# Improved Competitive Guarantees for QoS Buffering


Alex Kesselman* alx@cs.tau.ac.il

Yishay Mansour* mansour@cs.tau.ac.il

Rob van Stee† vanstee@uni-freiburg.de



**Abstract**

We consider a network providing Differentiated Services (Diffserv) which allow Internet service providers (ISP) 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 *bounded-delay* type, each packet has a maximum delay time by which it must be transmitted, or otherwise it is lost. In both models, the buffer space is limited, and packets are lost if the buffer is full. 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 for 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 bounded delay model that 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 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


*School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel.

†Institute for Computer Science, Albert-Ludwigs-Universitt Freiburg, Georges-Khler-Allee 79, D-79110 Freiburg i. Br., Germany.


Work supported by the Deutsche Forschungsgemeinschaft, Project AL 464/3-1, and by the European Community, Projects APPOL and APPOL II.

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 new *bounded delay model*. This model is warranted by networks that guarantee the QoS parameter of end-to-end delay. Specifically, each packet arrives with a prescribed *allowed delay* time. A packet must be transmitted within this time, else it is lost. Note that in the bounded delay 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 $\sigma$ by $V_{\text{ALG}}(\sigma)$.

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

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 policy can have a competitive ratio better than $5/4$, and demonstrates that the greedy policy is at least $4$-competitive. In [7] the greedy policy has been shown to achieve the competitive ratio of $2$ in both FIFO and the bounded delay models. Our model is identical to that of [7]. The loss of a policy is analyzed in [8], where they present a policy 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 policy. The problem of whether the competitive ratio of $2$ of the natural greedy policy can be improved has been open for a long time. It 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 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, we assume that in the 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 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. Packets may arrive to the queue at any time and send events are synchronized with time. 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 the transmitted packets. Time is slotted. At the beginning of a time slot a set of packets (possibly empty) arrives and at the end of time slot a packet is scheduled if any. 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 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 $Q(t)$. We consider two variants of the bounded delay model. In the *uniform* bounded delay model, there is a single fixed bound on the delay of all packets, and in the *variable* bounded delay model, there may be different delay bounds for different packets. This last model is equivalent to real-time scheduling of unit-size weighted jobs with deadlines.

# 3 FIFO Buffers

## 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.

The parameter of $PG$ is the preemption factor $\beta$. For sufficiently large value of $\beta$, $PG$ performs like the greedy algorithm and only drops packets in case of overflow. On the other hand, too small values of $\beta$ can cause excessive preemptions of packets and a large loss of value. Thus, we need to optimize the value of $\beta$ in order

**The $\beta$-Preemptive Greedy Algorithm.**

1. When a packet $p$ of value $v(p)$ arrives, drop the first packet $p'$ in the FIFO order such that $v(p') \leq v(p)/\beta$, if any ($p'$ is *preempted*).
2. Accept $p$ if there is free space in the buffer.
3. If $p$ is not accepted then drop it if the buffer is full and $v(p)$ is less than the minimal value among the packets currently in the buffer ($p$ is *rejected*).
4. Otherwise, drop the packet $p'$ with the minimal value from the buffer and accept $p$ ($p$ *pushes out* $p'$).

Figure 1: Algorithm PG.

to achieve a balance between maximizing the current throughput and minimizing the potential future loss.

Next we will introduce a few useful definitions. We say that a packet $p$ *transitively preempts* a packet $p'$ if $p$ either preempts $p'$ or $p$ preempts or pushes out another packet $p''$, which in its turn transitively preempts $p'$. We also say that a packet $p$ *replaces* a packet $p'$ if (1) $p$ transitively preempts $p'$ and (2) $p$ is eventually scheduled. The *chain of replacement* contains all packets transitively preempted by $p$. We say that $p$ *directly replaces* $p'$ if in the chain of replacement between them no packet except $p'$ is preempted (e.g. $p$ may push out $p''$ that preempts $p'$).

## 3.2 Overload Intervals

The main concept of the proof is that of *overload intervals*. Before introducing a formal definition, we will give some intuition. Consider a time $t$ at which a packet of value $\alpha$ is rejected or pushed out and $\alpha$ is the largest value among the packets that are rejected or pushed out at this time. Note that all packets in the buffer at the end of time slot $t$ have value at least $\alpha$. Such an event defines an $\alpha$-*overloaded* interval $\mathcal{I} = [t_s, t_f)$. The interval starts at the earliest time $t_s$ before time $t$ such that only packets of value $\alpha$ or greater are served in $[t_s, t]$, or after the preceding overload interval with a higher overload value ends. Then if no packet is dropped after time $t$, $\mathcal{I}$ ends at the last time at which a packet of value at least $\alpha$ is scheduled (see Figure 2). In case at some time $t' > t$ a packet of value $\gamma$ is rejected or pushed out, $\gamma$ is the largest value among the packets that are rejected or pushed out at this time, and a packet from $Q(t)$ 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 $\gamma < \alpha$ then we start a new interval when the first packet from $Q(t') \setminus Q(t)$ is eventually scheduled if any. Otherwise, if all packets from $Q(t') \setminus Q(t)$ are preempted, we create a zero length interval $\mathcal{I}' = [t_f, t_f)$ whose overload value is $\gamma$. Next we define the notion of overload interval more formally.

**Definition 3.1** *An $\alpha$-*overflow *takes place at time $t$ if a packet of value $\alpha$ is rejected or pushed out at this time, where $\alpha$ 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.*

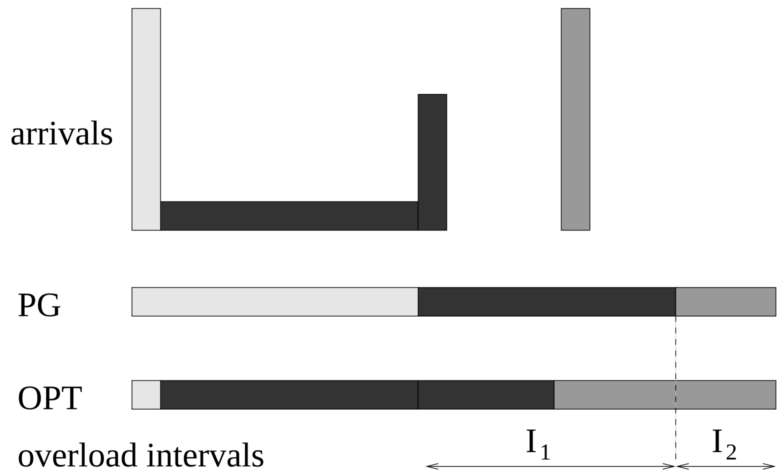


Figure 2: An example of overload interval.

We construct overload intervals starting from the highest overload value and ending with the lowest overload value.

**Definition 3.3** *An interval $\mathcal{I} = [t_s, t_f)$, with $t_f \geq t_s$, is an $\alpha$-*overloaded *interval if at least one packet with value $\alpha$ associated with $\mathcal{I}$ is rejected or pushed out, no packet associated with $\mathcal{I}$ with higher value is rejected or pushed out, only packets with value at least $\alpha$ are served during $\mathcal{I}$, all packets served during $\mathcal{I}$ arrive no later than the last $\alpha$-overflow, and $\mathcal{I}$ is a maximal such interval taking into account overload intervals with higher overload values.*

We note that overload intervals are *disjoint*.

**Definition 3.4** *A packet $p$* belongs to *an $\alpha$-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 or pushed out no later than the last $\alpha$-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, rejected or pushed out.*

Thus directly before and after such an interval, either the buffer is empty, a lower-value packet is served (possibly as part of a $\gamma$-overloaded interval with $\gamma < \alpha$), or there is a $\gamma$-overloaded interval with $\gamma > \alpha$. Whenever an $\alpha$-overloaded interval $\mathcal{I}$ is immediately followed by a $\gamma$-overloaded interval $\mathcal{I}'$ with $\gamma > \alpha$, we have that in the first time step of $\mathcal{I}'$ a packet of value $\gamma$ is rejected or pushed out.

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

**Observation 1** *Any packet that has been rejected or pushed out belongs to exactly one overload interval.*

Next we introduce some useful definitions related to an overload interval.

**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 four distinct subsets: scheduled packets (*$\text{PG}(\mathcal{I})$*), preempted packets (*$\text{PREEMPT}(\mathcal{I})$*), rejected packets (*$\text{REJECT}(\mathcal{I})$*) and packets that were pushed out (*$\text{PUSHED}(\mathcal{I})$*). 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* $\mathcal{I}_1 = [t_s^1, t_f^1), \mathcal{I}_2 = [t_s^2, t_f^2)$, $\ldots, \mathcal{I}_k = [t_s^k, t_f^k)$ *with overload values* $w_1, \ldots, w_k$ *such that* $t_f^i \leq t_s^{i+1}$ *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 $\text{BELONG}(\mathcal{I}_i), \text{PG}(\mathcal{I}_i), \ldots$ by $\text{BELONG}_i, \text{PG}_i, \ldots$ 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* $\text{REJECT}_i \cup \text{PUSHED}_i$ *have value at most* $w_i$ *while all packets in* $\text{PG}_i$ *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 $\beta$. Optimizing the value of $\beta$, 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 $\alpha$ that is scheduled by OPT, it schedules another packet of value $\alpha$ and an additional packet with a non-negligible value, roughly $\alpha/\beta$, which allows us to break the ratio of $2$, achieved by the greedy algorithm.

In the sequel we fix an input sequence $\sigma$. 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**$(\sigma)$**:**

1. Assign the value of each packet from $\text{PG} \cap \text{OPT}$ to itself.
2. Assign the value of each packet from DROP that has been *preempted* 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 packet from DROP that belongs to the sequence under consideration and has been *rejected* or *pushed out* using the assignment routine for the overload sequence.

Figure 3: The main routine.

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

$$\begin{aligned} \text{OPT}_i &= \text{OPT} \cap \text{BELONG}_i, \\ \text{SHARED}_i &= \text{OPT} \cap \text{PG}_i, \end{aligned}$$

$$\begin{aligned} \text{OVFLOPT}_i &= \text{OPT} \cap (\text{REJECT}_i \cup \text{PUSHED}_i), \\ \text{PRMOPT}_i &= \text{OPT} \cap \text{PREEMPT}_i \end{aligned}$$

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

**Definition 3.7** *For $1 \le i \le k$, $\text{OUT}_i$ is 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 $\text{DROP}(S)$.

For the sake of analysis, we make some simplifying assumptions. Afterward we show how to relax them.

1. For any $1 \le i \le k$ the number of packets in $\text{OVFLOPT}_i$ is not less than the number of packets in $\text{PG}_i \setminus \text{SHARED}_i$ plus the number of packets in $\text{OUT}_i$, that is $|\text{OVFLOPT}_i| \ge |\text{PG}_i \setminus \text{SHARED}_i| + |\text{OUT}_i|$.
2. No packet from $\text{EXTRA}(S)$ belongs to another overload sequence (the set $\text{EXTRA}(S)$ will be defined later).

We say that a packet is *available* after executing the first two steps of the main assignment routine if it has been assigned at most a $1 + \frac{2}{\beta(\beta-1)}$ fraction of its value. (The meaning of this definition will become clear later.) The sequence assignment routine presented on Figure 4 assigns the value of all packets from $\text{OVFLOPT}(S)$.

Next we show that the mapping 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 need auxiliary lemmas.

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

**Claim 3.1** *For an overload sequence $S$ the following holds: $|\text{OPT}(S)| - |\text{PG}(S)| \le B + |\text{OUT}(S)| - |\text{PREVP}(S)|$.*

**Proof:** Let $t'$ be the last time during $S$ at which a packet from $\text{BELONG}(S)$ has been rejected or pushed out. It must be the case that $t_f^k - t' \ge B - |\text{OUT}(S)|$ 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 claim. ∎

The following lemma shows that if the buffer contains a large number of "valuable" packets then $PG$ sends packets with non-negligible value.

**Sequence Assignment Routine**$(S)$**:**

1. For interval $\mathcal{I}_i$ s.t. $1 \leq i \leq k$, assign the value of each of the $|\text{PG}_i \setminus \text{SHARED}_i| + |\text{OUT}_i|$ most valuable packets from $\text{OVFLOPT}_i$ to a packet in $(\text{PG}_i \setminus \text{SHARED}_i) \cup \text{REPLACE}_i$.

2. Let $\text{UNASG}_i$ be the subset of the $\text{OVFLOPT}_i$ of packets that remained unassigned, $\text{UNASG}(S) = \cup_{i=1}^{k} \text{UNASG}_i$, $\text{SMALL}(S)$ be the subset of $\text{UNASG}(S)$ containing the $\max(|\text{UNASG}(S)| - B/2, 0)$ packets with the lowest value and $\text{PGREP}_m = \text{PG}_m \cup \text{REPLACE}_m$. 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 $\beta$. For each packet from $\text{EXTRA}(S)$ that is assigned more than a $1 + \frac{2}{\beta(\beta-1)}$ fraction of its value, 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 $\text{PGREP}_m$ and a packet from $\text{EXTRA}(S)$ so that each packet is assigned a $f \leq 1 - \epsilon$ fraction of its value. (The proper value of $f$ will be determined later.)

4. Assign a $1 - 1/\beta$ fraction of the value of each packet from $\text{UNASG}(S)$ that is unassigned yet to an *available* packet in $\text{PGREP}_m$ and and a $1/\beta$ fraction of its value to some packet from $\text{PGREP}_m$ 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.

**Lemma 3.2** *If at time $t$, $|Q(t, \geq w)| \geq B/2$ and the earliest packet from $Q(t, \geq w)$ arrived before or at time $t - B/2$ then the packet scheduled at the next time slot has value at least $w/\beta$.*

**Proof:** Let $p$ be the first packet from $Q(t, \geq w)$ 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 $w/\beta$ 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 $|X| \leq B$. 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 $w/\beta$, if any. This shows that all packets in $X$ have been either served or dropped by time $t$. ∎

Next we will use Lemma 3.2 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.

**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 $\beta$.*

**Proof:** Recall that $|\text{SMALL}(S)| = \max(|\text{UNASG}(S)| - B/2, 0)$. To avoid trivialities, assume that $|\text{UNASG}(S)| > B/2$. Let us denote $|\text{UNASG}_i|$ by $x_i$ and the set of packets from $\text{OPT}_i \setminus \text{PRMOPT}_i$ that have been scheduled by

OPT before time $t_s^i$ by $\text{PREDOPT}_i$. Note that

$$x_i = |\text{OVFLOPT}_i| - |\text{PG}_i \setminus \text{SHARED}_i| - |\text{OUT}_i|.$$

We argue that for any interval $\mathcal{I}_j$, $|\text{PREDOPT}_j| \geq x_j$. If it is not the case then we obtain that the schedule of OPT is unfeasible by an argument similar to that of Claim 3.1 (observe that $|\text{OPT}_j| \geq |\text{PG}_j| + |\text{OUT}_j|$). We also claim that $|\text{PREDOPT}_m| \geq \sum_{i=m}^{k} x_i$ and $\text{PREDOPT}_m$ 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 unfeasible or can be improved by switching a packet $p \in \cup_{i=m+1}^{k}(\text{OPT}_i \setminus \text{SHARED}_i)$ 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 $\text{MAXUP}_j$ be the set of the $x_j$ most valuable packets from $\text{PREDOPT}_j$ for $1 \leq j < m$. It must be the case that the value of the $l$-th largest packet in $\text{MAXUP}_j$ is at least as large as that of the $l$-th largest packet in $\text{UNASG}_j$ 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 $\text{OVFLOPT}_j$ are also the $x_j$ least valuable packets from $\text{OVFLOPT}_j \cup \text{SHARED}_j$. Now for $j$ starting from $k$ and down to $m-1$, let $\text{MAXDOWN}_j$ 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 $w_m$. Finally, let $\text{MAXUP}_m$ 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 $\text{MAXDOWN}_j$ is greater than any packet in $\text{REJECT}_j$ for $m+1 \leq j < k$. Similarly to the case of $j < m$, one can show that the value of the $l$-th largest packet in $\text{MAXUP}_m$ is at least as large as that of the $l$-th largest packet in $\text{UNASG}_m$ 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 $t_n$ be the time at which OPT schedules $n$-th packet from $\text{MAXP}(S)$. We also denote by $\text{MAXP}(S, t_n)$ the set of packets from $\text{MAXP}(S)$ that arrived by time $t_n$. Consider time $t_n$ s.t. $B/2+1 \leq n \leq |\text{UNASG}(S)|$ and let $\text{LARGE}(t_n)$ be the set of $B/2$ largest packets in $\text{MAXP}(S, t_n)$. We show that at time $t_n$, $PG$ schedules a packet with value of at least $w'/\beta$, where $w'$ is the minimal value among packets in $\text{LARGE}(t_n)$. If all packets from $\text{LARGE}(t_n)$ are present in the buffer at time $t_n$ then we are done by Lemma 3.2. Note that the earliest packet from $\text{LARGE}(t_n)$ arrived before or at time $t_n - B/2$ since OPT schedules all of them by time $t_n$. In case a packet $p$ from $\text{LARGE}(t_n)$ has been dropped, by the definition of $PG$ and the construction of the intervals, the packet scheduled at this time has value at least $v(p) > w'/\beta$.

We define the set

$$\text{EXTRA}(S) = \cup_{n=B/2+1}^{|\text{UNASG}(S)|}\text{PG}(t_n).$$

Observe that the last packet from $\text{EXTRA}(S)$ is sent earlier than $t_s^m$ and thus $\text{EXTRA}(S) \cap \text{PG}_m = \emptyset$. It is easy to see that 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.

Consider an overload sequence $S$ that is processed by the sequence assignment routine. By Claim 3.1, we obtain that the number of unassigned packets is bounded from above by:

$$|\text{UNASG}(S)| \leq B - |\text{PRMOPT}(S)| - |\text{PREVP}(S)| \tag{1}$$

since

$$|\text{UNASG}(S)| = |\text{OVFLOPT}(S)| + |\text{SHARED}(S)| - |\text{PG}(S)| - |\text{OUT}(S)|$$

and

$$|\text{OPT}(S)| = |\text{SHARED}(S)| + |\text{OVFLOPT}(S)| + |\text{PRMOPT}(S)|.$$

Observe that each packet $p$ that replaces a packet $p'$ with value $w$ can be assigned a value of $w$ if $p' \in$ 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 $\text{ASG}^1$ be the subset of $\text{PGREP}_m$ containing packets each of which has been assigned at the first two steps of the main assignment routine more than a $1 + \frac{2}{\beta(\beta-1)}$ fraction of its value. By the mapping construction, every such packet must have directly replaced a packet from $OPT$. We show that all packets directly replaced by packets from $\text{ASG}^1$ belong to $\text{PRMOPT}(S) \cup \text{PREVP}(S)$. Consider such a packet $p$. If $p$ is directly preempted by a packet from $\text{ASG}^1$ then we are done. Else, suppose that $p$ is pushed out by a packet $p'$, which is transitively preempted by a packet from $\text{ASG}^1$. In this case by the overload sequence construction, $p$ must belong to $S$. Thus, we have that:

$$|\text{ASG}^1| \leq |\text{PRMOPT}(S)| + |\text{PREVP}(S)|.$$

We denote by $\text{ASG}^2$ the subset of $\text{PGREP}_m$ containing packets that have been assigned some value at Step 3 of the sequence assignment routine. Note that

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

Finally, let $\text{ASG}^3$ and $\text{ASG}^4$ be the subsets of $\text{PGREP}_m$ 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 $\text{UNASG}(S)$, respectively. Note that

$$|\text{ASG}^3| = |\text{ASG}^4| = |\text{UNASG}(S)| - 2\max(|\text{UNASG}(S)| - B/2, 0).$$

Now we will show that the assignment is feasible, i.e. the pairwise intersection of $\text{ASG}^1$, $\text{ASG}^2$ and $\text{ASG}^3$ is empty and $\text{ASG}^4 \cap (\text{ASG}^2 \cup \text{ASG}^3) = \emptyset$ . By (1) we have that

$$|\text{ASG}^1| + |\text{ASG}^2| + |\text{ASG}^3| \leq B.$$

Obviously,

$$|\text{ASG}^4| \leq B - |\text{ASG}^2| - |\text{ASG}^3|.$$

We are done since Observation 3 implies that the number of packets in $\text{PGREP}_m$ is at least $B$. ∎

**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 $\beta$ for $\beta > 2$.*

**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 $w/\beta$ 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 $p_1 \in \text{SMALL}(S)$, $p_2 \in (\text{UNASG}(S) \setminus \text{SMALL}(S))$ and a pair of packets $p_3$, $p_4$ from $\text{PG}_m \cup$

$\text{OUT}_m$. Let $p_5 \in \text{EXTRA}(S)$ 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

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

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 are ready to compute the overall ratio. It is easy to see that any packet in $\text{ASG}^1$ and $\text{BELONG}(S) \setminus \text{PGREP}_m$ 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 $\text{ASG}^2$ and $\text{ASG}^3$ are available. Thus any packet in $\text{ASG}^2$ and $\text{ASG}^3$ 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

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

∎

Now let us go back to the assumption (1), that is $x_i = |\text{OVFLOPT}_i| - (|\text{PG}_i \setminus \text{SHARED}_i| + |\text{OUT}_i|) \geq 0$. 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 $\mathcal{I}_l, \ldots, \mathcal{I}_r$. Otherwise, there exist two indices $i$, $j$ s.t. $i < j \leq m$ or $i > j \geq m$, $x_i > 0$ and $x_j < 0$. Thus, there are a packet $p \in \text{OPT}_i$ and a packet $p' \in \text{PG}_j \setminus \text{OPT}_j$ 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 $\text{EXTRA}(S)$ belongs to another overload sequence $S'$. In this case we sharp the bound of Claim 3.1 applied to both sequences.

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

**Proof:** According to the proof of Claim 3.1, $t_f^m - t_l \geq B - |\text{OUT}(S)|$ where $t_l$ is the last time during $S$ at which a packet from $\text{BELONG}(S)$ has been rejected or pushed out. Let $z = |\text{EXTRA}(S) \cap \text{BELONG}(S')|$. We argue that

$\text{OPT}$ has scheduled at most $t_l + 2B - {t'}_s^1 - |\text{PREVP}(S')|$ packets from $\text{BELONG}(S) \cup \text{BELONG}(S')$. That is due to the fact that the earliest packet from $\text{BELONG}(S')$ arrived at or after time ${t'}_s^1 - B + |\text{PREVP}(S')|$. Observe that between time ${t'}_s^1$ and time $t_f^k$ at most $B - z - |\text{PREVP}(S)|$ packets outside of $\text{BELONG}(S) \cup \text{BELONG}(S')$ have been scheduled by $PG$. Hence, $PG$ has scheduled at least $t_l + z + |\text{PREVP}(S)| - {t'}_s^1 - |\text{OUT}(S)|$ packets from $\text{BELONG}(S) \cup \text{BELONG}(S')$, which yields the claim. ∎

Using Claim 3.6, we can extend our analysis to any number consecutive overload sequences without affecting the resulting ratio.

### 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 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 $\phi$.*

**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 $1, \beta, \ldots, \beta^B$ arrive one by one. The $PG$ algorithm preempts all of them but the last packet while $\text{OPT}$ schedules all the packets. Thus, the ratio between the value of $\text{OPT}$ and $PG$ is close to $\beta/(\beta - 1)$ for sufficiently large $B$.

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

To optimize the lower bound, i.e. maximize $lb = \min(\beta/(\beta - 1), 2 - 1/\beta)$, 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 $lb = \phi$. ∎

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 $A$ has a competitive ratio of at least $\mathcal{R}$.*

**Proof:** Suppose that $\text{ALG}$ 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 $2, \ldots, l_1$, a packet of value $v$ arrives, where $t + l_1$ is the time at which $\text{ALG}$ 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 $\text{ALG}$ 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 $i = 1, \ldots, n - 1$ and define $s_i = \sum_{j=1}^{i}(l_j v^{j-1})$ for $i = 1, \ldots, n$.

If the sequence stops during phase $i < n$, then ALG earns $l_1 + l_2 v + l_3 v^2 + \ldots + l_i v^{i-1} + l_i v^i = s_i + l_i v^i$ while OPT can earn at least $l_1 v + l_2 v^2 + \ldots + (l_{i-1} + B)v^{i-1} + l_i v^i = v \cdot s_i + Bv^{i-1}$. 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 $s_i/v^i \to B/(v^2(\mathcal{R}-1))$ for $i \to \infty$, 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 $l_1 + l_2 v + l_3 v^2 + \ldots + l_n v^{n-1} + B \cdot v^n = s_n + Bv^n$ whereas OPT can earn at least $l_1 v + l_2 v^2 + \ldots + l_n v^n + B \cdot v^n = v \cdot s_n + Bv^n$. The implied ratio is

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

∎

# 4 Bounded Delay Buffers

In this section we consider the 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 bounded delay (BD)). A similar claim has been made in [7].

**Lemma 4.1** *For any input sequence $\sigma$, 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 $V_{\text{OPT}}^{BD}(\sigma) = V_{\text{OPT}}^{FIFO}(\sigma)$.*

**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 become 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. ∎

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 $\sigma$. By our assumption, $c \cdot V_{PG}^{FIFO}(\sigma) \geq V_{\mathrm{OPT}}^{FIFO}(\sigma)$. Lemma 4.1 implies that $V_{\mathrm{OPT}}^{BD}(\sigma) = V_{\mathrm{OPT}}^{FIFO}(\sigma)$. Clearly, $V_{SPG}^{BD}(\sigma) = V_{PG}^{FIFO}(\sigma)$. Thus, we obtain that $c \cdot V_{SPG}^{BD}(\sigma) \geq V_{\mathrm{OPT}}^{BD}(\sigma)$, which yields the theorem. ∎

# 5 Conclusion

In this paper we study QoS buffering in the FIFO and uniform bounded delay models. Our main result consists of 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.

# References


[1] W. A. Aiello, Y. Mansour, S. Rajagopolan and A. Rosén, "Competitive Queue Policies for Differentiated Services," *Proceedings of INFOCOM 2000*, pp. 431-440.

[2] N. Andelman, Y. Mansour and An Zhu, "Competitive Queueing Policies for QoS Switches," *The 14th ACM-SIAM SODA*, Jan. 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", *Proceedings of ACM SIGCOMM'99*, pp. 109-120.

[7] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber and M. Sviridenko, "Buffer Overflow Management in QoS Switches," *Proceedings of STOC 2001*, pp. 520-529.

[8] A. Kesselman and Y. Mansour, "Loss-Bounded Analysis for Differentiated Services," *Proceedings of SODA 2001*, pp. 591-600.

[9] Z. Lotker and B. Patt-Shamir, "Nearly optimal FIFO buffer management for DiffServ," *Proceedings of PODC 2002*, pp. 134-142.

[10] Y. Mansour, B. Patt-Shamir and Ofer Lapid, "Optimal Smoothing Schedules for Real-Time Streams," *Proccedings of PODC 2000*, pp. 21-29.

[11] M. May, J. Bolot, A. Jean-Marie, and C. Diot, "Simple Performance Models of Differentiated Services Schemes for the Internet," *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 Differentation and Delay Class Adaptation in Core-Stateless Networks,” *Proceedings of IEEE Infocom 2000*, pp. 421-430, March 2000.

[14] V. Paxson and and S. Floyd, “Wide-Area Traffic: The Failure of Poisson Modeling,” *IEEE/ACM Transactions on Networking*, Vol. 3, No. 3, pp. 226-244, June 1995.

[15] S. Sahu, D. Towsley and J. Kurose, ”A Quantitative Study of Differentiated Services for the Internet,” *Proceedings of IEEE Global Internet’99*, pp. 1808-I817, December 1999.

[16] N. Semret, R. Liao, A. Campbell and A. Lazar, “Peering and Provisioning of Differentiated Internet Services,” *Proceedings of INFOCOM 2000*, pp. 414-420, March 2000.

[17] D. Sleator and R. Tarjan, “Amortized Efficiency of List Update and Paging Rules,” *CACM 28*, pp. 202-208, 1985.

[18] I. Stoica and H. Zhang, “ Providing Guaranteed Services without Per Flow Management,” *Proceedings of SIGCOM 1999*, pp. 81-94.

[19] A. Veres and M. Boda, “The Chaotic Nature of TCP Congestion Control,” *Proceedings of INFOCOM 2000*, pp. 1715-1723, March 2000.