Comparison-based FIFO Buffer Management in QoS Switches[⋆]

Kamal Al-Bawani[1], Matthias Englert[2], and Matthias Westermann[3]

[1] Department of Computer Science, RWTH Aachen University, Germany
kbawani@cs.rwth-aachen.de
[2] DIMAP and Department of Computer Science, University of Warwick, UK
englert@dcs.warwick.ac.uk
[3] Department of Computer Science, TU Dortmund, Germany
matthias.westermann@cs.tu-dortmund.de

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 FIFO buffer and only one packet can be transmitted. Packets may be kept in the buffer of limited size and, due to the FIFO property, the sequence of transmitted packets has to be a subsequence of the arriving packets. The differentiated service concept is implemented by attributing each packet with a non-negative value corresponding to its service level. A buffer management algorithm can reject arriving packets and preempt buffered packets. The goal is to maximize the total value of transmitted packets.
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 algorithms proves to be robust in the realm of QoS switches. Kesselman et al. (SIAM J. Comput., 2004) present a comparison-based algorithm that is 2-competitive. For a long time, it has been an open problem whether a comparison-based algorithm exists with a competitive ratio below 2. 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.

Keywords: Online algorithms, competitive analysis, network switches, buffer management, Quality of Service, comparison-based

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, i.e., a FIFO queue, of bounded

[⋆] The second and third author's work was supported by ERC Grant Agreement No. 307696.

capacity. Each packet has a non-negative value attributing its service level (also known as class of service (CoS)). Packets are stored in the buffer and only one packet can be transmitted in each time step. Due to the FIFO property, the sequence of transmitted packets has to be a subsequence of the arriving packets. A buffer management algorithm can reject arriving packets and preempt packets that were previously inserted into the buffer. The goal is to maximize the total value of transmitted packets.

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., [15, 17]). Therefore, we do not make any prior assumptions about the arrival behavior of packets, and instead resort to the framework of competitive analysis [16], which is the typical worst-case analysis used to assess the performance of online algorithms, i.e., algorithms whose input is revealed piece by piece over time, and the decision they make in each time step is 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.

Kesselman et al. [13] present the following simple GREEDY algorithm: 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. show it is 2-competitive. Since the introduction of GREEDY, it has been an open problem to show whether a comparison-based algorithm exists with a competitive ratio below 2.

1.1 Related Work

In their seminal work, Mansour, Patt-Shamir and Lapid show that GREEDY is 4-competitive. Kesselman et al. [13] show that the exact competitive ratio of GREEDY is $2 - 1/B$, where B is the size of the buffer.

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 ([7]). *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.*

For our model of a single FIFO queue, Andelman [4] employs the 0/1 principle to give a randomized comparison-based algorithm with a competitive ratio of 1.75. In fact, this is the only randomized algorithm known for this model.

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 model of a single FIFO queue has been extensively studied. Kesselman, Mansour, and van Stee [12] 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 [10] show that PG is in fact 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. [13] give a lower bound of 1.282 on the competitive ratio of any deterministic algorithm. Englert and Westermann [10] 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 problem of online buffer management has also been studied under several other models. For example, the bounded delay model, where packets have deadlines besides their values [9, 14]. A recent and comprehensive survey on this problem and most of its variants is given in [11].

1.2 Our Results

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. We also give an algorithm, CPG, that matches our lower bound in the case of *monotonic sequences*, i.e., packets arrive in a non-decreasing order according to their values. Note that GREEDY remains 2-competitive in the case of monotonic sequences. For general sequences, we give a lower bound of 1.829 on the competitive ratio of CPG.

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. 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 algorithms must come at a price, namely, a significantly degraded performance.

1.3 Model and Notations

We consider a single buffer that can store up to B packets. All packets are assumed to be of unit size, and each packet p is associated with a non-negative value, denoted by $v(p)$, that corresponds to its level of service. The buffer is implemented as a FIFO queue, i.e., packets are stored and sent in the order of their arrival.

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 ALG(σ), and is defined as the total value of packets that ALG sends. We aim at maximizing this benefit. We denote by OPT an optimal (offline) algorithm that sends packets in FIFO order.

2 Lower Bound

The following theorem shows that no deterministic comparison-based algorithm can be better than $1 + 1/\sqrt{2} \approx 1.707$. 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 that ONL takes to send and preempt all packets that it has in 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 $t_i \geq B/\sqrt{2}$,

$$\frac{\text{OPT}}{\text{ONL}} = \frac{t_i + B}{B} \geq \frac{B/\sqrt{2} + B}{B} \quad .$$

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 $t_i \geq t_{i+1}$,

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

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 $p$ and the
value of $r$ is less than the value of the packet that is behind $r$ (if any).
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 $p$;
```

(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 $t_i \geq B/\sqrt{2}$. □

3 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 [8]. 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.

A packet r is preempted upon the arrival of another packet p if r is the first packet in the queue (in the FIFO order) such that r is preemptable by p and the value of r is less than the value of the packet that is behind r in the queue (if any). After a packet r is preempted, CPG invokes a subroutine CHARGE to deduct a total of β units from the credits of the preempting 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.
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 and the value of r is less than the value of the packet that is behind r (if any).

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

3.1 Monotonic Sequences

In this section, we consider input sequences in which packets arrive with non-decreasing values, i.e., for any two packets p and q, $v(p) \leq v(q)$ if and only if p arrives before q. We observe that the 2-competitive greedy algorithm from [13] remains 2-competitive in this case.

Theorem 3. *Choosing $\beta = \sqrt{2} + 1$, the competitive ratio of* CPG *is at most $1 + 1/\sqrt{2} \approx 1.707$.*

For the rest of the analysis, we fix an event sequence σ of only 0- and 1-packets. Furthermore, 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 Algorithms. 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 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. 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.

Remark 1. $|Q(t_f)| = B$.

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 1. *Consider any arrival event e. Let t be the time immediately after 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. We show the lemma by contradiction. Let e be the first arrival event in σ, such that immediately after e, say at time t, $c(D(t)) \geq \beta$. Hence, immediately before e, say at time t', $\beta > c(D(t')) \geq \beta - 1$, since the total credit of the queue cannot increase by more than 1 in each arrival event.

Now, let p be the packet arriving in e and let q be the head packet at the arrival of p. Recall that σ is monotonic. Thus, the packets behind q in the queue and packet p are all at least as valuable as q. Hence, adding the credit of p to $c(D(t'))$, these packets would preempt q upon the arrival of p, and thus the total credit would decrease by 1. Therefore, $c(D(\cdot))$ does not change between t' and t which contradicts the definition of e. $\square$

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'. 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'. Similarly, we define $\textsc{sent}^*(t, t')$ and $\textsc{lost}^*(t, t')$ for OPT.

Lemma 2. *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:

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

$\square$

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

Proof. First, we present further notations. If an algorithm ALG does not send anything in a sent event t, we say that ALG sends a $\emptyset$-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'.

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

$$|\textsc{sent}^*(0, t_f)| - |\textsc{sent}(0, t_f)| = \delta_{1/0}(0, t_f) - \delta_{\emptyset/1}(0, t_f) \le \delta_{1/0}(0, t_f) \ .$$

Hence, given Lemma 2, it suffices to show that $\delta_{1/0}(0, t_f) \le \lfloor \frac{\beta}{\beta+1} B \rfloor$.

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,

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

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 (which is a 0-packet) without preempting it. Since the credits of packets are used only in preemption, the credits of these 1-packets must be used to preempt other packets before r. Let R be the set of these preempted packets. Obviously,

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

Since the packets of R cannot be preempted before the arrivals of the packets of M_1, all of them must be then 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, $M_0 \cap R \cap \{p\} = \emptyset$. 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 4. $|\text{SENT}(0,T)| \geq (\beta - 1)\left(|\text{LOST}(0,T)| - |\text{LOST}^*(0,T)|\right)$.

Proof. Obviously, we can write $|\text{SENT}(0,T)|$ as follows:

$$\begin{aligned} |\text{SENT}(0,T)| &= |\text{SENT}(0,t_0)| + |Q(t_0)| + |\text{ARR}(t_0,T)| - |\text{LOST}(t_0,T)| \\ &\geq |Q(t_0)| + |\text{ARR}(t_0,T)| - |\text{LOST}(t_0,T)| \ . \end{aligned}$$

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 further notice that all these packets are preempted using packets that arrive after t_0. This is trivial in case $t_0 = 0$, and follows from Lemma 1 in case $t_0 = t_f$. (In fact, in the latter case, at most one packet of $\textsc{lost}(t_0, T)$ can be preempted using the credits of packets that are in the queue at time t_f, but this anomaly can be covered by introducing an additive constant in the competitive ratio of CPG.) Since preempting a packet requires a credit of β, preempting the packets of $\textsc{lost}(t_0, T)$ implies the arrival of at least new $\beta\,|\textsc{lost}(t_0, T)|$ 1-packets that are inserted into the queue after t_0. Thus, $|\textsc{arr}(t_0, T)| \geq \beta\,|\textsc{lost}(t_0, T)|$, and hence we can rewrite $|\textsc{sent}(0, T)|$ in the following way:

$$\begin{aligned}|\textsc{sent}(0,T)| &\geq |Q(t_0)| + \beta\,|\textsc{lost}(t_0,T)| - |\textsc{lost}(t_0,T)| \\ &= |Q(t_0)| + (\beta - 1)\,|\textsc{lost}(t_0,T)| \\ &\geq |Q(t_0)| + (\beta - 1)\,(|\textsc{lost}(t_0,T)| - |\textsc{lost}^*(t_0,T)|) \ .\end{aligned}$$

Now, if $t_0 = 0$, then $|Q(t_0)| = 0$ and thus the lemma follows immediately. If $t_0 = t_f$, we continue as follows:

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

where the first inequality follows from Remark 1, the second inequality from Lemma 3, and the equality from the fact that $\beta - 1 = (\beta+1)/\beta$, for $\beta = \sqrt{2}+1$. □

Now, we use Lemma 4 to show that $|\textsc{sent}^*(0,T)| \leq \frac{\beta}{\beta-1}\,|\textsc{sent}(0,T)|$, which obviously completes the proof of Theorem 3:

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

3.2 General Sequences

Theorem 3 shows that CPG is an optimal comparison-based algorithm in the case of monotonic sequences. In this section, we investigate how this algorithm performs on general sequences.

We notice that Lemma 1 does not necessarily hold for general sequences. Therefore, after an overflow of 1-packets takes place, the total credit of the 1-packets in the online buffer can significantly exceed β and thus some of these packets may be used in a subsequent time steps to preempt other packets from the same group, i.e., the group of the B 1-packets from the overflow slot. Consequently, the lower bound of B on the number of CPG's sent 1-packets may no longer hold in the general case, resulting in a competitive ratio worse than 1.707. Such a bad scenario for CPG is illustrated in the proof of the following theorem and leads to a lower bound of 1.829 on its competitive ratio.

Theorem 4. *For any value of β,* CPG *cannot be better than 1.829-competitive.*

Proof. The adversary generates one of the following two sequences based on the value of β:

Case 1. $\beta \leq 2.206$: In the first time slot, B 1-packets are generated in an increasing order (with respect to their original values). After that, no more packets arrive. Clearly, OPT sends all the B packets, while in CPG, every β packets preempt a packet from the front. Thus, CPG preempts B/β in total. Hence, its competitive ratio is given by

$$\frac{\text{OPT}}{\text{CPG}} = \frac{B}{B - B/\beta} = \frac{\beta}{\beta - 1} \geq 1.829 \ .$$

Case 2. $\beta > 2.206$: In the first time slot, $(B-1)$ 0-packets are generated followed by a single 1-packet. Then, over the next $\beta B/(\beta+1) - 1$ time slots, a single 1-packet is generated in each slot. Let M_1 denote the set of those 1-packets that arrive in the first $\beta B/(\beta+1)$ time slots. After that, in slot number $\beta B/(\beta+1)+1$, B 1-packets arrive at once. Let M_2 denote the set of these packets. Finally, in the next $B/(\beta(\beta+1))$ time slots, a single 1-packet arrives in each slot. Let M_3 denote the set of these packets. After that, no more packets arrive.

Clearly, OPT sends all the 1-packets in the sequence. To minimize the number of 1-packets sent by CPG, the adversary can choose the original values of the 1-packets in the following malicious way. First, the values of packets in M_2 are all strictly less than the smallest value in M_1. Let M_2' denote the set of the first $B/(\beta+1)$ packets in M_2. The packets of M_2' are ordered as follows. For each group of β packets, starting from the earliest, the first packet is strictly smaller than the $\beta - 1$ packets behind it, and all the β packets of this group are strictly smaller than all packets before them in M_2'. For example, for $\beta = 3$, theses groups may look like $|50, 51, 51|40, 41, 41|30, 31, 31| \cdots$. For the rest of M_2, i.e., the set $M_2 \setminus M_2'$, packets are given values that are strictly less than the smallest value in M_2'. Finally, the packets in M_3 are all assigned a value that is equal to the greatest value in M_2'.

Obviously, CPG accepts all the $\beta B/(\beta+1)$ packets of M_1 and uses them to preempt $B/(\beta+1)$ 0-packets. Meanwhile, the rest of the B 0-packets are sent in the first $\beta B/(\beta+1)$ time slots. Thus, the packets of M_1 will be all in the queue of CPG when the packets of M_2 arrive. Clearly, this leads to an overflow of 1-packets and only the packets of M'_2 can be accepted in this time slot. These packets are inserted with full credits into the queue, and thus when each packet from M_3 arrives, it groups with $\beta - 1$ packets from M'_2 to preempt the first packet in one β-group of M'_2, according to the above description of M'_2. Therefore, CPG sends a total of B 1-packets only, and hence its competitive ratio is given by

$$\begin{aligned} \frac{\textsc{opt}}{\textsc{cpg}} &= \frac{|M_1| + |M_2| + |M_3|}{B} \\ &= \frac{\beta}{\beta+1} + 1 + \frac{1}{\beta(\beta+1)} \\ &= \frac{\beta(\beta+1) + \beta^2 + 1}{\beta(\beta+1)} \geq 1.829 \ . \end{aligned}$$

□

4 Conclusions

Our main result 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 in the case of monotonic sequences. For general sequences, CPG is shown to be no better than 1.829-competitive. 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.

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 *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 *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 *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 *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 *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 *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 *In Proc. of the 31st Int. Colloquium on Automata, Languages and Programming (ICALP)*, pages 196–207, 2004.
9. Matthias Englert and Matthias Westermann. Considering suppressed packets improves buffer management in QoS switches. In *In Proc. of the 18th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 209–218, 2007.
10. Matthias Englert and Matthias Westermann. Lower and upper bounds on FIFO buffer management in QoS switches. *Algorithmica*, 53(4):523–548, 2009.
11. Michael H. Goldwasser. A survey of buffer management policies for packet switches. *SIGACT News*, 41:100–128, 2010.
12. Alex Kesselman, Yishay Mansour, and Rob van Stee. Improved competitive guarantees for QoS buffering. *Algorithmica*, 43(1-2):97–111, 2005.
13. 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.
14. Fei Li, Jay Sethuraman, and Clifford Stein. Better online buffer management. In *In Proc. of the 18th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 199–208, 2007.
15. Vern Paxson and Sally Floyd. Wide-area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.
16. Daniel Sleator and Robert Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.
17. Andras Veres and Miklós Boda. The chaotic nature of TCP congestion control. In *In Proc. of IEEE INFOCOM*, pages 1715–1723, 2000.