p_2) = 1$. Thus, the optimal algorithm makes again 2 by sending both packets and ALG makes only 1 as p_1 expires before it is sent. □

The 2-bounded case is a special case of the s-bounded case as well as the cases of agreeable and general deadlines. Thus, the lower bound of 2 holds for all those cases. Even for general deadlines, a simple greedy strategy achieves a competitive ratio of 2 [20]. Therefore, this lower bound is also tight. It also shows that the restriction to comparison-based online algorithms makes the problem much harder since there do exist deterministic online algorithms that are not comparison-based and achieve competitive ratios below 2.

3.2 Two deterministic algorithms

In the following, we consider the more restrictive s-uniform case. First, we introduce some notation. We say that a packet p is *pending* at time t if it is stored in the buffer at time t. Obviously, a pending packet is deleted from the queue once it exceeds its deadline or is sent. At any time t, a *provisional schedule* P_t is a mapping between the set of pending packets at t and the set of send events starting from t. An *optimal provisional schedule* P_t^* is a provisional schedule with the maximum total value among all provisional schedules at time t. It is well known that the set of provisional schedules at time t is a matroid over the set of packets [14]. Therefore, P_t^* can be computed by a greedy algorithm which considers packets in descendeing order of their value.

We now define two simple comparison-based algorithms, GR and EDF, as follows.

- GREEDY (GR). At any send event t, send the packet with the greatest value that is pending at t.
- EARLIEST DEADLINE FIRST (EDF). At any send event t, compute an optimal provisional schedule P_t^*, then send the packet with the earliest deadline in P_t^*.

Ties are broken arbitrarily.

The next theorem gives a lower bound of $2 - 1/s$ on the the competitive ratio of both GR and EDF.

Theorem 10. *The competitive ratio of both* GR *and* EDF *cannot be better than* $2 - 1/s$ *in the* s*-uniform case.*

Proof. As pointed out in the introduction, due to the relationship between the FIFO and s-uniform model, a lower bound on the competitive ratio of EDF can be implied from a lower bound of GREEDY in the FIFO model. Notice that, in the s-uniform case, EDF keeps the most valuable s packets at any time and sends packets in the order of their arrival. This is clearly how GREEDY works on a FIFO queue of capacity $B = s$, and we know from [20] that the competitive ratio of GREEDY cannot be better than $2 - 1/B$.

To show a lower bound of $2 - 1/s$ on the competitive ratio of GR, we generate the following packet sequence. In the first time slot, a burst of s packets arrives, all of the same value v. In the subsequent $s - 1$ time slots, a single packet arrives in each slot, each of a value $v' > v$. After that, no more packets arrive.

Clearly, GR sends a packet of value v in the first slot and then it sends each packet of value v' immediately after its arrival. Thus, GR fails to send the remaining $s - 1$ packets of value v as they expire at time $s + 1$. On the other hand, OPT can send all packets of the sequence in the order of their arrival. Therefore, if it turns out that all packets are 1-packets, GR will only make a benefit of s while OPT will make a benefit of $2s - 1$. □

3.3 One randomized algorithm

In this section, we show that a randomized combination of GR and EDF gives us a better algorithm. Our main result in the bounded-delay model is a randomized comparison-based algorithm that tosses a biased coin at the beginning, then sticks to one of the two deterministic algorithms until the end. This kind of algorithm is called *barely randomized* [24]. More specifically, our algorithm, which we call RAND, chooses to run GR with probability 4/5 and EDF with probability 1/5. The following theorem shows that RAND is 5/3-competitive in the s-uniform case.

Theorem 11. *In the s-uniform case, the competitive ratio of* RAND *is at most 5/3.*

Proof. We first fix an event sequence σ of only 0- and 1-packets. Without loss of generality, we again assume that OPT sends only the 1-packets of σ. Furthermore, since OPT knows in advance the "right" packets to send, we assume that OPT sends its packets in order of arrival.

As discussed in the introduction, the FIFO model is related to the s-uniform bounded-delay model. Both models are equivalent from the point of view of OPT. Furthermore, EDF corresponds to algorithm GREEDY discussed in Section 1. GREEDY, in the FIFO model, results in the same benefit that EDF obtains in the s-uniform bounded-delay model.

GR on the other hand cannot be implemented in the FIFO model. Therefore, in the following, when we argue about GR, we will still take the viewpoint of the s-uniform bounded-delay model, but to analyze EDF, we will argue using FIFO queues.

Next, we show how to partition σ into what we call *active* and *idle* intervals. Each such interval consists of one or more complete time slots. An interval i is called *active* if GR sends a 1-packet in each slot of this interval. In contrast to that, an interval i is called *idle* if it contains no slot in which GR sends a 1-packet. σ is partitioned into such intervals, such that active and idle intervals alternate.

Given the above definition of intervals, we observe that 1-packets arrive in active intervals only. Thus, we focus only on active intervals. Let A_i denote the set of 1-packets that arrive in active interval i. Moreover, let S_i^{OPT} and S_i^{GR} denote the set of 1-packets that OPT and GR send from the set A_i, respectively.

Without loss of generality, we assume that OPT sends all 1-packets sent by GR in i. This is due to the fact that feasible schedules have a matroid structure. So by forcing OPT to send a specific 1-packet, OPT only has to sacrifice one other 1-packet instead. So the overall benefit of OPT stays the same. Therefore, $S_i^{\text{GR}} \subseteq S_i^{\text{OPT}}$. Let $X_i = S_i^{\text{OPT}} \setminus S_i^{\text{GR}}$, i.e., the set of 1-packets that arrive in i and are sent by OPT but they expire before GR sends them. Hence, the following observation.

Observation 12. *For any active interval i, $|S_i^{\text{OPT}}| = |S_i^{\text{GR}}| + |X_i|$.*

We can also get an easy bound on $|X_i|$.

Observation 13. *For any active interval i, $|X_i| < s$.*

Proof. Since all packets from the set A_i arrive within interval i and all packets have a lifespan of s, at most $s-1$ of them can be sent after interval i ends. Therefore, $|S_i^{\text{OPT}}| < |S_i^{\text{GR}}| + s$. Combining this with Observation 12, gives us the claim. □

Furthermore, notice that all packets of X_i arrive within interval i and expire at the end of i at the latest; otherwise, GR would send some of them in the following interval which violates the idleness of that interval. Therefore, either $|X_i| = 0$ or interval i must consist of at least s time steps. Observe that GR sends only 1-packets in active intervals and thus the length of interval i is equal to $|S_i^{\text{GR}}|$. Hence the following observation.

Observation 14. *For any active interval i, either $|X_i| = 0$ or $s \leq |S_i^{\text{GR}}|$.*

The next lemma marks the point at which the second deterministic algorithm EDF enters the scene and it is the last ingredient that we need to conclude the proof of Theorem 11. We first define a new type of intervals which we call EDF-*interval*. An EDF-interval i is identified with an active interval i and its start- and endpoint are determined as follows. It starts with the first time step in which OPT sends a packet of A_i

and ends with the last time step in which OPT sends a packet of A_i. By the definition of active intervals and since OPT sends its packet in their order of arrival, EDF-intervals do not overlap.

Now, let S_i^{EDF} denote the 1-packets that EDF sends in the EDF-interval i. Note that S_i^{EDF} is defined in a different way than S_i^{GR} and S_i^{OPT}, since S_i^{EDF} is based on the time when a packet is sent rather than in which interval a packet arrives. The following lemma provides us with lower bounds on $|S_i^{\text{EDF}}|$.

Lemma 15. *For an active interval i,*

1. EDF *sends at least $|X_i|$ 1-packets from the start of* EDF*-interval i until the end of active interval i, and*
2. EDF *sends at least $2|X_i| - s$ 1-packets from the end of active interval i (excluding the last step of the interval) until the end of* EDF*-interval i (including the last step of the interval).*

Proof. We first observe that the FIFO queue of EDF always contains at least as many 1-packets as the FIFO queue of OPT.

Now, we show the first statement of the lemma. Consider the time step t exactly s time steps before interval i ends. Note that all packets in X_i must have arrived by time t since they expire before i ends.

If the buffer is ever completely full with 1-packets between the start of EDF-interval i and t, we are done because EDF is guaranteed to send s 1-packets in the s time steps between t and the end of active interval i, and, by Observation 13, $|X_i| < s$.

Suppose this is not the case. Then EDF stores (and eventually transmits before the end of active interval i) every 1-packet that arrives between the start of EDF-interval i and t as well as all 1-packets that are stored in the FIFO queue at the beginning of EDF-interval i. Hence, EDF transmits all packets of X_i that arrive after the start of EDF-interval i. Suppose k many of the packets in X_i arrive before the start of EDF-interval i. But since OPT only starts sending packets from $A_i \supseteq X_i$ after EDF-interval i starts, it must have all these k packets stored in its FIFO queue at the beginning of EDF-interval i. Consequently, EDF also has at least k 1-packets stored in its FIFO queue at the beginning of EDF-interval i. Thus, EDF sends at least k 1-packets plus all packets from X_i that arrive after EDF-interval i starts which is at least as much as $|X_i|$.

Next, we show the second statement of the lemma. Observe that active interval i has length $|S_i^{\text{GR}}|$. Therefore, if OPT sends $|S_i^{\text{GR}}| + |X_i|$ 1-packets arriving during interval i, at least $|X_i|$ of these packets must be sent after interval i ends. However, since all these packets arrive before interval i ends, they must be in OPT's queue at the end of interval i. Therefore, EDF also has at least $|X_i|$ 1-packets stored in its queue at that point in time.

Now consider which packets EDF sends in the following $|X_i|$ steps. Note that all these time steps are within the EDF-interval i since OPT has to send at least $|X_i|$ more packets of A_i. Specifically, consider how many 0-packets EDF can send in these $|X_i|$ steps. The queue has size at most s and contains at least $|X_i|$ 1-packets at the end of active interval i. Therefore, there can be at most $s - |X_i|$ 0-packets in total positioned in the queue in front of the last of the 1-packets. Therefore EDF sends at most $s - |X_i|$ 0-packets in the $|X_i|$ steps following the end of interval i and thus, it sends at least $2|X_i| - s$ 1-packets in those steps. (Notice that if any of the $|X_i|$ 1-packets is dropped, the buffer must be full with 1-packets in this case and thus we are guaranteed to send 1-packets from then on anyway.) □

The first part of the lemma implies

$$|S_i^{\text{EDF}}| \geq |X_i| \ . \tag{3}$$

Adding up the bound from the first and second part of the lemma and combining this with Observation 14, additionally gives us

$$|S_i^{\text{EDF}}| \geq 3 \cdot |X_i| - |S_i^{\text{GR}}| \ . \tag{4}$$

Now, we conclude the proof of Theorem 11 as follows. As mentioned earlier, we only consider active intervals because 1-packets only arrive in these intervals. For any active interval i, let OPT_i and RAND_i denote the benefit of OPT and RAND related to interval i, respectively. RAND_i is given as the weighted average of $|S_i^{\text{GR}}|$ and $|S_i^{\text{EDF}}|$. Given Observation 12, the competitive ratio is written as

$$\frac{\text{OPT}_i}{E[\text{RAND}_i]} = \frac{|S_i^{\text{OPT}}|}{\frac{4}{5}\,|S_i^{\text{GR}}| + \frac{1}{5}\,|S_i^{\text{EDF}}|} = \frac{|S_i^{\text{GR}}| + |X_i|}{\frac{4}{5}\,|S_i^{\text{GR}}| + \frac{1}{5}\,|S_i^{\text{EDF}}|} \ .$$

Let $|X_i| = x\,|S_i^{\text{GR}}|$. By Observations 13 and 14, x is between 0 and 1. Now, we distinguish between two cases based on the value of x. If $x < 1/2$, then by Equation 3, $|S_i^{\text{EDF}}| \geq x\,|S_i^{\text{GR}}|$. Thus,

$$\frac{\text{OPT}_i}{E[\text{RAND}_i]} \leq \frac{|S_i^{\text{GR}}| + x\,|S_i^{\text{GR}}|}{\frac{4}{5}\,|S_i^{\text{GR}}| + \frac{1}{5}\,x\,|S_i^{\text{GR}}|} = \frac{1+x}{\frac{4}{5} + \frac{x}{5}} \leq \frac{5}{3} \ .$$

If $x \geq 1/2$, then by Equation 4, $|S_i^{\text{EDF}}| \geq (3x-1)\,|S_i^{\text{GR}}|$. Thus,

$$\frac{\text{OPT}_i}{E[\text{RAND}_i]} \leq \frac{|S_i^{\text{GR}}| + x\,|S_i^{\text{GR}}|}{\frac{4}{5}\,|S_i^{\text{GR}}| + \frac{1}{5}\,(3x-1)\,|S_i^{\text{GR}}|} = \frac{1+x}{\frac{4}{5} + \frac{3x-1}{5}} = \frac{5\,(1+x)}{3\,(1+x)} = \frac{5}{3} \ . \qquad \square$$

4 Conclusions

Our main result in the FIFO model is a lower bound of $1 + 1/\sqrt{2} \approx 1.707$ on the competitive ratio of any deterministic comparison-based algorithm, and an algorithm, CPG, that matches this lower bound, for large B, in the case of monotonic sequences. However, for general sequences, the intriguing question of whether there exists a deterministic comparison-based online algorithm with a competitive ratio below 2 remains open.

For the bounded-delay model, we show that no deterministic comparison-based algorithm exists with a competitive ratio below 2 in the general case as well as in the cases of s-bounded and agreeable deadlines. In the s-uniform case, we present a randomized comparison-based algorithm that is 5/3-competitive.

References

[1] William Aiello, Yishay Mansour, S. Rajagopolan, and Adi Rosén. Competitive queue policies for differentiated services. *Journal of Algorithms*, 55(2):113–141, 2005.

[2] Nir Andelma and Yishay Mansour. Competitive management of non-preemptive queues with multiple values. In *Proc. of the 17th Int. Conf. on Distributed Computing (DISC)*, pages 166–180, 2003.

[3] Nir Andelma, Yishay Mansour, and An Zhu. Competitive queueing policies for QoS switches. In *Proc. of the 14th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 761–770, 2003.

[4] Nir Andelman. Randomized queue management for DiffServ. In *Proc. of the 17th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 1–10, 2005.

[5] Yossi Azar and Ilan Reuven Cohen. Serving in the dark should be done non-uniformly. In *Proc. of the 42nd Int. Colloquium on Automata, Languages and Programming (ICALP)*, pages 91–102, 2015.

[6] Yossi Azar, Ilan Reuven Cohen, and Iftah Gamzu. The loss of serving in the dark. In *Proc. of the 45th ACM Symp. on Theory of Computing (STOC)*, pages 951–960, 2013.

[7] Yossi Azar and Yossi Richter. The zero-one principle for switching networks. In *Proc. of the 36th ACM Symp. on Theory of Computing (STOC)*, pages 64–71, 2004.

[8] Nikhil Bansal, Lisa Fleischer, Tracy Kimbrel, Mohammad Mahdian, Baruch Schieber, and Maxim Sviridenko. Further improvements in competitive guarantees for QoS buffering. In *Proc. of the 31st Int. Colloquium on Automata, Languages and Programming (ICALP)*, pages 196–207, 2004.

[9] Marcin Bienkowski, Marek Chrobak, and Łukasz Jeż. Randomized algorithms for buffer management with 2-bounded delay. In *Proc. of the 6th Workshop on Approximation and Online Algorithms (WAOA)*, pages 92–104, 2008.

[10] Marcin Bienkowski, Marek Chrobak, and Łukasz Jeż. Randomized competitive algorithms for online buffer management in the adaptive adversary model. *Theoretical Computer Science*, 412(39):5121–5131, 2011.

[11] Francis Y. L. Chin, Marek Chrobak, Stanley P. Y. Fung, Wojciech Jawor, Jiří Sgall, and Tomáš Tichý. Online competitive algorithms for maximizing weighted throughput of unit jobs. *Journal of Discrete Algorithms*, 4(2):255–276, 2006.

[12] Francis Y. L. Chin and Stanley P. Y. Fung. Online scheduling for partial job values: Does timesharing or randomization help? *Algorithmica*, 37:149–164, 2003.

[13] Marek Chrobak, Wojciech Jawor, Jiří Sgall, and Tomáš Tichý. Improved online algorithms for buffer management in QoS switches. *ACM Transactions on Algorithms*, 3(4), 2007.

[14] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

[15] Matthias Englert and Matthias Westermann. Lower and upper bounds on FIFO buffer management in QoS switches. *Algorithmica*, 53(4):523–548, 2009.

[16] Matthias Englert and Matthias Westermann. Considering suppressed packets improves buffer management in quality of service switches. *SIAM Journal on Computing*, 41(5):1166–1192, 2012.

[17] Michael H. Goldwasser. A survey of buffer management policies for packet switches. *SIGACT News*, 41:100–128, 2010.

[18] Łukasz Jeż, Fei Li, Jay Sethuraman, and Clifford Stein. Online scheduling of packets with agreeable deadlines. *ACM Transactions on Algorithms*, 9(1):Article 5, 2012.

[19] Alex Kesselman, Yishay Mansour, and Rob van Stee. Improved competitive guarantees for QoS buffering. *Algorithmica*, 43(1-2):97–111, 2005.

[20] Alexander Kesselman, Zvi Lotker, Yishay Mansour, Boal Patt-Shamir, Baruch Schieber, and Maxim Sviridenko. Buffer overflow management in QoS switches. *SIAM Journal on Computing*, 33(3):563–583, 2004.

[21] Fei Li, Jay Sethuraman, and Clifford Stein. An optimal online algorithm for packet scheduling with agreeable deadlines. In *Proc. of the 16th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 801–802, 2005.

[22] Fei Li, Jay Sethuraman, and Clifford Stein. Better online buffer management. In *Proc. of the 18th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 199–208, 2007.

[23] Vern Paxson and Sally Floyd. Wide-area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.

[24] Nick Reingold, Jeffery Westbrook, and Daniel Dominic Sleator. Randomized competitive algorithms for the list update problem. *Algorithmica*, 11(1):15–32, 1994.

[25] Daniel Sleator and Robert Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[26] Andras Veres and Miklós Boda. The chaotic nature of TCP congestion control. In *In Proc. of the 19th Annual IEEE International Conference on Computer Communications (INFOCOM)*, pages 1715–1723, 2000.

[27] An Zhu. Analysis of queueing policies in QoS switches. *Journal of Algorithms*, 53(2):137–168, 2004.