



Centrum voor Wiskunde en Informatica

REPORTRAPPORT

SEN

Software Engineering



Software ENgineering

Improved Competitive Guarantees for QoS Buffering

Alex Kesselman, Yishay Mansour, Rob van Stee

REPORT SEN-E0317 DECEMBER 19, 2003

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).

CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

Copyright © 2003, Stichting Centrum voor Wiskunde en Informatica

P.O. Box 94079, 1090 GB Amsterdam (NL)

Kruislaan 413, 1098 SJ Amsterdam (NL)

Telephone +31 20 592 9333

Telefax +31 20 592 4199

ISSN 1386-369X

Improved Competitive Guarantees for QoS Buffering

ABSTRACT

We consider a network providing Differentiated Services (Diffserv), which allow Internet Service Providers (ISP's) to offer different levels of Quality of Service (QoS) to different traffic streams. We study two types of buffering policies that are used in network switches supporting QoS. In the FIFO type, packets must be transmitted in the order they arrive. In the uniform *bounded-delay* type, there is a maximum delay time associated with the switch and each packet must be transmitted within this time, or otherwise it is dropped. In both models, the buffer space is limited, and packets are lost when the buffer overflows. Each packet has an intrinsic value, and the goal is to maximize the total value of transmitted packets. Our main contribution is an algorithm for the FIFO model with arbitrary packet values that for the first time achieves a competitive ratio better than 2, namely $2 - \epsilon$ for a constant $\epsilon > 0$. We also describe an algorithm for the uniform bounded delay model which simulates our algorithm for the FIFO model, and show that it achieves the same competitive ratio.

2000 Mathematics Subject Classification: 68W25;68W40

1998 ACM Computing Classification System: F.2.2

Keywords and Phrases: buffering policies;QoS;online algorithms

Note: Work supported by the Deutsche Forschungsgemeinschaft, Project AL 464/3-1, and by the European Community, Projects APPOL and APPOL II. Work partially supported by the Netherlands Organization for Scientific Research (NWO), project number SION 612-061-000.

Improved Competitive Guarantees for QoS Buffering *

Alex Kesselman[†]
alex@cs.tau.ac.il

Yishay Mansour[†]
mansour@cs.tau.ac.il

Rob van Stee[‡]
Rob.van.Stee@cwj.nl

November 17, 2003

Abstract

We consider a network providing Differentiated Services (Diffserv), which allow Internet Service Providers (ISP's) to offer different levels of Quality of Service (QoS) to different traffic streams. We study two types of buffering policies that are used in network switches supporting QoS. In the *FIFO* type, packets must be transmitted in the order they arrive. In the uniform *bounded-delay* type, there is a maximum delay time associated with the switch and each packet must be transmitted within this time, or otherwise it is dropped. In both models, the buffer space is limited, and packets are lost when the buffer overflows. Each packet has an intrinsic value, and the goal is to maximize the total value of transmitted packets. Our main contribution is an algorithm for the FIFO model with arbitrary packet values that for the first time achieves a competitive ratio better than 2, namely $2 - \epsilon$ for a constant $\epsilon > 0$. We also describe an algorithm for the uniform bounded delay model which simulates our algorithm for the FIFO model, and show that it achieves the same competitive ratio.

1 Introduction

Today's prevalent Internet service model is the best-effort model (also known as the "send and pray" model). This model does not permit users to obtain better service, no matter how critical their requirements are, and no matter how much they may be willing to pay for better service. With the increased use of the Internet for commercial purposes, such a model is not satisfactory any more. However, providing any form of stream differentiation is infeasible in the core of the Internet.

Differentiated Services were proposed as a compromise solution for the Internet Quality of Service (QoS) problem. In this approach each packet is assigned a predetermined QoS, thus aggregating traffic to a small number of classes [3]. Each class is forwarded using the same per-hop behavior at the routers, thereby simplifying the processing and storage requirements. Over the past few years Differentiated Services has attracted a great deal of research interest in the networking community [18, 6, 16, 13, 12, 5]. We abstract the DiffServ model as

* A preliminary version of this paper appeared in *Proceedings of the 11th Annual European Symposium on Algorithms (ESA 2003)*, pages 361–372.

[†]School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel.

[‡]Centre for Mathematics and Computer Science (CWI), Kruislaan 413, NL-1098 SJ Amsterdam, The Netherlands. Work supported by the Deutsche Forschungsgemeinschaft, Project AL 464/3-1, and by the European Community, Projects APPOL and APPOL II. Work partially supported by the Netherlands Organization for Scientific Research (NWO), project number SION 612-061-000.

follows: packets of different QoS priority have distinct values and the system obtains the value of a packet that reaches its destination.

To improve the network utilization, most Internet Service Providers (ISP) allow some under-provisioning of the network bandwidth employing the policy known as *statistical multiplexing*. While statistical multiplexing tends to be very cost-effective, it requires satisfactory solutions to the unavoidable events of overload. In this paper we study such scenarios in the context of *buffering*. More specifically, we consider an output port of a network switch with the following activities. At each time step, an arbitrary set of packets arrives, but only one packet can be transmitted. A buffer management algorithm has to serve each packet online, i. e., without knowledge of future arrivals. It performs two functions: selectively rejects and preempts packets, subject to the buffer capacity constraint, and decides which packet to send. The goal is to maximize the total values of packets transmitted.

We consider two types of buffer models. In the classical *First-In-First-Out* (FIFO) model packets can not be sent out of order. Formally, for any two packets p, p' sent at times t, t' , respectively, we have that if $t' > t$, then packet p has not arrived after packet p' . If packets arrive at the same time, we refer the order in which they are processed by the buffer management algorithm, which receives them one by one. Most of today's Internet routers deploy the FIFO buffering policy. The second model we consider is the uniform *bounded delay* model. This model is warranted by networks that guarantee the QoS parameter of end-to-end delay. Specifically, each switch guarantees a prescribed allowed delay time. A packet must be transmitted within this time, else it is lost. Note that in this model packets can be reordered. In both models the buffer size is fixed, so when too many packets arrive, buffer overflow occurs and some packets must be discarded.

Giving a realistic model for Internet traffic is a major problem in itself. Network arrivals have often been modeled as a Poisson process both for ease of simulation and analytic simplicity. Initial works on DiffServ have focused on such simple probabilistic traffic models [11, 15]. However, recent examinations of Internet traffic [14, 19] have challenged the validity of the Poisson model. Moreover, measurements of real traffic suggest the existence of significant traffic variance (burstiness) over a wide range of time scales.

We analyze the performance of a buffer management algorithm by means of competitive analysis. Competitive analysis, introduced by Sleator and Tarjan [17] (see also [4]), compares an on-line algorithm to an optimal offline algorithm OPT , which knows the entire sequence of packet arrivals in advance. Denote the value earned by an algorithm ALG on an input sequence σ by $V_{\text{ALG}}(\sigma)$.

Definition 1.1 *An online algorithm A is c -competitive iff for every sequence of packets σ , $V_{\text{OPT}}(\sigma) \leq c \cdot V_A(\sigma)$.*

An advantage of competitive analysis is that a uniform performance guarantee is provided over all input instances, making it a natural choice for Internet traffic.

In [1] different non-preemptive policies are studied for the two distinct values model. Recently, this work has been generalized to multiple packet values [2], where they also present a lower bound of $\sqrt{2}$ on the performance of any online algorithm in the preemptive model. Analysis of preemptive queuing policies for arbitrary packet values in the context of smoothing video streams appears in [10]. This paper establishes an impossibility result, showing that no online algorithm can have a competitive ratio better than $5/4$, and demonstrates that the greedy algorithm is at least 4-competitive. In [7] the greedy algorithm has been shown to achieve the competitive ratio of 2 in both FIFO and the bounded delay models. An analysis of the loss of an algorithm appears in [8], where they present an algorithm with competitive ratio better than 2 for the case of two and

exponential packet values. In [9] they study the case of two packet values and present a 1.3-competitive algorithm. Our model is identical to that of [7]. The problem of whether the competitive ratio of 2 of the natural greedy algorithm can be improved has been open for a long time and in this paper we solve it positively.

Our Results. The main contribution of this paper is an algorithm for the FIFO model for *arbitrary* packet values that achieves a competitive ratio of $2 - \epsilon$ for a constant $\epsilon > 0$. In particular, this algorithm accomplishes a competitive ratio of 1.983 for a particular setting of parameters. This is the first upper bound below the bound of 2 that was shown in [7]. We also show a lower bound of 1.419 on the performance of any online algorithm, improving on [2], and a specific lower bound of $\phi \approx 1.618$ on the performance of our algorithm. Then we describe an algorithm for the uniform bounded delay model that simulates our algorithm for the FIFO model, and demonstrate that it achieves the same competitive ratio. In contrast to previous work (cf. [7]), we assume that in the uniform bounded delay model the buffer size is fixed.

The rest of the paper is organized as follows. In Section 2 we define our model. The FIFO and the uniform bounded delay models are studied in Section 3 and Section 4, respectively. Section 5 contains the concluding remarks.

2 Model Description

We consider a QoS buffering system that is able to hold B packets. Time is slotted. At the beginning of a time slot a set of packets (possibly empty) arrives and at the end of the time slot a packet is scheduled if any. The buffer management algorithm has to decide at each step which of the packets to drop and which to transmit, subject to the buffer capacity constraint. The value of packet p is denoted by $v(p)$. The system obtains the value of the packets it sends, and the aim of the buffer management algorithm is to maximize the total value of transmitted packets.

We denote by $A(t)$ the set of packets arriving at time slot t , by $Q(t)$ the set of packets in the buffer after the arrival phase at time slot t , and by $ALG(t)$ the packet sent (or scheduled) at the end of time slot t if any by an algorithm ALG . At any time slot t , $|Q(t)| \leq B$ and $|ALG(t)| \leq 1$, whereas $|A(t)|$ can be arbitrarily large. We also denote by $Q(t, \geq w)$ the subset of $Q(t)$ of packets with value at least w .

As mentioned in the Introduction, we consider both FIFO buffers and uniform bounded delay buffers in this paper. In the FIFO model, the packet transmitted at time t is always the first (oldest) packet in the buffer among the packets in $Q(t)$. In the D -uniform bounded delay model, there is a single fixed bound of D on the delay of all packets.

3 FIFO Buffers

In this section we study the FIFO model. First we present the algorithm PG and analyze its performance. Then we demonstrate some lower bounds.

3.1 Algorithm PG

The main idea of the algorithm PG is to make proactive preemptions of low value packets when high value packets arrive. The algorithm is similar to the one presented in [8], except that each high value packet can preempt at most one low value packet. Intuitively, we try to decrease the delay that a high value packet suffers due to low value packets preceding it in the FIFO order. A formal definition is given in Figure 1.

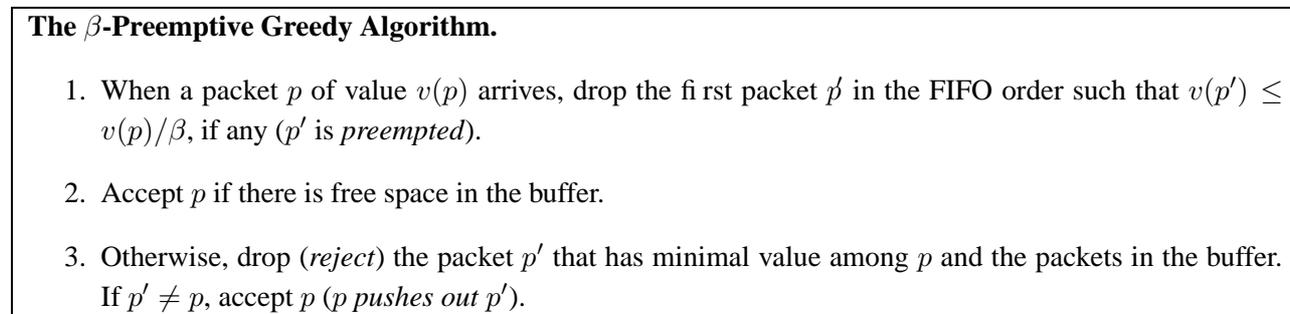


Figure 1: Algorithm PG.

The parameter of PG is the preemption factor β . For sufficiently large value of β , PG performs like the greedy algorithm and only drops packets in case of overflow. On the other hand, too small values of β can cause excessive preemptions of packets and a large loss of value. Thus, we need to optimize the value of β in order to achieve a balance between maximizing the current throughput and minimizing the potential future loss.

The following lemma is the key to showing a competitive ratio below 2. It demonstrates that if the buffer contains a large number of “valuable” packets, then PG sends packets with non-negligible value. This property does not hold for the greedy algorithm [7].

Lemma 3.1 *If at time t , $|\mathcal{Q}(t, \geq w)| \geq B/2$ and the earliest packet from $\mathcal{Q}(t, \geq w)$ arrived before or at time $t - B/2$ then the packet scheduled at the next time step has value at least w/β .*

Proof: Let p be the first packet from $\mathcal{Q}(t, \geq w)$ in the FIFO order and let $t' \leq t - B/2$ be the arrival time of p . Let X be the set of packets with value less than w/β that were in the buffer before p at time t' . We show that no packet from X is present in the buffer at time $t + 1$. We have that $|X| < B$. At least $B/2$ packets are served between t' and t . All these packets preceded p since p is still in the buffer at time t . So at most $B/2$ packets in X are not (yet) served at time t . However, at least $B/2$ packets with value greater than or equal to w have arrived by time t and each of them preempts from the buffer the first packet in the FIFO order with value of at most w/β , if any. This shows that all packets in X have been either served or dropped by time t . ■

In general, we want to assign the value of packets that OPT serves and PG drops to packets served by PG. Note that the schedule of PG contains a sequence of packet rejections and preemptions. We will add structure to this sequence and give a general assignment method based on *overload intervals*.

3.2 Overload Intervals

Before introducing a formal definition, we will give some intuition. Consider a time t at which a packet of value α is rejected and α is the largest value among the packets that are rejected at this time. Note that all packets

in the buffer at the end of time slot t have value at least α . Such an event defines an α -overloaded interval $\mathcal{I} = [t_s, t_f)$, which starts at time $t_s = t$.

In principle, \mathcal{I} ends at the last time at which a packet in $\mathcal{Q}(t)$ is scheduled (i.e. at time $t + B - 1$ or earlier). However, in case at some time $t' > t$ a packet of value γ is rejected, γ is the largest value among the packets that are rejected at this time, and a packet from $\mathcal{Q}(t)$ is still present in the buffer, we proceed as follows.

If $\gamma = \alpha$, we extend \mathcal{I} to include t' . In case $\gamma > \alpha$, we start a new interval with a higher overload value. Otherwise, if $\gamma < \alpha$, a new interval begins when the first packet from $\mathcal{Q}(t) \setminus \mathcal{Q}(t)$ is eventually scheduled if any. Otherwise, if all packets from $\mathcal{Q}(t) \setminus \mathcal{Q}(t)$ are preempted, we create a zero length interval $\mathcal{I}' = [t_f, t_f)$ whose overload value is γ . Next we define the notion of overload intervals more formally.

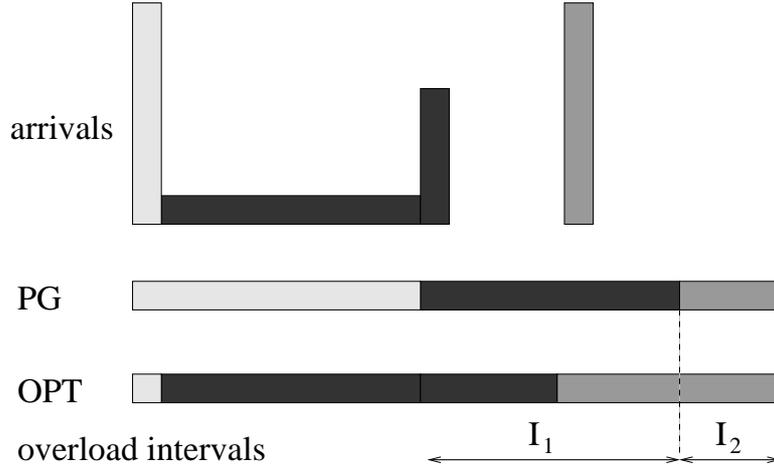


Figure 2: An example of overload intervals. Light packets have value 1, dark packets value $\beta - \epsilon$, medium packets value 2. The arrival graph should be interpreted as follows: B packets of value 1 arrive at time 1, 1 packet of value $\beta - \epsilon$ arrives at times $2, \dots, B - 1$, etc. Note that I_2 does not start until I_1 is finished.

Definition 3.1 An α -overflow takes place when a packet of value α is rejected, where α is said to be the overload value.

Definition 3.2 A packet p is said to be associated with interval $[t, t')$ if p arrived later than the packet scheduled at time $t - 1$ if any and earlier than the packet scheduled at time t' if any.

Intuitively, p is associated with the interval in which it is scheduled, or in which it would have been scheduled if it had not been dropped w.r.t. the FIFO order.

Definition 3.3 An interval $\mathcal{I} = [t_s, t_f)$, with $t_f \geq t_s$, is an α -overloaded interval if the maximum value of a rejected packet associated with it is α , all packets served during \mathcal{I} were present in the buffer in time of an α -overflow, and \mathcal{I} is a maximal such interval that does not overlap overload intervals with higher overload values.

Thus, we construct overload intervals starting from the highest overload value and ending with the lowest overload value. We note that only packets with value at least α are served during an α -overloaded interval.

Definition 3.4 A packet p belongs to an α -overloaded interval $\mathcal{I} = [t_s, t_f)$ if p is associated with \mathcal{I} and (i) p is served during \mathcal{I} , or (ii) p is rejected no earlier than the first and no later than the last α -overflow, or (iii) p

is preempted and it arrived no earlier than the first and no later than the last packet that belongs to \mathcal{I} that is served or rejected.

Whenever an α -overloaded interval \mathcal{I} is immediately followed by a γ -overloaded interval \mathcal{I}' with $\gamma > \alpha$, we have that in the first time step of \mathcal{I}' a packet of value γ is rejected. This does not hold if $\gamma < \alpha$. We give an example in Figure 2.

The following observation states that overload intervals are well-defined.

Observation 1 *Each rejected packet belongs to exactly one overload interval and overload intervals are disjoint.*

Next we introduce some useful definitions related to an overload interval. A packet p *transitively preempts* a packet p' if p either preempts p' or p preempts or pushes out another packet p'' , which transitively preempts p' . A packet p *replaces* a packet p' if (1) p transitively preempts p' and (2) p is eventually scheduled. A packet p *directly replaces* p' if in the set of packets transitively preempted by p no other packet except p' is preempted (e.g. p may push out p'' that preempts p').

Definition 3.5 *For an overload interval \mathcal{I} let $\text{BELONG}(\mathcal{I})$ denote the set of packets that belong to \mathcal{I} . This set consists of three distinct subsets:*

- *scheduled packets* ($\text{PG}(\mathcal{I})$),
- *preempted packets* ($\text{PREEMPT}(\mathcal{I})$) and
- *rejected packets* ($\text{REJECT}(\mathcal{I})$).

Finally, denote by $\text{REPLACE}(\mathcal{I})$ the set of packets that replace packets from $\text{PREEMPT}(\mathcal{I})$. These packets are either in $\text{PG}(\mathcal{I})$ or are served later.

We divide the schedule of PG into maximal sequences of consecutive overload intervals of increasing and then decreasing overload value.

Definition 3.6 *An overload sequence S is a maximal sequence containing intervals $\mathcal{I}_1 = [t_s^1, t_f^1)$, $\mathcal{I}_2 = [t_s^2, t_f^2)$, \dots , $\mathcal{I}_k = [t_s^k, t_f^k)$ with overload values w_1, \dots, w_k such that $t_f^i = t_s^{i+1}$ for $1 \leq i \leq k-1$, $w_i < w_{i+1}$ for $1 \leq i \leq m-1$ and $w_i > w_{i+1}$ for $m \leq i \leq k-1$, where k is the number of intervals in S and w_m is the maximal overload value among the intervals within S .*

Ties are broken by associating an overload interval with the latest overload sequence. We will abbreviate $\text{BELONG}(\mathcal{I}_i)$, $\text{PG}(\mathcal{I}_i)$, \dots by BELONG_i , PG_i , \dots . We make the following observation, which follows from the definition of an overload interval.

Observation 2 *For $1 \leq i \leq k$, all packets in REJECT_i have value at most w_i while all packets in PG_i have value at least w_i .*

3.3 Analysis of the PG Algorithm

In this section we will analyze the performance of the PG algorithm. We show that PG achieves a competitive ratio of $2 - \epsilon$, where $\epsilon(\beta) > 0$ is a constant depending only on β . Optimizing the value of β , we get that for

$\beta = 15$ the competitive ratio of PG is close to 1.983, that is $\epsilon \approx 0.017$. The crux of the proof is to show that when PG drops a packet of value say α that is scheduled by OPT, it schedules another packet of value α and an additional packet with a non-negligible value, roughly α/β , which allows us to break the ratio of 2, achieved by the greedy algorithm.

In the sequel we fix an input sequence σ . Let us denote by OPT and PG the set of packets scheduled by OPT and PG, respectively. We also denote by DROP the set of packets scheduled by OPT and dropped by PG, that is $\text{OPT} \setminus \text{PG}$. In a nutshell, we will construct a *fractional* assignment in which we will assign to packets in PG the value $V_{\text{OPT}}(\sigma)$ so that each packet is assigned at most a $2 - \epsilon$ fraction of its value. The general assignment scheme is presented on Figure 3.

Main Assignment Routine(σ):

1. Assign the value of each packet from $\text{PG} \cap \text{OPT}$ to itself.
2. Assign the value of each *preempted* packet from DROP to the packet *replacing* it.
3. Consider all overload sequences starting from the earliest one and up to the latest one. Assign the value of each *rejected* packet from DROP that belongs to the sequence under consideration using the assignment routine for the overload sequence.

Figure 3: The main routine.

Before we describe the assignment routine for the overload sequence we need some definitions. Consider an overload sequence S . We introduce the following notation:

$$\begin{aligned} \text{OPT}_i &= \text{OPT} \cap \text{BELONG}_i, \\ \text{REJOPT}_i &= \text{OPT} \cap \text{REJECT}_i, \\ \text{PRMOPT}_i &= \text{OPT} \cap \text{PREEMPT}_i \end{aligned}$$

We write $\text{PG}(S) = \cup_{i=1}^k \text{PG}_i$ and define $\text{OPT}(S)$, $\text{REJOPT}(S)$, and $\text{PRMOPT}(S)$ analogously.

Definition 3.7 For $1 \leq i \leq k$, let OUT_i be the set of packets that have been replaced by packets outside S .

Clearly, $\text{OUT}_i \subseteq \text{PREEMPT}_i$. Two intervals \mathcal{I}_i and \mathcal{I}_j are called *adjacent* if either $t_f^i = t_s^j$ or $t_s^i = t_f^j$. The next observation will become important later.

Observation 3 For an interval \mathcal{I}_i , if $|\text{PG}_i| + |\text{OUT}_i| < B$ then \mathcal{I}_i is adjacent to another interval \mathcal{I}_j such that $w_j > w_i$.

Suppose that the arrival time of the earliest packet in $\text{BELONG}(S)$ is t_a and let $\text{EARLY}(S) = \cup_{t=t_a}^{t_s^1-1} \text{PG}(t)$ be the set of packets sent between t_a and time t_s^1 . Intuitively, packets from $\text{EARLY}(S)$ are packets outside S that interact with packets from S and may be later assigned some value of packets from $\text{DROP}(S)$.

Let $\text{PREVP}(S)$ be the subset of $\mathcal{Q}(t_a) \setminus \text{BELONG}(S)$ containing packets preempted or pushed out by packets from $\text{BELONG}(S)$. The next lemma bounds the difference between the number of packets in $\text{OPT}(S)$ and $\text{PG}(S)$.

Lemma 3.2 For an overload sequence S the following holds: $|\text{OPT}(S)| - |\text{PG}(S)| \leq B + |\text{OUT}(S)| - |\text{PREVP}(S)|$.

Proof: Let t' be the last time during S at which a packet from $\text{BELONG}(S)$ has been rejected. It must be the case that $t_f^k - t' \geq B - |\text{OUT}(S)|$ since at time t' the buffer was full of packets from $\text{BELONG}(S)$ and any packet outside $\text{BELONG}(S)$ can preempt at most one packet from $\text{BELONG}(S)$. We argue that OPT has scheduled at most $t' + 2B - t_s^1 - |\text{PREVP}(S)|$ packets from $\text{BELONG}(S)$. That is due to the fact that the earliest packet from $\text{BELONG}(S)$ arrived at or after time $t_s^1 - B + |\text{PREVP}(S)|$. On the other hand, PG has scheduled at least $t' + B - t_s^1 - |\text{OUT}(S)|$ packets from $\text{BELONG}(S)$, which yields the lemma. ■

Definition 3.8 A packet is available after executing the first two steps of the main assignment routine if it did not directly replace a packet that OPT serves.

Note that an available packet might still have indirectly replaced a packet served by OPT . However, the fact that it did not directly replace such a packet allows us to upper bound the value assigned to it in the first two steps of the assignment routine. We will use this fact later.

The sequence assignment routine presented in Figure 4 assigns the value of all packets from $\text{REJOPT}(S)$.

Sequence Assignment Routine(S):

1. For interval \mathcal{I}_i s.t. $1 \leq i \leq k$, assign the value of each of the $|\text{PG}_i \setminus \text{OPT}_i| + |\text{OUT}_i|$ most valuable packets from REJOPT_i to a packet in $(\text{PG}_i \setminus \text{OPT}_i) \cup \text{REPLACE}_i$.
2. Let UNASG_i be the subset of REJOPT_i containing packets that remained unassigned, $\text{UNASG}(S) = \cup_{i=1}^k \text{UNASG}_i$, and $\text{SMALL}(S)$ be the subset of $\text{UNASG}(S)$ containing the $\max(|\text{UNASG}(S)| - B/2, 0)$ packets with the lowest value. Find a set $\text{EXTRA}(S)$ of packets from $(\text{PG}(S) \setminus \text{PG}_m) \cup \text{EARLY}(S)$ s.t. $|\text{EXTRA}(S)| = |\text{SMALL}(S)|$ and the value of the l -th largest packet in $\text{EXTRA}(S)$ is at least as large as that of the l -th largest packet in $\text{SMALL}(S)$ divided by β . For each *unavailable* packet in $\text{EXTRA}(S)$, remove from it a $\frac{2}{\beta}$ fraction of its value (this value will be reassigned at the next step).
3. Assign the value of each pair of packets from $\text{SMALL}(S)$ and $\text{UNASG}(S) \setminus \text{SMALL}(S)$ to a pair of *available* packets from $\text{PG}_m \cup \text{REPLACE}_m$ and the packet from $\text{EXTRA}(S)$. Assign to these packets also the value removed from the packet in $\text{EXTRA}(S)$, if any. Do this in such a way that each packet is assigned at most $1 - \epsilon$ times its value.
4. Assign a $1 - 1/\beta$ fraction of the value of each packet from $\text{UNASG}(S)$ that is not yet assigned to an *available* packet in $\text{PG}_m \cup \text{REPLACE}_m$ that has not been assigned any value at Step 3 or the current step of the assignment routine and a $1/\beta$ fraction of its value to some packet from $\text{PG}_m \cup \text{REPLACE}_m$ that has not been assigned any value at Step 3 or the current step of this assignment routine (note that this packet may have been assigned some value by the main routine).

Figure 4: The sequence assignment routine.

For the sake of analysis, we make some simplifying assumptions.

1. For any $1 \leq i \leq k$, $|\text{REJOPT}_i| \geq |\text{PG}_i \setminus \text{OPT}_i| + |\text{OUT}_i|$.
2. No packet from $\text{EXTRA}(S)$ belongs to another overload sequence (the set $\text{EXTRA}(S)$ will be defined later).

We show that the assignment routine is feasible under the assumptions (1) and (2). Then we derive an upper bound on the value assigned to any packet in PG. Finally, we demonstrate how to relax these assumptions.

First we will use Lemma 3.1 to show that for each but the $B/2$ largest packets from $\text{UNASG}(S)$, PG has scheduled some extra packet with value that constitutes at least a $1/\beta$ fraction of its value. The following crucial lemma explicitly constructs the set $\text{EXTRA}(S)$ for the sequence assignment routine. Basically, this set will consist of packets that PG served at times at which OPT served other (presumably more valuable) packets.

Lemma 3.3 *For an overload sequence S , we can find a set $\text{EXTRA}(S)$ of packets from $(\text{PG}(S) \setminus \text{PG}_m) \cup \text{EARLY}(S)$ such that $|\text{EXTRA}(S)| = |\text{SMALL}(S)|$ and the value of the l -th largest packet in $\text{EXTRA}(S)$ is at least as large as that of the l -th largest packet in $\text{SMALL}(S)$ divided by β .*

Proof: Recall that $|\text{SMALL}(S)| = \max(|\text{UNASG}(S)| - B/2, 0)$. To avoid trivialities, assume that $|\text{UNASG}(S)| > B/2$ and let $x_i = |\text{UNASG}_i|$.

By assumption (1),

$$x_i = |\text{REJOPT}_i| - |\text{PG}_i \setminus \text{OPT}_i| - |\text{OUT}_i| \geq 0.$$

Thus,

$$\begin{aligned} |\text{OPT}_i \setminus \text{PRMOPT}_i| &= |\text{REJOPT}_i| + |\text{OPT}_i \cap \text{PG}_i| \\ &= x_i + |\text{PG}_i \setminus \text{OPT}_i| + |\text{OPT}_i \cap \text{PG}_i| + |\text{OUT}_i| \\ &= x_i + |\text{PG}_i| + |\text{OUT}_i|. \end{aligned}$$

Let PREDOPT_i be the set of packets from $\text{OPT}_i \setminus \text{PRMOPT}_i$ that have been scheduled by OPT before time t_s^i . We must have $|\text{PREDOPT}_i| \geq x_i$ since the buffer of PG is full of packets from $\cup_{j=\min(i,m)}^k \text{BELONG}_j$ at time t_s^i . If it is not the case, then we obtain that the schedule of OPT is infeasible using an argument similar to that of Lemma 3.2.

We also claim that $|\text{PREDOPT}_m| \geq \sum_{i=m}^k x_i$ and PREDOPT_m contains at least $\sum_{i=m+1}^k x_i$ packets with value greater than or equal to w_m . Otherwise, the schedule of OPT is either infeasible or can be improved by switching a packet $p \in \cup_{i=m+1}^k (\text{OPT}_i \setminus \text{PG}_i)$ and a packet $p' \in \text{BELONG}_m \setminus \text{OPT}_m$ s.t. $v(p) < w_m$ and $v(p') \geq w_m$.

Let MAXUP_j be the set of the x_j most valuable packets from PREDOPT_j for $1 \leq j < m$. It must be the case that the value of the l -th largest packet in MAXUP_j is at least as large as that of the l -th largest packet in UNASG_j for $1 \leq l \leq |\text{UNASG}_j|$. That is due to the fact that by Observation 2 the x_j least valuable packets from REJOPT_j are also the x_j least valuable packets from $\text{OPT}_j \setminus \text{PRMOPT}_j$.

Now for j starting from k and down to $m - 1$, let MAXDOWN_j be the set containing arbitrary x_j packets from $\text{PREDOPT}_m \setminus (\cup_{i=m+1}^{j-1} \text{MAXDOWN}_i)$ with value at least w_m . (Recall that PREDOPT_m contains at least $\sum_{i=m+1}^k x_i$ packets with value greater than or equal to w_m .) Finally, let MAXUP_m be the set of the x_m most valuable packets from $\text{PREDOPT}_m \setminus (\cup_{i=m+1}^k \text{MAXDOWN}_i)$. Clearly, any packet in MAXDOWN_j is greater than any packet in REJECT_j for $m + 1 \leq j < k$. Similarly to the case of $j < m$, we obtain that the value of the l -th largest packet in MAXUP_m is at least as large as that of the l -th largest packet in UNASG_m for $1 \leq l \leq |\text{UNASG}_m|$.

Let $\text{MAXP}(S) = (\cup_{i=1}^m \text{MAXUP}_i) \cup (\cup_{i=m+1}^k \text{MAXDOWN}_i)$ and let t_i be the time at which OPT schedules the i -th packet from $\text{MAXP}(S)$. We also denote by $\text{MAXP}(S, t_i)$ the set of packets from $\text{MAXP}(S)$ that arrived by

time t_i . For $B/2 + 1 \leq n \leq |\text{UNASG}(S)|$, let $\text{LARGE}(t_n)$ be the set of $B/2$ largest packets in $\text{MAXP}(S, t_i)$. We define

$$\text{EXTRA}(S) = \bigcup_{i=B/2+1}^{|\text{UNASG}(S)|} \text{PG}(t_i).$$

That is, the set $\text{EXTRA}(S)$ consists of the packets served by PG while OPT served packets from the PREDOPT sets.

We show that at time t_i , PG schedules a packet with value of at least w'/β , where w' is the minimal value among packets in $\text{LARGE}(t_i)$. If all packets from $\text{LARGE}(t_i)$ are present in the buffer at time t_i then we are done by Lemma 3.1. Note that the earliest packet from $\text{LARGE}(t_i)$ arrived before or at time $t_i - B/2$ since OPT schedules all of them by time t_i . In case a packet p from $\text{LARGE}(t_i)$ has been dropped, then by the definition of PG and the construction of the intervals, PG schedules at this time a packet that has value at least $v(p) > w'/\beta$.

Observe that the last packet from $\text{EXTRA}(S)$ is sent earlier than t_s^m and therefore $\text{EXTRA}(S) \cap \text{PG}_m = \emptyset$. Therefore, the set defined above satisfies the condition of the lemma. ■

Now we are ready to state the main theorems.

Theorem 3.4 *The mapping routine is feasible.*

Proof: If all assignments are done at Step 1 or Step 2 of the main assignment routine then we are done.

Else, consider an overload sequence S that is processed by the sequence assignment routine. By Lemma 3.2, we obtain that the number of unassigned packets is bounded from above by:

$$\begin{aligned} |\text{UNASG}(S)| &= |\text{REJOPT}(S)| + |\text{PG}(S) \cap \text{OPT}(S)| - |\text{PG}(S)| - |\text{OUT}(S)| \\ &= |\text{OPT}(S)| - |\text{PRMOPT}(S)| - |\text{PG}(S)| - |\text{OUT}(S)| \\ &\leq B - |\text{PRMOPT}(S)| - |\text{PREVP}(S)|. \end{aligned} \tag{1}$$

Observe that each packet p that replaces a packet p' with value w can be assigned a value of w if $p' \in \text{OPT}$. In addition, if p' belongs to another overload sequence S' then p can be assigned an extra value of w at Step 3 or Step 4 of the sequence assignment routine.

Remember that $\text{PGREP}_m = \text{PG}_m \cup \text{REPLACE}_m$. Let ASG_1 be the subset of $\text{PG}_m \cup \text{REPLACE}_m$ containing the unavailable packets after the first two steps of the main assignment routine. By definition, every such packet directly replaced a packet from OPT . We show that all packets directly replaced by packets from ASG_1 belong to $\text{PRMOPT}(S) \cup \text{PREVP}(S)$. Consider such a packet p . If p is directly preempted by a packet from ASG_1 then we are done. Else, we have that p is preempted by a packet p' , which is pushed out (directly or indirectly) by a packet from ASG_1 . In this case, by the overload sequence construction, p' must belong to S , and therefore p belongs to $\text{PRMOPT}(S) \cup \text{PREVP}(S)$. Thus,

$$|\text{ASG}_1| \leq |\text{PRMOPT}(S)| + |\text{PREVP}(S)|.$$

We denote by ASG_2 the subset of $\text{PG}_m \cup \text{REPLACE}_m$ containing packets that have been assigned some value at Step 3 of the sequence assignment routine. We have

$$|\text{ASG}_2| = 2 \max(|\text{UNASG}(S)| - B/2, 0).$$

Finally, let ASG_3 and ASG_4 be the subsets of $PG_m \cup REPLACE_m$ containing packets that have been assigned at Step 4 of the sequence assignment routine a $1 - 1/\beta$ and a $1/\beta$ fraction of the value of a packet from $UNASG(S)$, respectively. Then

$$|ASG_3| = |ASG_4| = |UNASG(S)| - 2 \max(|UNASG(S)| - B/2, 0).$$

Now we will show that the assignment is feasible. By (1), we have that

$$|ASG_1| + |ASG_2| + |ASG_3| \leq B$$

while Observation 3 implies that $|PG_m \cup REPLACE_m| \geq B$. Finally,

$$|ASG_4| \leq B - |ASG_2| - |ASG_3|,$$

which follows by case analysis. This implies that during the sequence assignment routine we can always find the packets that we need. ■

Theorem 3.5 *Any packet from PG is assigned at most a $2 - \epsilon$ fraction of its value, where $\epsilon(\beta) > 0$ is a constant depending on β .*

Proof: If all assignments are done at Step 1 or Step 2 of the main assignment routine then obviously no packet is assigned more than a $1 + 1/(\beta - 1)$ fraction of its value. That is due to the fact that each packet of value w may either preempt a packet of value at most w/β or push out another packet p' , of value less than w , replacing the packet(s) transitively preempted by p' .

Next we will derive the ratio f that is used at Step 3 of the sequence assignment routine. Consider a pair of packets $p_1 \in SMALL(S)$, $p_2 \in (UNASG(S) \setminus SMALL(S))$ and a pair of packets p_3, p_4 from $PG_m \cup OUT_m$. Let $p_5 \in EXTRA(S)$ be the extra packet used in the assignment. Note that $v(p_1) \leq v(p_2) \leq w_m$, $\min(v(p_3), v(p_4)) \geq w_m$ and $v(p_5) \geq v(p_1)/\beta$. Let also $v(p_1) = w = w_m - \delta$. The ratio f that accounts for the value of the relevant packets is as follows:

$$f = \frac{v(p_1) + v(p_2) + v(p_5) \cdot \frac{2}{\beta}}{v(p_3) + v(p_4) + v(p_5)} \leq \frac{2w_m - \delta + \frac{2w}{\beta^2}}{2w_m + \frac{w}{\beta}}.$$

In case $\delta \geq \frac{w_m}{\beta}$, we have that

$$f < \frac{2w_m - \frac{w_m}{\beta} + (1 - \frac{1}{\beta})\frac{2w_m}{\beta^2}}{2w_m} = \frac{2 - \frac{1}{\beta} + \frac{2(\beta-1)}{\beta^3}}{2}.$$

If $\delta < \frac{w_m}{\beta}$ then

$$f < \frac{2w_m + (1 - \frac{1}{\beta})\frac{2w_m}{\beta^2}}{2w_m + (1 - \frac{1}{\beta})\frac{w_m}{\beta}} = \frac{2 + \frac{2(\beta-1)}{\beta^3}}{2 + \frac{\beta(\beta-1)}{\beta^3}}$$

for $\beta > 2$. Thus, we obtain that

$$f = \max \left(\frac{2 - \frac{1}{\beta} + \frac{2(\beta-1)}{\beta^3}}{2}, \frac{2 + \frac{2(\beta-1)}{\beta^3}}{2 + \frac{\beta(\beta-1)}{\beta^3}} \right).$$

At this point we can compute the overall ratio. Observe that any packet in ASG_1 and $BELONG(S) \setminus PGREP_m$ can be assigned at most a $1 + \frac{2}{\beta-1} + \frac{1}{\beta}$ fraction of its value, where a fraction of $\frac{1}{\beta}$ is due to Step 4 of the sequence assignment routine. By the construction, all packets in ASG_2 and ASG_3 are available. Thus any packet in ASG_2 and ASG_3 can be assigned at most a $1 + f + \frac{2}{\beta(\beta-1)}$ and a $2 - \frac{1}{\beta} + \frac{2}{\beta(\beta-1)}$ fraction of its value, respectively. Hence, we obtain that no packet is assigned more than a $2 - \epsilon$ fraction of its value, where

$$\epsilon(\beta) = 2 - \min_{\beta} \max \left(1 + \frac{2}{\beta-1} + \frac{1}{\beta}, 1 + f + \frac{2}{\beta(\beta-1)}, 2 - \frac{1}{\beta} + \frac{2}{\beta(\beta-1)} \right).$$

■

Optimizing the value of β , we get that for $\beta = 15$ the competitive ratio of PG is close to 1.983.

Now let us go back to the assumption (1), that is $x_i = |REJOPT_i| - (|PG_i \setminus OPT_i| + |OUT_i|) \geq 0$. We argue that there exist two indices $l \leq m$ and $r \geq m$ s.t. $x_i \geq 0$ for $l \leq i \leq r$ and $x_i \leq 0$ for $1 \leq i < l$ or $l < i \leq k$. In this case we can restrict our analysis to the subsequence of S containing the intervals $\mathcal{I}_l, \dots, \mathcal{I}_r$.

Assume towards a contradiction that there exist two indices i, j s.t. $i < j \leq m$ or $i > j \geq m$, $x_i > 0$ and $x_j < 0$. Then there exist a packet $p \in OPT_i$ and a packet $p' \in PG_j \setminus OPT_j$ s.t. $v(p') > v(p)$. We obtain that the schedule of OPT can be improved by switching p and p' .

It remains to consider the assumption (2), that is no packet from $EXTRA(S)$ belongs to another overload sequence S' . In this case we sharp the bound of Lemma 3.2 applied to both sequences.

Lemma 3.6 *For any two consecutive overload sequences S' and S the following holds: $|OPT(S)| + |OPT(S')| - |PG(S)| - |PG(S')| \leq 2B + |OUT(S)| - |PREVP(S)| - |PREVP(S')| - |EXTRA(S) \cap BELONG(S')|$.*

Proof: According to the proof of Lemma 3.2, $t_f^m - t_l \geq B - |OUT(S)|$ where t_l is the last time during S at which a packet from $BELONG(S)$ has been rejected. Let $z = |EXTRA(S) \cap BELONG(S')|$. We argue that OPT has scheduled at most $t_l + 2B - t_s^1 - |PREVP(S')|$ packets from $BELONG(S) \cup BELONG(S')$. That is due to the fact that the earliest packet from $BELONG(S')$ arrived at or after time $t_s^1 - B + |PREVP(S')|$. Observe that between time t_s^1 and time t_f^k at most $B - z - |PREVP(S)|$ packets outside of $BELONG(S) \cup BELONG(S')$ have been scheduled by PG. Hence, PG has scheduled at least $t_l + z + |PREVP(S)| - t_s^1 - |OUT(S)|$ packets from $BELONG(S) \cup BELONG(S')$, which yields the lemma. ■

Using Lemma 3.6, we can extend our analysis to any number consecutive overload sequences without affecting the resulting ratio since we “gain” $|EXTRA(S) \cap BELONG(S')|$ additional packets that are available for mapping.

3.4 Lower Bounds

In this section we will show a specific lower bound of $\phi \approx 1.618$ on the performance of the PG algorithm for any choice of the parameter β and a general lower bound of 1.419 on the performance of any online algorithm. The latter bound slightly improves the bound of $\sqrt{2} \approx 1.414$ obtained in [2].

Theorem 3.7 *The PG algorithm has a competitive ratio of at least ϕ .*

Proof: Suppose that the buffer is empty at time $t = 0$ and consider the following scenarios. In the first scenario at time $t = 0$, B packets with values $1, \beta, \dots, \beta^B$ arrive one by one. The PG algorithm preempts all of them

but the last packet while OPT schedules all the packets. In this case the ratio between the value of OPT and that of PG is close to $\beta/(\beta - 1)$, for sufficiently large B .

In the second scenario, at time $t = 0$, a burst of B packets of value $1 + \epsilon$ arrives. Then we have k phases, each of length B . The i -th phase takes place during $[B \cdot (i - 1), \dots, (B \cdot i) - 1]$. In every time slot throughout i -th phase, one packet of value $\beta^i + \epsilon$ arrives. Finally, at time $t = Bk$, a burst of B packets of value $\beta^k + \epsilon$ arrives. The PG algorithm schedules all but the last B packets of value $\beta^k + \epsilon$. On the other hand, OPT sends all but the first B packets of value $1 + \epsilon$. Hence, the ratio between the value of OPT and PG is nearly $2 - 1/\beta$ for sufficiently large k .

To optimize the lower bound, i.e., maximize $\min(\beta/(\beta - 1), 2 - 1/\beta)$, we equate both of these ratios: $\beta/(\beta - 1) = 2 - 1/\beta$. We get that $\beta = \frac{3+\sqrt{5}}{2} = \phi + 1$ and thus PG is at least ϕ -competitive. ■

Now let us turn to a general lower bound. Define $v^* = \sqrt[3]{19 + 3\sqrt{33}}$ and $\mathcal{R} = (19 - 3\sqrt{33})(v^*)^2/96 + v^*/6 + 2/3 \approx 1.419$.

Theorem 3.8 *Any online algorithm ALG has a competitive ratio of at least \mathcal{R} .*

Proof: Suppose that ALG maintains a competitive ratio less than \mathcal{R} and let $v = v^*/3 + 4/(3v^*) + 4/3 \approx 2.839$. We define a sequence of packets as follows. At time $t = 1$, B packets with value 1 arrive. At each time $2, \dots, l_1$, a packet of value v arrives, where $t + l_1$ is the time at which ALG serves the first packet of value v (i.e. the time at which there remain no packets of value 1). Depending on l_1 , the sequence either stops at this point or continues with a new phase.

Basically, at the start of phase i , B packets of value v^{i-1} arrive. During the phase, one packet of value v^i arrives at each time step until ALG serves one of them. This is the end of the phase. If the sequence continues until phase n , then in phase n only B packets of value v^{n-1} arrive. Let us denote the length of phase i by l_i for $i = 1, \dots, n - 1$ and define $s_i = \sum_{j=1}^i (l_j v^{j-1})$ for $i = 1, \dots, n$.

If the sequence stops during phase $i < n$, then ALG earns $l_1 + l_2 v + l_3 v^2 + \dots + l_i v^{i-1} + l_i v^i = s_i + l_i v^i$ while OPT can earn at least $l_1 v + l_2 v^2 + \dots + (l_{i-1} + B)v^{i-1} + l_i v^i = v \cdot s_i + Bv^{i-1}$. The implied competitive ratio is $(v \cdot s_i + Bv^{i-1})/(s_i + l_i v^i)$. We only stop the sequence in this phase if this ratio is at least \mathcal{R} , which depends on l_i . We now determine the value of l_i for which the ratio is exactly \mathcal{R} . Note that $l_i v^i = (s_i - s_{i-1}) \cdot v$. We have that $(v \cdot s_i + Bv^{i-1})/(s_i + l_i v^i) = \mathcal{R}$ implies

$$s_i = \frac{v\mathcal{R}s_{i-1} + Bv^{i-1}}{\mathcal{R}(v+1) - v}, s_0 = 0 \Rightarrow s_i = \frac{v^i - (\frac{\mathcal{R}v}{\mathcal{R}(v+1)-v})^i}{(\mathcal{R}-1)v^2} B.$$

It can be seen that $s_i/v^i \rightarrow B/(v^2(\mathcal{R}-1))$ for $i \rightarrow \infty$, since $\mathcal{R}/(\mathcal{R}(v+1) - v) < 1$ for $\mathcal{R} > 1$.

Thus, if under ALG the length of phase i is less than l_i , the sequence stops and the ratio is proved. Otherwise, if ALG continues until phase n , it earns $l_1 + l_2 v + l_3 v^2 + \dots + l_n v^{n-1} + B \cdot v^n = s_n + Bv^n$ whereas OPT can earn at least $l_1 v + l_2 v^2 + \dots + l_n v^n + B \cdot v^n = v \cdot s_n + Bv^n$. The implied ratio is

$$\frac{vs_n + Bv^n}{s_n + Bv^n} = \frac{v\frac{s_n}{v^n} + B}{\frac{s_n}{v^n} + B} \rightarrow \frac{\frac{v}{v^2(\mathcal{R}-1)} + 1}{\frac{1}{v^2(\mathcal{R}-1)} + 1} = \frac{v + v^2(\mathcal{R}-1)}{1 + v^2(\mathcal{R}-1)} = \mathcal{R}.$$

■

4 Uniform Bounded Delay Buffers

In this section we consider the uniform bounded delay model. We show that the value gained by OPT in the B -uniform bounded delay model equals to that of OPT in the FIFO model. Moreover, we demonstrate that OPT does not need a buffer with capacity greater than B . Let us denote by $V_A^M(\sigma)$ the value gained by the algorithm A in the model M (either FIFO or uniform bounded delay (UBD)). A similar claim has been made in [7].

Lemma 4.1 *For any input sequence σ , the value of OPT in the B -uniform bounded delay model with buffer of infinite capacity equals the value of OPT in the FIFO model with buffer of capacity B , that is $V_{\text{OPT}}^{\text{UBD}}(\sigma) = V_{\text{OPT}}^{\text{FIFO}}(\sigma)$.*

Proof: We argue that any feasible schedule in the uniform bounded delay model can be transformed to an equivalent feasible schedule in the FIFO model in which the same set of packets is sent. Assume wlog that OPT in the bounded model schedules all packets that are accepted into the buffer. If it is not the case, one can admit only packets that are eventually sent without affecting the value of the solution. We claim that at any time the buffer of OPT contains at most B packets. Otherwise, the delay of some packet must be greater than B and it has to be dropped. That contradicts to our assumption. The further transformation is done by swapping packets so that the FIFO order is maintained. Note that the FIFO order coincides with the Earliest Deadline First (EDF) order and such a swapping is always possible. ■

Now consider the algorithm SPG in the B -uniform bounded delay model that simulates the PG algorithm in the FIFO model, that is accepts, drops and sends the same packets.

Theorem 4.2 *The competitive ratio of SPG in the B -uniform bounded delay model equals to that of PG in the FIFO model.*

Proof: Suppose that PG is c -competitive in the FIFO model and fix an input sequence σ . By our assumption, $c \cdot V_{\text{PG}}^{\text{FIFO}}(\sigma) \geq V_{\text{OPT}}^{\text{FIFO}}(\sigma)$. Lemma 4.1 implies that $V_{\text{OPT}}^{\text{UBD}}(\sigma) = V_{\text{OPT}}^{\text{FIFO}}(\sigma)$. Clearly, $V_{\text{SPG}}^{\text{UBD}}(\sigma) = V_{\text{PG}}^{\text{FIFO}}(\sigma)$. Therefore, we obtain that $c \cdot V_{\text{SPG}}^{\text{UBD}}(\sigma) \geq V_{\text{OPT}}^{\text{UBD}}(\sigma)$, which establishes the theorem. ■

5 Conclusion

In this paper we study QoS buffering in the FIFO and uniform bounded delay models. Our main results are algorithms in both models for arbitrary packet values that for the first time achieve a competitive ratio strictly better than 2. One of the interesting future research directions is to close a significant gap between the lower and upper bounds. Another open problem is whether we can break the competitive ratio of 2 in the variable bounded delay model, where the delay is specified individually per each packet.

References

- [1] W.A. Aiello, Y. Mansour, S. Rajagopalan and A. Rosén. Competitive Queue Policies for Differentiated Services. In *Proceedings of IEEE Infocom 2000*, pp.431–440.

- [2] N. Andelman, Y. Mansour and A. Zhu. Competitive Queueing Policies for QoS Switches. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'03)*, pp.761–770. ACM/SIAM, 2003.
- [3] Y. Bernet, A. Smith, S. Blake and D. Grossman. A Conceptual Model for Diffserv Routers. *Internet draft*, March 2000.
- [4] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
- [5] D. Clark and J. Wroclawski. An Approach to Service Allocation in the Internet. *Internet draft*, July 1997.
- [6] C. Dovrolis, D. Stiliadis and P. Ramanathan. Proportional Differentiated Services: Delay Differentiation and Packet Scheduling. In *Proceedings of ACM SIGCOMM'99*, pp.109–120. ACM, 1999.
- [7] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber and M. Sviridenko. Buffer Overflow Management in QoS Switches. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC 2001)*, pp.520–529. ACM, 2001.
- [8] A. Kesselman and Y. Mansour. Loss-Bounded Analysis for Differentiated Services. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*, pp.591–600. ACM/SIAM, 2001.
- [9] Z. Lotker and B. Patt-Shamir. Nearly optimal FIFO buffer management for DiffServ. In *Proceedings of the 21st ACM Symposium on Principles of Distributed Computing (PODC 2002)*, pp.134–142. ACM, 2002.
- [10] Y. Mansour, B. Patt-Shamir and O. Lapid. Optimal Smoothing Schedules for Real-Time Streams. In *Proceedings of the 19th ACM Symposium on Principles of Distributed Computing (PODC 2000)*, pp.21–29. ACM, 2000.
- [11] M. May, J. Bolot, A. Jean-Marie, and C. Diot. Simple Performance Models of Differentiated Services Schemes for the Internet. In *Proceedings of IEEE Infocom 1999*, pp.1385–1394, March 1999.
- [12] K. Nichols, V. Jacobson and L. Zhang. A Two-bit Differentiated Services Architecture for the Internet. *Internet draft*, July 1999.
- [13] T. Nandagopal, N. Venkitaraman, R. Sivakumar and V. Bharghavan. Relative Delay Differentiation and Delay Class Adaptation in Core-Stateless Networks. In *Proceedings of IEEE Infocom 2000*, pp.421–430, March 2000.
- [14] V. Paxson and S. Floyd. Wide-Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Transactions on Networking* 3(3):226–244, June 1995.
- [15] S. Sahu, D. Towsley and J. Kurose. A Quantitative Study of Differentiated Services for the Internet. In *Proceedings of IEEE Global Internet'99*, pp.1808–1817, December 1999.
- [16] N. Semret, R. Liao, A. Campbell and A. Lazar. Peering and Provisioning of Differentiated Internet Services. In *Proceedings of INFOCOM 2000*, pp.414–420, March 2000.
- [17] D. Sleator and R. Tarjan. Amortized Efficiency of List Update and Paging Rules. *Communications of the ACM*, 28:202–208, 1985.

- [18] I. Stoica and H. Zhang. Providing Guaranteed Services without Per Flow Management. In *Proceedings of ACM SIGCOMM'99*, pp.81–94. ACM, 1999.
- [19] A. Veres and M. Boda. The Chaotic Nature of TCP Congestion Control. In *Proceedings of IEEE Infocom 2000*, pp.1715–1723, March 2000.