

Further Improvements in Competitive Guarantees for QoS Buffering

Nikhil Bansal¹, Lisa K Fleischer¹, Tracy Kimbrel¹, Mohammad Mahdian²,
Baruch Schieber¹, and Maxim Sviridenko¹

¹ IBM Watson Research Center, Yorktown Heights, NY 10598.
{nikhil,lkf,kimbrel,sbar,sviri}@us.ibm.com

² Department of Mathematics, MIT, Cambridge MA 02139.
mahdian@theory.lcs.mit.edu

Abstract. We study the behavior of algorithms for buffering packets weighted by different levels of Quality of Service (QoS) guarantees in a single queue. Buffer space is limited, and packet loss occurs when the buffer overflows. We describe a modification of the previously proposed “preemptive greedy” algorithm of for buffer management and give an analysis to show that this algorithm achieves a competitive ratio of at most 1.75. This improves upon recent work showing a 1.98 competitive ratio, and a previous result that shows a simple greedy algorithm has a competitive ratio of 2.

1 Introduction

Quality of Service guarantees for network service allow service providers to address the service requirements of their customers by providing different levels of service. In the network setting where traffic volumes may exceed network capacity, effective management of packets at network buffers is key to achieving QoS guarantees. By differentiating service levels, packets of different types of customers may be treated according to the level of service they require. The importance of this issue is reflected in the interest devoted to it in the networking community [9, 12, 13, 22, 21, 23, 24].

We consider the problem of buffer management for a single queue with a limited capacity buffer. Each time step, many packets may enter the buffer, some packets may be dropped, and the packet at the head of the queue is *delivered*. Packets are delivered on a FIFO basis: once packets enter the buffer, they are not reordered. We abstract the differentiated service model by attributing different values to different packets according to their service level. The goal is to deliver the set of packets with highest total value.

We study the behavior of FIFO queues, since FIFO helps to both ensure a level of fairness and prevent packets from timing out, but more importantly, FIFO ensures that packets arrive at their destination in the order they were transmitted. In video streaming, packets from the same source may belong to different service levels according to the type of information they contain [19]. FIFO queues in a QoS environment ensure that these packets arrive in order.

Our Contribution. In this paper, we use competitive analysis to show that a modification of the preemptive greedy algorithm of Kesselman et al. [15] achieves a competitive ratio of 1.75. The algorithm of Kesselman et al. [15] is the first algorithm to break the bound of 2. Their improvement to 1.98 is small, but the algorithmic framework is important, and we believe it is a good approach for achieving better ratios.

Packets are dropped for two reasons:

1. *Evicted* packets are packets (either already in the buffer, or just arrived) that are dropped because the buffer is too full. In this case, the minimum value packet is dropped.
2. For a given parameter $\beta > 1$, when a packet of value v arrives at the buffer, if there is a packet with value less than v/β in the buffer, then one such packet is dropped. This is a *preempted* packet. Kesselman et al. [15] propose to preempt the first such packet in FIFO order. We drop the first such packet that is a local minimum in FIFO order: the packet our algorithm drops has value strictly smaller than the value of the packet following it in the FIFO queue. To obtain the guarantee of 1.75, we use $\beta = 4$.

Previous work. Prior to [15], the best known competitive ratio for the single queue problem was 2 which was obtained by the greedy algorithm that drops minimum value packets only when the buffer is full [16]. If the decision to drop a packet can be made only when the packet arrives, and once a packet is accepted into the buffer it must be delivered, then the best possible competitive ratio is $\Theta(\log \alpha)$ where α is the ratio of the maximum valued packet to the minimum [2, 4].

Many other models and approaches to studying this problem have been proposed. A recent paper of Albers and Schmidt looks at the unweighted case for a multiqueue switch [3]. Using competitive analysis, other papers study the case with just two weights [2, 18], unlimited capacity buffers [8, 17], multiqueue switches [3, 6], and multiple-node networks [1, 7]. Other approaches to studying this problem include probabilistic models of packet injection [10, 20] and adversarial queueing theory [5, 11].

Recently, Azar and Richter [7] define a class of algorithms for which guarantees for the case when all packets have values in $\{0, 1\}$ can be extended to arbitrary values. The class of algorithms they define considers only the relative order of values of packets. Our algorithm does not fall into this class since it behaves differently when given packets of value 1 and $1 + \epsilon$ than when given packets of value 1 and $v \geq \beta + \epsilon$ for a small positive ϵ . However, Lemma 1 in our paper can be considered a generalization of the zero-one principle of Azar and Richter [7].

2 Model Description

In this section, we give a formal description of the model considered in this paper. Our model is the same as the FIFO model studied in [15] and equivalent to the one considered by [16].³

In our setting a switch may deliver one packet per unit time. Packets might arrive at any time, and only one packet can be delivered at the end of each time slot. There is a buffer that can be used to store B packets. The buffer is FIFO, i.e., the delivered packets need to follow the arrival order. Due to the bounded size of the buffer, sometimes packets must be dropped. A buffer management algorithm must decide at each step which of the packets to drop, which to keep in the buffer, and which to deliver, satisfying some constraints specified below. Fix a buffer management algorithm A . At the *end* of each time slot t , there is a set of packets $Q_A(t)$ such that $|Q_A(t)| \leq B$ stored in the buffer; we define $Q_A(0) = \emptyset$. Each packet p has a positive real value denoted $v(p)$. At time t , a single packet from $Q_A(t)$, denoted $A(t)$, is delivered. The algorithm is also allowed to not deliver any packet at time t , in which case we let $A(t) = \emptyset$, and define $v(\emptyset) = 0$. During the next time slot (time slot $t + 1$), a set of packets $\text{Arr}(t + 1)$ arrives. A subset of $Q_A(t) \cup \text{Arr}(t + 1) \setminus \{A(t)\}$, denoted $\text{Dr}_A(t)$, is *dropped*, and the rest of packets are kept in the buffer. The set of packets in the buffer at the end of time slot $t + 1$ is $Q_A(t + 1) = Q_A(t) \cup \text{Arr}(t + 1) \setminus (\text{Dr}_A(t) \cup \{A(t)\})$. Notice that we allow the buffer management algorithm to drop a packet that was placed in the buffer during an earlier time slot.

We consider only instances that are finite, i.e., those in which there is a time T such that no new packet arrives after time $T - B$, and therefore we can assume without loss of generality that the algorithm will not deliver any packet after time T . The *value delivered* by the algorithm is the sum of the values of all packets delivered by the algorithm. For a set P of packets define $v(P)$ to be the total value of the packets in the set. Also, we denote the set of packets that an algorithm A delivers by $S_A = \{A(t) : t = 0, \dots, T\}$. In this notation the value delivered by algorithm A is $v(S_A)$.

Our goal is to compare the ratio of the value of packets delivered by our online algorithm to the value of packets delivered by an optimal clairvoyant algorithm (i.e., an offline algorithm that sees the complete input sequence in advance, and using this information delivers a set of packets with maximum value), denoted OPT. An online algorithm A is called c -competitive, if for every instance, $v(S_{\text{OPT}})$ is at most c times $v(S_A)$.

3 Algorithm

We analyze a modification which we call PG of the *preemptive greedy algorithm* of Kesselman et al. [15] with parameter $\beta > 1$. This algorithm is shown in Figure 1.

³ the parameter B in their model is equivalent to $B + 1$ in ours.

The Modified β -Preemptive Greedy (PG) Algorithm.

When a packet p of value $v(p)$ arrives, do the following:

1. Find the first packet p' in FIFO order such that p' has value less than $v(p)/\beta$ and less than the value of the packet following p' in the buffer (if any). If such a packet exists, preempt it.
2. If there is free space in the buffer, accept p
3. Otherwise (the buffer is full), then evict a packet p' with the smallest value among $\{p\}$ and those in the buffer, and keep the rest in the buffer.

Fig. 1. The algorithm PG

Every time a new packet p arrives, our algorithm determines whether there exists a packet p' in the buffer such that $v(p') < v(p)/\beta$. If so, the algorithm finds the first such packet p' in the buffer. If the next packet in the buffer has value no greater than the value of p' , we let p' be the next packet and iterate, until we find the first packet p' that has value less than $v(p)/\beta$, and less than the value of the next packet in the buffer.¹ We drop p' from the buffer, and say that p' is *preempted* by p . Next, the algorithm determines whether there is free space for p in the buffer, either because some p' was preempted or because there was free space to begin with. If so, p is added to the tail of the buffer; otherwise, a packet of smallest value among $\{p\}$ and those in the buffer is *evicted*, and the rest are kept in the buffer. At the end of each time slot, the algorithm delivers the packet at the head of the buffer (if such a packet exists).

Kesselman et al. [15] give an example that shows that the competitive ratio of their algorithm is at least $\max\{2 - 1/\beta, \beta/(\beta - 1)\}$. This example yields the same lower bound on the competitive ratio of our algorithm. The main result of our paper is to show that this lower bound is tight for $\beta \geq 4$.

Theorem 1. *For every $\beta \geq 4$, the algorithm in Figure 1 has a competitive ratio of $2 - 1/\beta$.*

Setting $\beta = 4$, the above theorem shows that the algorithm PG is a 1.75-competitive algorithm for the buffer management problem.

4 Analysis of the Algorithm

In this section, we prove Theorem 1. The proof is composed of three steps: First, we use the theory of linear programming (specifically, the fact that every linear program has a basic feasible solution) to limit the set of possible instances we need to consider to find the competitive ratio of our algorithm. Using this, we

¹ The original algorithm of Kesselman et al. [15] always preempts the first packet in the buffer of value less than $v(p)/\beta$, without comparing it to the value of the next packet in the buffer.

can focus on one such instance and describe its structure in terms of several parameters, and also compute the value delivered by our algorithm and the optimal algorithm on this instance in terms of these parameters. In the second step, we prove several inequalities relating these parameters. This allows us to bound the competitive ratio of our algorithm by solving a maximization linear program which, following the terminology of [14], we call a *factor-revealing LP*. The third and final step is to prove an upper bound on the value of an optimal solution of this linear program. By LP duality, this step is equivalent to finding a feasible solution to the dual of the factor-revealing LP.

4.1 Structure of the worst-case examples

In this section we show that it is enough to analyze the performance of our algorithm on instances in which the value of each packet is either 0 or β^i for some $i \geq 0$, but ties are allowed to be broken by an adversary. More precisely, we have the following lemma.

Lemma 1. *Fix an instance size T . The worst case competitive ratio of the modified preemptive greedy algorithm over instances of size at most T is realized by an input sequence in which the value of each packet is zero or a power of β , and an adversary determines for every two packets p and p' and number γ , where $v(p) = \gamma v(p')$, whether the inequality $v(p) \geq \gamma v(p')$ is “true” or “false”.*

Proof. Consider an arbitrary instance \mathcal{R} of size T . For every two packets p and p' , write all inequalities of the form “ $v(p) ? v(p')$ ” or “ $v(p) ? \beta v(p')$ ” (where $?$ is either $>$, $=$, or $<$) that are true in the instance \mathcal{R} . Let \mathcal{C} denote the collection of these inequalities. Now, observe that PG considers the value of a packet only to compare it to either the value of another packet, β times the value of another packet, or the value of another packet divided by β . Therefore, on every instance that differs from \mathcal{R} in the values of the packets, but is the same as \mathcal{R} in the arrival times and satisfies the inequalities in \mathcal{C} , PG will behave exactly as it does on \mathcal{R} , and hence the subsequence of packets delivered will be the same. Next, we can write a linear program with a set of variables corresponding to the set $\{v(p)\}$ of packet values. This LP consists of the inequalities in \mathcal{C} with equalities added to strict inequalities (i.e., $<$ changed to \leq and $>$ changed to \geq) and an extra normalization inequality $\sum_{p \in S_{\text{PG}}(\mathcal{R})} v(p) = 1$, and the objective is to maximize $\sum_{p \in S_{\text{OPT}}(\mathcal{R})} v(p)$. Take an optimal basic feasible solution of this LP, and construct an instance \mathcal{R}' which is the same as \mathcal{R} , except that values of the packets are replaced by those of the basic feasible solution of the LP. We also let the adversary break the ties in \mathcal{R}' in the same direction as in \mathcal{R} . It is clear that the algorithm PG delivers the same subsequence of packets in \mathcal{R}' as in \mathcal{R} , and therefore, the competitive ratio of PG on \mathcal{R}' is at least as great as on \mathcal{R} . Let \mathcal{L} denote the collection of linear programs that consist of inequalities of the form “ $v(p) ? v(p')$ ” or “ $v(p) ? \beta v(p')$ ” (where $?$ is either \leq or \geq), and $\sum_{p \in S_1} v(p) = 1$ for a set $S_1 \subseteq \{1, \dots, T\}$, and let \mathcal{A} denote the set of instances that are basic feasible solutions of linear programs in \mathcal{L} . By the above argument,

the competitive ratio of PG is at most its competitive ratio on instances in \mathcal{A} , if an adversary is allowed to break the ties. Since the number of instances in \mathcal{A} and the number of ways to break ties is finite, it follows that there is an instance in \mathcal{A} that achieves the worst ratio. \square

From now on we only consider input sequences in which each packet value is either zero or a power of β . In the rest of this section, we define a hierarchical structure of time intervals and several parameters that describe the behavior of our algorithm and of OPT on a given instance.

Intervals. An *interval of type i* is a time interval I such that at every step $t \in I$ the algorithm PG delivers a packet of value at least β^i , and I is a maximal interval with this property; i.e., in the time step immediately before the beginning and in the step immediately after the end of I the algorithm delivers either nothing, or a packet of value less than β^i . Let \mathcal{I}_i denote the set of maximal intervals of type i , and let \mathcal{I} be the union over all i of \mathcal{I}_i . Since the same interval can be in \mathcal{I}_i for more than one value of i , we consider \mathcal{I} as a multiset.

The *last eviction* in an interval I of type i is the latest time step t in I such that at time t a packet of value *at least* β^i is evicted from the buffer. If no such time step exists, we say that I has *no evictions*. (Note that there may be an eviction within I of a packet of value $\beta^{i'} < \beta^i$ after the last eviction of I ; we will consider this to belong to the interval of type i' containing I and those containing it, but not to I .) Let \mathcal{E} be the set of intervals that have evictions, and let \mathcal{N} be the set of intervals that have no evictions.

The interval structure. We define an *interval structure* as a sequence of ordered rooted trees, with a partition of its vertex set into two subsets \mathcal{E} and \mathcal{N} . The intervals in \mathcal{I} can be represented using such a structure (denoted by \mathcal{T}) as follows. The root of each tree is an interval in \mathcal{I}_0 , and the children of each interval $I \in \mathcal{I}_i$ are the maximal intervals of type $i+1$ that are contained in I . These children are ordered by increasing time from left to right (e.g., the leftmost child corresponds to the interval that ends before the others begin). The trees are also ordered by increasing time. A vertex is in \mathcal{E} if the corresponding interval has an eviction; otherwise, it is in \mathcal{N} . We denote the set of children of I by $\text{child}(I)$ and the parent of I by $\text{par}(I)$. Also, we call the leftmost (rightmost, respectively) child of I that is in \mathcal{E} the first (last, respectively) child of I , and denote it by $\text{lastch}(I)$ ($\text{firstch}(I)$, respectively). Notice that, despite the name, I can have other children (that have no evictions) to the left of $\text{firstch}(I)$ or to the right of $\text{lastch}(I)$. The set of nodes that are the last children of their parents is denoted by LC . Also, we let $\text{rsib}(I)$ denote the set of siblings of I that are to the right of I and are in \mathcal{E} .

In the next section, we will show that for every interval structure \mathcal{T} , the competitive ratio of the algorithm PG on instances whose corresponding interval structure is \mathcal{T} can be bounded by the solution of a linear program indexed by \mathcal{T} .

Packets assigned to each interval. Consider an interval I of type i . We assign all packets of value at least β^i delivered or evicted during I to this interval. For a

preempted packet p , we define $\text{next}(p)$ as the first packet after p in arrival order that is delivered by PG. We assign p to I if p has value at least β^i and $\text{next}(p)$ is assigned to I . To simplify the notation, we let $\text{next}(p) = p$ for a packet p that is delivered or evicted by PG. Let P_I denote the set of packets assigned to an interval I .

Lemma 2. *For each packet p of value at least β^i there is a unique interval I of type i such that p is assigned to I .*

Proof. The statement is trivial if p is delivered by PG. If p is evicted by PG, then the packet delivered at the same time step must have a value at least that of p , and hence p must be assigned to a unique interval of type i .

Now suppose that p is preempted. In order to show that there is a unique interval of type i that p is assigned to, it is enough to show that $\text{next}(p)$ is of value at least β^i . Consider the sequence $p = p_1, p_2, \dots, p_j = \text{next}(p)$ where for each $1 \leq k < j$, p_k is followed immediately in the buffer by p_{k+1} at the time p_k is evicted or preempted. In each case, whether an eviction or a preemption, $v(p_k) \leq v(p_{k+1})$, and the lemma follows. \square

Lemma 3. *Let I be an interval of type i with evictions, and let $t \in I$ be a time at which a packet of value at least β^i is evicted. Then every packet that is in the buffer at time t is assigned to I .*

Proof. Assume for the sake of contradiction that there is a packet p in the buffer at time t that is not assigned to I . Therefore, p must not be evicted at time t , nor should it be delivered in the interval I . All packets in the buffer at time t have value at least β^i ; thus p cannot be delivered in an interval after I . Therefore p must be dropped some time after t . Let $p_1 = \text{next}(p)$, and p_2 be the packet delivered at the time step after the end of I . By definition, $v(p_2) < \beta^i$, so p_2 cannot be present in the buffer at time t . Therefore p_2 arrives after p . This means that p_1 must arrive before p_2 , for otherwise p_1 would not be the first packet after p that is delivered. But then p_1 must be delivered earlier than p_2 , and hence $p_1 \in P_I$. Thus, $p \in P_I$, a contradiction. \square

We are now ready to define several parameters for each interval I . These definitions are presented in Table 1. In the next section, we will prove inequalities relating these parameters, and then we will treat them as variables in a linear program. The following lemma states the value delivered by PG and OPT in terms of these parameters. Here for every interval I of type $i > 0$, $b(I)$ denotes $(\beta - 1)\beta^{i-1}$. If I is an interval of type 0, then $b(I) = 1$.

Lemma 4. *We have $v(S_{\text{PG}}) = \sum_{I \in \mathcal{I}} b(I)A(I)$ and $v(S_{\text{OPT}}) = \sum_{I \in \mathcal{I}} b(I)X(I)$.*

Proof. Let I be an interval of type i . By Lemma 2 and the definition of $A(I)$, the number of packets of value exactly β^i in $P_I \cap S_{\text{PG}}$ is $A(I) - \sum_{J \in \text{child}(I)} A(J)$.

Param	Definition
$A(I)$	$ P_I \cap S_{\text{PG}} $, or equivalently, the length of I
$X(I)$	$ P_I \cap S_{\text{OPT}} $
$A^e(I)$	$I \in \mathcal{E}$: # of packets in P_I delivered by PG after the last eviction of I
$R^+(I)$	# of packets in P_I that are preempted by packets not in P_I
$R^-(I)$	$I \in \mathcal{N}$: # of packets in P_I that preempt packets not in P_I $I \in \mathcal{E}$: # of packets in P_I that preempt packets in $\{p : \text{next}(p) \notin P_I\}$
$R^d(I)$	$I \in \mathcal{N}$: 0 $I \in \mathcal{E}$: # of packets in P_I that preempt packets not in P_I after the last eviction of I
$R^e(I)$	$I \in \mathcal{N}$: # of packets in P_I that are preempted by packets in P_I $I \in \mathcal{E}$: # of packets in P_I that are preempted by other packets in P_I after the last eviction in I

Table 1. Parameters associated with an interval

Therefore, the total value delivered by PG is

$$\begin{aligned}
\sum_i \sum_{I \in \mathcal{I}_i} \beta^i (A(I) - \sum_{J \in \text{child}(I)} A(J)) &= \sum_i \sum_{I \in \mathcal{I}_i} \beta^i A(I) - \sum_i \sum_{I \in \mathcal{I}_{i+1}} \beta^i A(I) \\
&= \sum_{I \in \mathcal{I}} b(I) A(I).
\end{aligned}$$

A similar argument yields the expression for $v(S_{\text{OPT}})$. □

4.2 Inequalities

In the following sequence of lemmas, we prove several inequalities between the parameters defined in the previous section.

Lemma 5. *For every $I \in \mathcal{E}$, $A^e(I) + R^+(I) \geq B + R^d(I)$.*

Proof. Consider the set of packets in P_I that are either in the buffer at the time of the last eviction (by Lemma 3 we know that all these packets are in P_I), or arrive after the last eviction. We construct a graph on this set of packets by forming an edge from a packet p_1 to a packet p_2 if p_1 preempts p_2 . By the definition of preemption, each vertex in this graph has in-degree and out-degree at most one, and so, this graph is a union of disjoint paths. Therefore, the number of vertices of in-degree zero is equal to the number of vertices of out-degree zero in this graph. Now, notice that the vertices corresponding to the B packets that were present in the buffer at the time of the last eviction, and also the vertices corresponding to the $R^d(I)$ packets that preempt packets outside P_I have outdegree zero. Therefore, the number of vertices of outdegree zero is at least $B + R^d(I)$. On the other hand, every vertex of indegree zero corresponds to a packet in P_I that is either not preempted (and therefore is delivered), or is

preempted by packets outside P_I . Thus, the number of vertices of indegree zero is at most $A^e(I) + R^+(I)$. Hence, $A^e(I) + R^+(I) \geq B + R^d(I)$. \square

Lemma 6. *For every interval $I \in \mathcal{N}$, $R^-(I) \leq A(I) + R^+(I)$.*

Proof. We use the same argument as in the proof of Lemma 5, except here we consider the graph of preemptions on the set of all packets in P_I . Since I has no evictions, the number of vertices of indegree zero is equal to $A(I) + R^+(I)$, and the number of vertices of outdegree zero is at least $R^-(I)$. \square

Lemma 7. *For every $I \in \mathcal{N}$, $X(I) \leq A(I) + R^e(I) + R^+(I)$.*

Proof. Since no packet in P_I is evicted, every packet must either be delivered, preempted by another packet in P_I , or preempted by a packet outside P_I . The number of such packets is $A(I)$, $R^e(I)$, and $R^+(I)$, respectively. \square

Due to space constraint, the proofs of the following lemmas are left to the full version of the paper.

Lemma 8. *For every $I \in \mathcal{E}$, let y_I denote the number of packets in P_I that OPT delivers before the start of I . Then, $X(I) \leq A(I) + R^e(I) + R^+(I) + y_I$.*

Lemma 9. *For every $I \in \mathcal{E}$, $X(I) \leq A(I) + B - R^-(I) + R^e(I) + R^+(I)$.*

Lemma 10. *For every interval $I \in \mathcal{E}$ of type $i > 0$,*

$$X(I) \leq \frac{1}{2} \left(A(I) + A(\text{par}(I)) - \sum_{J \in \text{sib}(I)} A(J) + B - A^e(\text{par}(I)) + A^e(\text{lastch}(\text{par}(I))) \right) + R^e(I) + R^+(I).$$

Lemma 11. *For every interval I of type 0, we have $X(I) \leq A(I) + \frac{B}{2} + R^e(I) + R^+(I)$.*

Lemma 12. *For every $i \geq 0$, $\sum_{I \in \mathcal{I}_i} (R^e(I) + R^+(I)) \leq \sum_{I \in \mathcal{I}_{i+1}} (R^e(I) + R^-(I) + R^d(I))$.*

The following theorem summarizes the results of this section.

Theorem 2. *Let \mathcal{T} be an interval structure, and $z_{\mathcal{T}}$ denote the solution of the following maximization program (which we call a factor-revealing LP) with variables OPT, ON, B , $A(I)$, $X(I)$, $A^e(I)$, $R^+(I)$, $R^-(I)$, $R^e(I)$, and $R^d(I)$ for every $I \in \mathcal{I}$.*

$$\text{maximize } \frac{\text{OPT}}{\text{ON}}$$

$$\text{subject to } \text{ON} = \sum_{I \in \mathcal{I}} b(I)A(I) \quad (1)$$

$$\text{OPT} = \sum_{I \in \mathcal{I}} b(I)X(I) \quad (2)$$

$$\forall I \in \mathcal{E} : A^e(I) + R^+(I) \geq B + R^d(I) \quad (3)$$

$$\forall I \in \mathcal{N} : R^-(I) \leq A(I) + R^+(I) \quad (4)$$

$$\forall I \in \mathcal{N} : X(I) \leq A(I) + R^e(I) + R^+(I) \quad (5)$$

$$\forall I \in \mathcal{E} : X(I) \leq A(I) + B - R^-(I) + R^e(I) + R^+(I) \quad (6)$$

$$\forall I \in \mathcal{E} \setminus \mathcal{I}_0 : X(I) \leq \frac{1}{2} \left(A(I) + A(\text{par}(I)) - \sum_{J \in \text{sib}(I)} A(J) + B - A^e(\text{par}(I)) + A^e(\text{lastch}(\text{par}(I))) \right) + R^e(I) + R^+(I) \quad (7)$$

$$\forall I \in \mathcal{I}_0 : X(I) \leq A(I) + \frac{B}{2} + R^e(I) + R^+(I) \quad (8)$$

$$\forall i : \sum_{I \in \mathcal{I}_i} (R^e(I) + R^+(I)) \leq \sum_{I \in \mathcal{I}_{i+1}} (R^e(I) + R^-(I) + R^d(I)) \quad (9)$$

$$\forall I \in \mathcal{I} : A^e(I) \leq A(I) \quad (10)$$

$$\forall I \in \mathcal{I} : A(I), X(I), A^e(I), R^+(I), R^-(I), R^e(I), R^d(I) \geq 0 \quad (11)$$

Then on every instance \mathcal{R} whose corresponding interval structure is \mathcal{T} , the ratio of the value delivered by the optimal solution to the value delivered by the algorithm PG is at most $z_{\mathcal{T}}$.

4.3 Analysis of the factor-revealing LP

The last step in the proof is to analyze the maximization program of Theorem 2, and prove that for every \mathcal{T} and $\beta \geq 4$, the solution of this program is at most $2 - 1/\beta$. This program is equivalent to a linear program. Therefore, by LP duality, this step can be done by multiplying each inequality with an appropriate multiplier, and adding them up. Due to space constraint, this analysis is left to the full version of the paper.

References

1. W. Aiello, R. Ostrovsky, E. Kushilevitz, and A. Rosen. Dynamic routing on networks with fixed size buffers. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 771–780, 2003.
2. William Aiello, Yishay Mansour, S. Rajagopalan, and Adi Rosen. Competitive queue policies for differentiated services. In *Proceedings of the IEEE INFOCOM*, pages 431–440, 2000.
3. S. Albers and M. Schmidt. On the performance of greedy algorithms in packet buffering. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, 2004.
4. N. Andelman, Y. Mansour, and A. Zhu. Competitive queueing policies for QoS switches. In *Proc. 14th ACM-SIAM Symp. on Discrete Algorithms*, pages 761–770, 2003.
5. Andrews, Awerbuch, Fernandez, Kleinberg, Leighton, and Liu. Universal stability results for greedy contention-resolution protocols. In *37th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 380–389, 1996.
6. Y. Azar and Y. Richter. Management of multi-queue switches in QoS networks. In *35th ACM Symposium on Theory of Computing*, pages 82–89, 2003.
7. Y. Azar and Y. Richter. The zero-one principle for switching networks. In *Proc. 34th ACM Symposium on Theory of Computing*, 2004.
8. Amotz Bar-Noy, Ari Freund, Shimon Landa, and Joseph (Seffi) Naor. Competitive on-line switching policies. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2002.
9. Y. Bernet, A. Smith, S. Blake, and D. Grossman. A conceptual model for diffserv routers. Internet draft, March 2000.
10. Alexander Birman, H. Richard Gail, Sidney L. Hantler, Zvi Rosberg, and Moshe Sidi. An optimal service policy for buffer systems. *Journal of the ACM*, 42(3):641–657, 1995.
11. Allan Borodin, Jon Kleinberg, Prabhakar Raghavan, Madhu Sudan, and David P. Williamson. Adversarial queuing theory. *Journal of the ACM*, 48(1):13–38, 2001.
12. D. Clark and J. Wroclawski. An approach to service allocation in the Internet. Internet draft, July 1997.
13. Constantinos Dovrolis, Dimitrios Stiliadis, and Parameswaran Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. In *SIGCOMM*, pages 109–120, 1999.
14. K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problem. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, 2002.
15. A. Kesselman, Y. Mansour, and R. van Stee. Improved competitive guarantees for QoS buffering. In *Proc. 11th Annual European Symposium on Algorithms (ESA)*, pages 361–372, 2003.
16. Alexander Kesselman, Zvi Lotker, Yishay Mansour, Boaz Patt-Shamir, Baruch Schieber, and Maxim Sviridenko. Buffer overflow management in QoS switches. In *ACM Symposium on Theory of Computing*, pages 520–529, 2001.
17. H. Koga. Balanced scheduling towards loss-free packet queueing and delay fairness. In *Proc. 12th Annual International Symposium on Algorithms and Computation*, pages 61–73, 2001.
18. Z. Lotker and B. Patt-Shamir. Nearly optimal fifo buffer management for Diff-Serv. In *Proc. 21st ACM-SIAM Symposium on Principles of Distributed Computing (PODC)*, pages 134–142, 2002.

19. Y. Mansour, B. Patt-Shamir, and O. Lapid. Optimal smoothing schedules for real-time streams. In *Proc. 19th ACM Symp. on Principles of Distributed Computing*, pp. 21–29, 2000.
20. Martin May, Jean-Chrysostome Bolot, Alain Jean-Marie, and Christophe Diot. Simple performance models of differentiated services schemes for the internet. In *Proc. IEEE INFOCOM*, pages 1385–1394, 1999.
21. T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. Delay differentiation and adaptation in core stateless networks. In *Proc. IEEE INFOCOM*, 2000.
22. K. Nichols, V. Jacobson, and L. Zhang. A twobit differentiated services architecture for the internet. Internet draft, 1997.
23. Nemo Semret, Raymond R.-F. Liao, Andrew T. Campbell, and Aurel A. Lazar. Peering and provisioning of differentiated internet services. In *Proc. IEEE INFOCOM*, pages 414–420, 2000.
24. Ion Stoica and Hui Zhang. Providing guaranteed services without per flow management. In *Proc. ACM SIGCOMM*, pages 81–94, 1999.