Improved Competitive Guarantees for QoS Buffering

Alex Kesselman[1], Yishay Mansour[1], and Rob van Stee[2,⋆]

[1] School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel.
{alx,mansour}@cs.tau.ac.il
[2] Centre for Mathematics and Computer Science (CWI), Kruislaan 413, NL-1098 SJ Amsterdam, The Netherlands. Rob.van.Stee@cwi.nl

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 FIFO buffering algorithms, where packets must be transmitted in the order they arrive. 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 arbitrary packet values that for the first time achieves a competitive ratio better than 2, namely $2 - \epsilon$ for a constant $\epsilon > 0$.

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 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

⋆ 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.

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. The algorithm 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 value of packets transmitted.

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 to 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. Since the buffer size is fixed, 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 and 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 $V_{\mathrm{ALG}}(\sigma)$.

Definition 1. *An online algorithm* ALG *is c-competitive iff for every input sequence* σ, $V_{\mathrm{OPT}}(\sigma) \leq c \cdot V_{\mathrm{ALG}}(\sigma)$.

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 algorithms 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 algorithms 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. The loss of an algorithm is analyzed 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. The problem of whether the competitive ratio of 2 of the natural greedy algorithm can be improved has been open for a long time. It this paper we solve it positively. Our model is identical to that of [7].

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.

2 Model Description

We consider a QoS buffering system that is able to hold B packets. 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 step a set of packets (possibly empty) arrives and at the end of time step a packet is scheduled if any. We denote by $\mathcal{A}(t)$ the set of packets arriving at time step t, by $\mathcal{Q}(t)$ the set of packets in the buffer after the arrival phase at time step t, and by $\textsc{alg}(t)$ the packet sent (or scheduled/served) at the end of time step t if any by an algorithm $\textsc{alg}$. At any time step t, $|\mathcal{Q}(t)| \leq B$ and $|\textsc{alg}(t)| \leq 1$, whereas $|\mathcal{A}(t)|$ can be arbitrarily large. We also denote by $\mathcal{Q}(t, \geq w)$ the subset of $\mathcal{Q}(t)$ of packets with value at least w.

As mentioned in the introduction, we consider FIFO buffers in this paper. Therefore, the packet transmitted at time t is always the first (oldest) packet in the buffer among the packets in $\mathcal{Q}(t)$.

3 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 β. For sufficiently large values 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 current throughput and minimizing potential future loss.

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

Lemma 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.*

```
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. Otherwise, drop (*reject*) the packet $p'$ that has minimal value among $p$ and the packets in
   the buffer. If $p' \neq p$, accept $p$ ($p$ *pushes out* $p'$).
```

Fig. 1. Algorithm PG.

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/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 $|X| \leq B$. 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/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*.

3.1 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 step 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 $Q(t)$ 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 $Q(t)$ 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 $Q(t') \setminus Q(t)$ 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 γ. Next we define the notion of overload interval more formally.

Definition 2. *An* α-overflow *takes place when a packet of value* α *is rejected, where* α *is said to be the* overload value.

Definition 3. *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.*

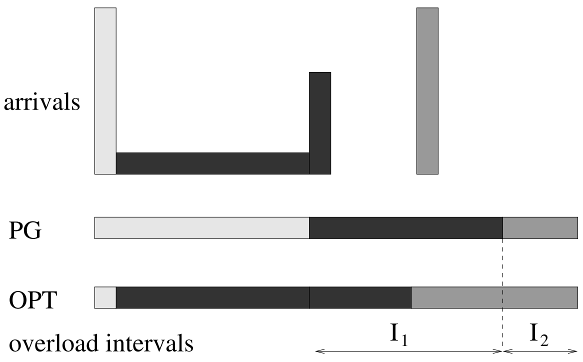

Fig. 2. An example of overload intervals. Light packets have value 1, dark packets value $\beta - \epsilon$, medium packets value 2. The arrival graph should be interpreted as follows: B packets of value 1 arrive at time 1, 1 packet of value $\beta - \epsilon$ arrives at times $2, \ldots, B - 1$, etc.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

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.

Definition 4. *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 5. *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 *A 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 packet except p' is preempted (e.g. p may push out p'' that preempts p').

1. Assign the value of each packet from PG $\cap$ 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.

Fig. 3. Main assignment routine.

Definition 6. *For an overload interval $\mathcal{I}$ let* BELONG$(\mathcal{I})$ *denote the set of packets that belong to $\mathcal{I}$. This set consists of three distinct subsets: scheduled packets (*PG$(\mathcal{I})$*), preempted packets (*PREEMPT$(\mathcal{I})$*) and rejected packets (*REJECT$(\mathcal{I})$*). Finally, denote by* REPLACE$(\mathcal{I})$ *the set of packets that replace packets from* PREEMPT$(\mathcal{I})$. *These packets are either in* 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 7. *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 $t_f^i = t_s^{i+1}$ 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 BELONG$(\mathcal{I}_i)$, PG$(\mathcal{I}_i)$, ... by BELONG$_i$, PG$_i$, ... 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.*

3.2 Analysis of the PG 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 OPT $\setminus$ 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 in Figure 3.

Before we describe the overload sequence assignment routine we need some definitions. Consider an overload sequence S. We introduce the following notation: OPT$_i$ = OPT $\cap$ BELONG$_i$, REJOPT$_i$ = OPT $\cap$ REJECT$_i$, PRMOPT$_i$ = OPT $\cap$ PREEMPT$_i$. We write PG$(S) = \cup_{i=1}^k$PG$_i$ and define analogously OPT(S), REJOPT(S), and PRMOPT(S).

Definition 8. *For $1 \leq i \leq k$,* OUT$_i$ *is the set of packets that have been replaced by packets outside S.*

Clearly, OUT$_i \subseteq$ 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 $\mathcal{Q}(t_a)\backslash\text{BELONG}(S)$ 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 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 9. *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.*

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 $\text{REJOPT}(S)$. 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 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 that OPT was serving other (presumably more valuable) packets.

1. For interval $\mathcal{I}_i$ s.t. $1 \le i \le k$, 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 this 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).

Fig. 4. Overload sequence assignment routine.

Lemma 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. By definition, $|\text{SMALL}(S)| = \max(|\text{UNASG}(S)| - B/2, 0)$. To avoid trivialities, assume that $|\text{UNASG}(S)| > B/2$ and let $x_i = |\text{UNASG}_i|$. By assumption (1)

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

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 $|\text{PREDOPT}_i| \ge x_i$ since the buffer of PG is full of packets from $\cup_{j=\min(i,m)}^{k} \text{BELONG}_j$ at time t_s^j. If it is not the case then we obtain that the schedule of OPT is infeasible using an argument similar to that of Lemma 2.

We also claim that $|\text{PREDOPT}_m| \ge \sum_{i=m}^{k} x_i$ 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 $v(p') \ge w_m$.

Let MAXUP_j be the set of the x_j most valuable packets from PREDOPT_j for $1 \le j < m$. 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 $\textsc{unasg}_j$ for $1 \le l \le |\textsc{unasg}_j|$. That is due to the fact that by Observation 2 the x_j least valuable packets from $\textsc{rejopt}_j$ are also the x_j least valuable packets from $\textsc{rejopt}_j \cup \textsc{pg}_j$.

Now for j starting from k and down to $m-1$, let $\textsc{maxdown}_j$ be the set containing x_j arbitrary packets from $\textsc{predopt}_m \setminus (\cup_{i=m+1}^{j-1} \textsc{maxdown}_i)$ with value at least w_m. (Recall that $\textsc{predopt}_m$ contains at least $\sum_{i=m+1}^{k} x_i$ packets with value greater than or equal to w_m.) Finally, let $\textsc{maxup}_m$ be the set of the x_m most valuable packets from $\textsc{predopt}_m \setminus (\cup_{i=m+1}^{k} \textsc{maxdown}_i)$. Clearly, any packet in $\textsc{maxdown}_j$ has greater value than any packet in $\textsc{reject}_j$ for $m+1 \le j \le k$. Similarly to the case of $j < m$, we obtain that the value of the l-th largest packet in $\textsc{maxup}_m$ is at least as large as that of the l-th largest packet in $\textsc{unasg}_m$ for $1 \le l \le |\textsc{unasg}_m|$.

Let $\textsc{maxp}(S) = (\cup_{i=1}^{m} \textsc{maxup}_i) \cup (\cup_{i=m+1}^{k} \textsc{maxdown}_i)$ and let t_i be the time at which $\textsc{opt}$ schedules the i-th packet from $\textsc{maxp}(S)$. We also denote by $\textsc{maxp}(S, t_i)$ the set of packets from $\textsc{maxp}(S)$ that arrived by time t_i. For $B/2+1 \le i \le |\textsc{unasg}(S)|$, let $\textsc{large}(t_i)$ be the set of $B/2$ largest packets in $\textsc{maxp}(S, t_i)$. We define

$$\textsc{extra}(S) = \cup_{i=B/2+1}^{|\text{UNASG}(S)|} \textsc{pg}(t_i).$$

That is, the set $\textsc{extra}(S)$ consists of the packets served by $\textsc{pg}$ while $\textsc{opt}$ was serving packets from the $\textsc{predopt}$ sets.

We show that at time t_i, $\textsc{pg}$ schedules a packet with value of at least w'/β, where w' is the minimal value among packets in $\textsc{large}(t_i)$. If all packets from $\textsc{large}(t_i)$ are present in the buffer at time t_i then we are done by Lemma 1. Note that the earliest packet from $\textsc{large}(t_i)$ arrived before or at time $t_i - B/2$ since $\textsc{opt}$ schedules all of them by time t_i. In case a packet p from $\textsc{large}(t_i)$ has been dropped, then by the definition of $\textsc{pg}$ and the construction of the intervals, $\textsc{pg}$ schedules at this time a packet that has value at least $v(p) > w'/\beta$.

Observe that the last packet from $\textsc{extra}(S)$ is sent earlier than t_s^m and therefore $\textsc{extra}(S) \cap \textsc{pg}_m = \emptyset$. It is easy to see that the set defined above satisfies the condition of the lemma. □

Theorem 1. *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 Lemma 2, we obtain that the number of unassigned packets is bounded from above by:

$$\begin{aligned} |\textsc{unasg}(S)| &= |\textsc{rejopt}(S)| + |\textsc{pg}(S) \cap \textsc{opt}(S)| - |\textsc{pg}(S)| - |\textsc{out}(S)| \\ &= |\textsc{opt}(S)| - |\textsc{prmopt}(S)| - |\textsc{pg}(S)| - |\textsc{out}(S)| \\ &\le B - |\textsc{prmopt}(S)| - |\textsc{prevp}(S)|. \end{aligned} \tag{1}$$

Observe that each packet p that replaces a packet p' with value w can be assigned a value of w if $p' \in \textsc{opt}$. 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.

Let $\textsc{asg}_1$ be the subset of $\textsc{pg}_m \cup \textsc{replace}_m$ containing the unavailable packets after the first two steps of the main assignment routine. By definition, every such packet directly replaced a packet from $\textsc{opt}$. We show that all packets directly replaced by packets from $\textsc{asg}_1$ belong to $\textsc{prmopt}(S) \cup \textsc{prevp}(S)$. Consider such a packet p. If p is directly preempted by a packet from $\textsc{asg}_1$ 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 $\textsc{asg}_1$. In this case, by the overload sequence construction, p' must belong to S, and therefore p belongs to $\textsc{prmopt}(S) \cup \textsc{prevp}(S)$. Thus, $|\textsc{asg}_1| \le |\textsc{prmopt}(S)| + |\textsc{prevp}(S)|$.

We denote by $\textsc{asg}_2$ the subset of $\textsc{pg}_m \cup \textsc{replace}_m$ containing packets that have been assigned some value at Step 3 of the sequence assignment routine. We have $|\textsc{asg}_2| = 2\max(|\textsc{unasg}(S)| - B/2, 0)$.

Finally, let $\textsc{asg}_3$ and $\textsc{asg}_4$ be the subsets of $\textsc{pg}_m \cup \textsc{replace}_m$ 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 $\textsc{unasg}(S)$, respectively. Then $|\textsc{asg}_3| = |\textsc{asg}_4| = |\textsc{unasg}(S)| - 2\max(|\textsc{unasg}(S)| - B/2, 0)$.

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

$$|\textsc{asg}_1| + |\textsc{asg}_2| + |\textsc{asg}_3| \le B$$

while Observation 3 implies that $|\textsc{pg}_m \cup \textsc{replace}_m| \ge B$. Finally,

$$|\textsc{asg}_4| \le B - |\textsc{asg}_2| - |\textsc{asg}_3|,$$

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

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

For the proof, and the calculation of $\epsilon(\beta)$, we refer to the full paper. Optimizing the value of β, we get that for $\beta = 15$ the competitive ratio of $\textsc{pg}$ is close to 1.983, that is $\epsilon(\beta) \approx 0.017$.

Now let us go back to the assumption (1), that is $x_i = |\textsc{rejopt}_i| - (|\textsc{pg}_i \setminus \textsc{opt}_i| + |\textsc{out}_i|) \ge 0$. We argue that there exist two indices $l \le m$ and $r \ge m$ s.t. $x_i \ge 0$ for $l \le i \le r$ and $x_i \le 0$ for $1 \le i < l$ or $l < i \le k$. In this case we can restrict our analysis to the subsequence of S containing the intervals $\mathcal{I}_l, \ldots, \mathcal{I}_r$.

For a contradiction, assume that there exist two indices i, j s.t. $i < j \le m$ or $i > j \ge m$, $x_i > 0$ and $x_j < 0$. Then there are a packet $p \in \textsc{opt}_i$ and a packet $p' \in \textsc{pg}_j \setminus \textsc{opt}_j$ s.t. $v(p') > v(p)$. We obtain that the schedule of $\textsc{opt}$ can be improved by switching p and p'.

It remains to consider the assumption (2), that is no packet from $\textsc{extra}(S)$ belongs to another overload sequence S'. In this case we improve the bound of Lemma 2 applied to both sequences.

Lemma 4. *For any two consecutive overload sequences* S' *and* S *the following holds:* $|\textsc{opt}(S)| + |\textsc{opt}(S')| - |\textsc{pg}(S)| - |\textsc{pg}(S')| \le 2B + |\textsc{out}(S)| - |\textsc{prevp}(S)| - |\textsc{prevp}(S')| - |\textsc{extra}(S) \cap \textsc{belong}(S')|$.

Proof. According to the proof of Lemma 2, $t_f^m - t_l \geq B - |\textsc{out}(S)|$ where t_l is the last time during S at which a packet from $\textsc{belong}(S)$ has been rejected. Let $z = |\textsc{extra}(S) \cap \textsc{belong}(S')|$. We argue that OPT has scheduled at most $t_l + 2B - {t'}_s^1 - |\textsc{prevp}(S')|$ packets from $\textsc{belong}(S) \cup \textsc{belong}(S')$. That is due to the fact that the earliest packet from $\textsc{belong}(S')$ arrived at or after time ${t'}_s^1 - B + |\textsc{prevp}(S')|$. Observe that between time ${t'}_s^1$ and time t_f^k at most $B - z - |\textsc{prevp}(S)|$ packets outside of $\textsc{belong}(S) \cup \textsc{belong}(S')$ have been scheduled by PG. Hence, PG has scheduled at least $t_l + z + |\textsc{prevp}(S)| - {t'}_s^1 - |\textsc{out}(S)|$ packets from $\textsc{belong}(S) \cup \textsc{belong}(S')$, which yields the lemma. □

Using Lemma 4, we can extend our analysis to any number of consecutive overload sequences without affecting the resulting ratio.

3.3 Lower Bounds

Theorem 3. *The* PG *algorithm has a competitive ratio of at least ϕ.*

We omit the proof due to space constraints.

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 4. *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 $2, \ldots, l_1$, 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 $i = 1, \ldots, n-1$ and define $s_i = \sum_{j=1}^{i}(l_i v^{i-1})/B$ for $i = 1, \ldots, n$.

If the sequence stops during phase $i < n$, then ALG earns $l_1 + l_2 v + l_3 v^2 + \ldots + l_i v^{i-1} + l_i v^i = B \cdot s_i + l_i v^i$ while OPT can earn at least $l_1 v + l_2 v^2 + \ldots + (l_{i-1} + 1)v^{i-1} + l_i v^i = B(v \cdot s_i + v^{i-1})$. The implied competitive ratio is $(v \cdot s_i + v^{i-1})/(s_i + l_i v^i/B)$. 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 $l_i v^i = (s_i - s_{i-1})/v$. We find

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

It can be seen that $s_i/v^i \to 1/(v^2(\mathcal{R}-1))$ 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 $l_1 + l_2 v + l_3 v^2 + \ldots + l_n v^{n-1} +$

$B \cdot v^n = B \cdot (s_n + v^n)$ whereas OPT can earn at least $l_1 v + l_2 v^2 + \ldots + l_n v^n + B \cdot v^n = B(v \cdot s_n + v^n)$. The implied ratio is

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

□

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*, July 1999.
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," *Journal of Algorithms*, Vol. 46, Issue 1, pp 79–95, January 2003.
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," *Proceedings 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.