Management of Multi-Queue Switches In QoS Networks

Yossi Azar *   Yossi Richter †

Abstract

The concept of Quality of Service (QoS) networks has gained growing attention recently, as the traffic volume in the Internet constantly increases, and QoS guarantees are essential to ensure proper operation of most communication based applications. A QoS switch serves m incoming queues by transmitting packets arriving at these queues through one output port, one packet per time unit. Each packet is marked with a value indicating its guaranteed quality of service. Since the queues have bounded capacity and the rate of arriving packets can be much higher than the transmission rate, packets can be lost due to insufficient queue space. The goal is to maximize the total value of transmitted packets. This problem encapsulates two dependent questions: admission control, namely which packets to discard in case of queue overflow, and scheduling, i.e. which queue to use for transmission in each time unit. We use competitive analysis to study online switch performance in QoS based networks. Specifically, we provide a novel generic technique that decouples the admission control and scheduling problems. Our technique transforms any single queue admission control strategy (preemptive or non-preemptive) to a scheduling and admission control algorithm for our general m queues model, whose competitive ratio is at most twice the competitive ratio of the given admission control strategy. We use our technique to derive concrete algorithms for the general preemptive and non-preemptive cases, as well as for the interesting special cases of the 2-value model and the unit value model. To the best of our knowledge this is the first result combining both scheduling and admission control decisions for arbitrary packets sequences in multi-queue switches. We also provide a 1.58-competitive randomized algorithm for the unit value case. This case is interesting by itself since most current networks (e.g. IP networks) only support a best-effort service in which all packets streams are treated equally.

1 Introduction

Overview: During recent years, network traffic has increased steadily, mainly due to the constant growing use of the Internet for both commercial and personal purposes. This phenomenon, combined with the fact that Internet traffic tends to fluctuate constantly, frequently overloads networking systems causing considerable degradation in the quality of communication based applications. As a result, the concept of networks supporting guaranteed Quality of Service (QoS) to

* `azar@tau.ac.il`. School of Computer Science, Tel-Aviv University, Tel-Aviv, 69978, Israel. Research supported in part by the Israeli Ministry of industry and trade and by the Israel Science Foundation.

† `yo@tau.ac.il`. School of Computer Science, Tel-Aviv University, Tel-Aviv, 69978, Israel. Research supported in part by the Israeli Ministry of industry and trade.

each traffic stream, in terms of bandwidth, latency, maximum drop rate etc., has received growing attention lately within the communication community. Since network overloads are frequent, QoS switches often have to cope with increasing amounts of overloaded traffic, while attempting to maximize the weighted throughput, where the weights correspond to the required quality of service for each packet. Hence, the quality of the decisions made by the switch can be measured by considering the total weight of packets it managed to pass through.

We model the problem of maximizing switch throughput in QoS networks as follows. A switch has m incoming FIFO queues and one output port. At each time unit new packets arrive to the queues, each packet marked with a value that corresponds to its guaranteed quality of service. Additionally, at each time unit the switch selects one non-empty queue and transmits the packet at the head of the queue through the output port. Since the m incoming queues have bounded capacities, arriving packets can overflow the queues, and some packets must be discarded. The goal is to maximize the total value of transmitted packets. We consider both the non-preemptive model and the preemptive model, where packets stored in the queues can be discarded in order to free space for new packets. Traditionally, similar problems were analyzed while assuming either some constant structure of the sequence of arriving packets, or a specific distribution of the arrival rates (see e.g. [4, 13]). We avoid any assumptions on the input and use competitive analysis to compare the performance of online algorithms to the optimal solution. In our work we face the combination of two dependent problems: admission control, namely which packets to discard in case of queue overflow, and scheduling, i.e. from which queue to transmit in every time unit. We present a generic technique that decouples the above problems and transforms any admission control strategy for a single queue (both preemptive and non-preemptive) to an algorithm for our general model (preemptive or non-preemptive, respectively). The competitive ratio of the constructed algorithm is at most twice the competitive ratio of the given admission control strategy. We therefore generalize known results [1, 2, 9, 10, 12] for preemptive and non-preemptive single queue admission control to the general model of m queues.

In addition, we also study the special case where all packets have unit value, and the goal is to maximize the number of transmitted packets. This model is interesting by itself, since the majority of current networks (most notably, IP networks) do not yet integrate full QoS capabilities, and provide a "best effort" service, where packets belonging to different traffic streams are treated equally within intermediate switches. We show a randomized algorithm and lower bounds for this special case.

Our results:

- Our main contribution is a generic technique to transform an admission control strategy for a single queue (both preemptive and non-preemptive) to a scheduling and admission control algorithm (preemptive or non-preemptive, respectively) for a switch with m queues. The competitive ratio of the constructed algorithm is at most twice the competitive ratio of the original single queue admission control algorithm.
- We use our generic technique to devise a 4-competitive algorithm for our general m queues preemptive model with packets of arbitrary values.
- In addition we employ our technique to construct the following algorithms:

- A $(2e\lceil \ln \alpha \rceil)$-competitive algorithm for the general non-preemptive model where α is the ratio between the largest value to the smallest one.
- An approximately 2.6-competitive algorithm for the preemptive 2-value model.
- A $(4 - \frac{2}{\alpha})$-competitive algorithm for the non-preemptive 2-value model where the values are restricted to 1 and α.
- We show that any "reasonable" online algorithm (defined later) is 2-competitive for the special case of unit value packets.

* We present a $(\frac{e}{e-1})$-competitive randomized algorithm for the unit value case. We also show deterministic and randomized lower bounds in this model.

We prove our upper bounds while assuming that all the queues in the switch are of equal size. We note that this is done for simplicity of notation only. The algorithms we present, including their bounds and analysis, remain the same when the queues have different sizes.

Our techniques: For our generic technique, we begin by considering a relaxation of our preemptive m queues model, in which packets can be transmitted in any order out of the queues, not necessarily FIFO. We present a natural algorithm in this preemptive relaxed model and analyze its performance by using a potential function. We then formulate our generic algorithm which is given a single queue admission control strategy as a parameter. The algorithm uses the given strategy for admission control in all m queues. In addition, the algorithm runs a simulation of the algorithm in the relaxed preemptive model and adopts its scheduling decisions. We prove that the simulation we use allows us to analyze our algorithm's performance in each queue separately.

We also investigate the special case of unit value packets. We construct a reduction to the problem of finding a maximum matching in a bipartite graph, whose unique property is its independence of the switching algorithm. This method can be combined with the techniques provided in [8] to produce a randomized algorithm which is 1.58-competitive. This is in contrast to a natural randomized algorithm that turns out to be no better than $(2 - o(1))$-competitive. The randomized lower bound is proved by modelling the problem as a Markov chain (which corresponds to a non-uniform random walk). A careful analysis of this Markov chain provides the lower bound.

Related results: The online problem of throughput maximization in switches supporting QoS has been studied extensively during recent years. Aiello *et al.* [1] initiated the study of different queuing policies for the 2-value non-preemptive model in which the switch has a single queue, preemption is not allowed and each packet has a value of either 1 or α. Recently, Andelman *et al.*[2] showed tight bounds for this case. The preemptive 2-value single queue model was initially studied by Kesselman and Mansour [10], followed by Lotker and Patt-Shamir [12] who showed almost tight bounds. The general preemptive single queue model, where packets can take arbitrary values, was investigated by Kesselman *et al.* [9], who proved that the natural greedy algorithm is 2-competitive (specifically $2\alpha/(1+\alpha)$-competitive where $\alpha \geq 1$ is the ratio between the largest value to the smallest one). Our work generalizes all the above results for the general m queues model. An alternative model to ours is the shared memory QoS switch, in which memory is shared among all queues. Hahne *et al.* [7] studied buffer management policies in this model while focusing on deriving upper and lower bounds for the natural Longest Queue Drop policy.

Koga [11] and Bar-Noy *et al.* [3] investigated the online problem of minimizing the length of the longest queue in a switch, which is in some sense the dual to the unit value case we study. In their model queues are unbounded in size, hence packets are not lost. Koga [11] proved that the natural greedy algorithm that always empties the longest queue is $\Theta(\log m)$-competitive. Bar-Noy *et al.* [3] suggested a different algorithm that simulates the greedy algorithm in the continuous model and is also $\Theta(\log m)$-competitive. Chrobak *et al.* [6] studied the more general problem of minimizing the length of the longest queue where queues can be emptied subject to conflicts constraints.

Paper structure: Section 2 includes formal definitions and notations. Our generic technique is shown in Section 3. In section 4 we present our randomized algorithm for the unit value case. Section 5 contains deterministic and randomized lower bounds for the unit value problem.

2 Problem definition and notations

We model the switch throughput maximization problem as follows. We are given a switch with m FIFO queues, where queue i has size B_i, and one output port. Packets are arriving online, each packet is destined to one of the queues and is associated with a non-negative value. We denote the online packets sequence by σ. Initially, the m queues are empty. We assume that time is discrete, and each time unit $t \geq 0$ is divided to two phases: at the beginning of the first phase of time t a set $\sigma(t)$ of packets arrive to the queues. Packets can be inserted to each queue without exceeding its capacity. Remaining packets must be discarded. In the second phase of time t, the switching algorithm may select one of the non-empty queues and transmit the packet at the head of the queue. The goal is to maximize the total value of transmitted packets. We consider both the non-preemptive model and the preemptive model, in which previously stored packets can be discarded from the queues. We also study two interesting special cases of our model: the 2-value case, in which packets values are restricted to 1 and α, and the unit value model in which all packets have unit value.

Given an online switching algorithm A we denote by $A(\sigma)$ the value of A given the sequence σ, and by $A^t(\sigma)$ the value of A until time t (inclusive). We denote the optimal (off-line) algorithm by OPT, and use similar notations for it.

A deterministic online algorithm A is c-competitive for a problem iff for every instance of the problem and every packets sequence σ we have: $OPT(\sigma) \leq c \cdot A(\sigma)$. We say that a randomized online algorithm A is c-competitive iff for every sequence σ the following holds: $OPT(\sigma) \leq c \cdot E[A(\sigma)]$. We claim that a problem has a lower bound c if no algorithm can achieve a competitive ratio strictly lower than c. Given an online algorithm A, we denote its competitive ratio by C_A.

We often focus on algorithms that transmit a packet *every time* a packet is available. We refer to such algorithms as *reasonable* online algorithms. One can easily verify that any online algorithm A can be transformed into a reasonable online algorithm A' such that $C_{A'} \leq C_A$.

3 Competitive algorithms for QoS switch management

In this section we present a generic technique that decouples the admission control and the scheduling problems and transforms any admission control strategy for a single queue (for both the preemptive and non-preemptive models) to a competitive algorithm for the general m queues model.

We use our generic technique to construct concrete competitive algorithms for both the preemptive and non-preemptive cases.

We begin by defining a natural greedy preemptive admission control strategy for a single queue.

Algorithm $GREEDY$
Enqueue a new packet if:

- The queue is not full.
- Or the packet with the smallest value in the queue has a lower value than the current packet. In this case the smallest packet is discarded and the new packet is enqueued.

We now turn to consider a relaxation of our preemptive model, in which packets can be transmitted from each queue in any order, not necessarily FIFO. We note that although this relaxation adds considerable strength to the online algorithm, the optimal solution remains the same. Therefore, when referring to the optimal solution we do not distinguish between the original FIFO model and its relaxation. We present the following natural greedy online algorithm for the preemptive relaxed model.

Algorithm TransmitLargest (TL**)**

1. **Admission control:** use algorithm $GREEDY$ for admission control in all m incoming queues.
2. **Scheduling:** at each time unit, transmit the packet with the largest value among all packets stored in the queues.

We return to our original FIFO model and present our generic technique **GenericSwitch** (abbreviated GS) for both the preemptive and non-preemptive models. We focus on *asynchronous* admission control strategies for a single queue, defined as follows.

Definition 1. An admission control strategy for a single FIFO queue is called *asynchronous* if it can handle arrival of packets at continuous time.

To the best of our knowledge, all known admission control strategies for a single queue are asynchronous. We next present the definition of GS with an asynchronous admission control strategy A as a parameter.

Algorithm GS^A**:**

1. **Admission control:** apply admission control strategy A to all m incoming queues.
2. **Scheduling:** run a simulation of algorithm TL (in the preemptive relaxed model) with the online input sequence σ. At each time unit transmit the packet at the head of the queue used by TL simulation.

In GS^A, admission control is carried out by exercising the asynchronous admission control strategy A on all queues. Consequently, algorithm GS^A is preemptive only if A itself is preemptive, and non-preemptive otherwise. Scheduling is handled independently by simulating the operation of algorithm TL (which is defined in the preemptive relaxed model) on the online input sequence σ, and adopting all its scheduling decisions, with no regard to the values of the packets transmitted by TL or the packets residing in its queues. It is crucial to note that we use the simulation of TL in the *preemptive* relaxed model even when we generate a *non-preemptive* algorithm GS^A from a single queue *non-preemptive* admission control strategy A. Our main result is that the competitive ratio of GS^A is at most twice the competitive ratio of A.

We begin our analysis by bounding the performance of algorithm TL.

Theorem 1. *Algorithm TL is 2-competitive in the preemptive relaxed model.*

Proof. For every $i = 1, \ldots, m$ denote by $\{v_{ij}^t\}$ the values of the packets stored in queue i at time t in TL, sorted from largest to smallest. Similarly, denote by $\{\bar{v}_{ij}^t\}$ the sorted values in queue i at time t in OPT. For simplicity of notation, we often omit the superscript t when meaning is clear, and we always consider $1 \leq j \leq B$, and pad the sequences with 0's, if necessary. Note that as a consequence, every time a packet is inserted to queue i, a value is discarded from the corresponding sorted sequence. We may view it as though we virtually extend the sorted sequence to $B+1$ entries, where entry $B+1$ holds the discarded value (which may be a padded '0' entry). If a packet is inserted to the queue upon its arrival we refer to it as an accepted packet, otherwise we refer to it as a rejected packet. Recall that whenever algorithm TL discards a packet, it is the packet with the smallest value in the queue. The following observation argues the same with regard to the optimal solution.

Observation 1. Whenever OPT discards a packet in the relaxed model, it is the packet with the smallest value in the queue.

For every queue i and time t we define $d_i^t = \sum_{j=1}^{B} (\bar{v}_{ij}^t - v_{ij}^t)_+$, where $x_+ = \max\{x, 0\}$. We define the following potential function: $\Phi^t = \sum_{i=1}^{m} d_i^t$. Note that $\Phi^t \geq 0$ for every t.

Lemma 2. *For every packets sequence σ and time unit $t \geq 0$ the following inequality holds: $OPT^t(\sigma) + \Phi^t \leq 2 \cdot TL^t(\sigma)$.*

Proof. We prove the lemma by induction on the time units. For $t = 0$ the inequality clearly holds. We assume correctness by the end of time unit $t-1$ and prove that the inequality holds when time unit t is finished. Denote by Δx the change incurred in the value of x when an operation takes place in the system, i.e. a packet arrives or a packet is transmitted. The next two claims prove that the inequality holds for every single operation occurring in the system during time t.

Claim 3. *For each packet arriving at the first phase of time t we have: $\Delta OPT + \Delta\Phi \leq 2 \cdot \Delta TL$.*

Proof. In the first phase of each time unit, packets are not transmitted, therefore $\Delta OPT = \Delta TL = 0$. Consider any packet arriving at time t. Let i be the queue to which the packet is destined. We examine the possible cases and prove that for all of them $\Delta\Phi \leq 0$. Clearly, $\Delta d_j = 0$ for all $j \neq i$, hence it suffices to check Δd_i. Note that we use here $\{v_{ij}\}$ and $\{\bar{v}_{ij}\}$ to denote the sorted values sequences *after* the insertion.

1. **The packet is accepted by both OPT and TL.** Let $k \leq B$ be the index of the new packet in the sequence of sorted values of queue i in OPT. Let $l \leq B$ be the corresponding index for TL. We check the possible cases:

(a) $\boldsymbol{k \leq l}$:

$$\begin{aligned}
\Delta d_i &= \sum_{j=k}^{B}(\bar{v}_{ij} - v_{ij})_+ - \left[\sum_{j=k+1}^{l}(\bar{v}_{ij} - v_{i(j-1)})_+ + \sum_{j=l+1}^{B+1}(\bar{v}_{ij} - v_{ij})_+\right] \\
&\leq \sum_{j=k}^{B}(\bar{v}_{ij} - v_{ij})_+ - \left[\sum_{j=k+1}^{l}(\bar{v}_{ij} - v_{i(j-1)})_+ + \sum_{j=l+1}^{B}(\bar{v}_{ij} - v_{ij})_+\right] \\
&= \sum_{j=k}^{l}(\bar{v}_{ij} - v_{ij})_+ - \sum_{j=k+1}^{l}(\bar{v}_{ij} - v_{i(j-1)})_+ = 0,
\end{aligned}$$

where the last equality results from the fact that $\bar{v}_{ij_1} \leq v_{ij_2}$ for every $k \leq j_1, j_2 \leq l$.

(b) $\boldsymbol{k > l}$:

$$\begin{aligned}
\Delta d_i &= \sum_{j=l}^{B}(\bar{v}_{ij} - v_{ij})_+ - \left[\sum_{j=l}^{k-1}(\bar{v}_{ij} - v_{i(j+1)})_+ + \sum_{j=k+1}^{B+1}(\bar{v}_{ij} - v_{ij})_+\right] \\
&\leq \sum_{j=l}^{B}(\bar{v}_{ij} - v_{ij})_+ - \left[\sum_{j=l}^{k-1}(\bar{v}_{ij} - v_{i(j+1)})_+ + \sum_{j=k+1}^{B}(\bar{v}_{ij} - v_{ij})_+\right] \\
&= \sum_{j=l}^{k}(\bar{v}_{ij} - v_{ij})_+ - \sum_{j=l}^{k-1}(\bar{v}_{ij} - v_{i(j+1)})_+ \\
&= \sum_{j=l}^{k}(\bar{v}_{ij} - v_{ij})_+ - \sum_{j=l}^{k-1}(\bar{v}_{ij} - v_{i(j+1)}) \qquad (1) \\
&= \sum_{j=l}^{k}(\bar{v}_{ij} - v_{ij})_+ - \left[\sum_{j=l}^{k}(\bar{v}_{ij} - v_{ij}) - \bar{v}_{ik} + v_{il}\right] \qquad (2) \\
&= \sum_{j=l}^{k}(\bar{v}_{ij} - v_{ij})_+ - \sum_{j=l}^{k}(\bar{v}_{ij} - v_{ij})_+ = 0, \qquad (3)
\end{aligned}$$

where in (2) $\bar{v}_{ik} = v_{il}$ and (1) and (3) follow from the fact that $\bar{v}_{ij_1} \geq v_{ij_2}$ for every $l \leq j_1, j_2 \leq k$.

2. **The packet is accepted by $\boldsymbol{OPT}$, rejected by $\boldsymbol{TL}$.** Let k be the index of the new packet in the list of sorted values in queue i in OPT. We have:

$$\Delta\Phi = \Delta d_i = \sum_{j=k}^{B}(\bar{v}_{ij} - v_{ij})_+ - \sum_{j=k+1}^{B+1}(\bar{v}_{ij} - v_{i(j-1)})_+ \leq \sum_{j=k}^{B}(\bar{v}_{ij} - v_{ij})_+ = 0,$$

where the last equality results from the fact that $\bar{v}_{ij_1} \leq v_{ij_2}$ for every $k \leq j_1, j_2 \leq B$, since TL rejected the new packet.

3. **The packet is accepted by $\boldsymbol{TL}$, rejected by $\boldsymbol{OPT}$.** Let l be the index of the new packet in the list of sorted values in queue i in TL. We get:

$$\Delta\Phi = \Delta d_i = \sum_{j=l}^{B}(\bar{v}_{ij} - v_{ij})_+ - \sum_{j=l}^{B}(\bar{v}_{ij} - v_{i(j+1)})_+ \leq 0,$$

where the last inequality follows from the fact that $(\bar{v}_{ij} - v_{ij})_+ \leq (\bar{v}_{ij} - v_{i(j+1)})_+$ for every $l \leq j \leq B$.

4. **The packet is rejected by both $\boldsymbol{OPT}$ and $\boldsymbol{TL}$.** Clearly, $\Delta\Phi = 0$.

□

Claim 4. *For the transmission phase in time t the following holds: $\Delta OPT + \Delta\Phi \leq 2 \cdot \Delta TL$.*

Proof. In this context we denote by $\{v_{ij}\}$ and $\{\bar{v}_{ij}\}$ the sorted values sequences *before* the transmission takes place. Let r be the queue from which TL takes a packet for transmission. Define $v_{r(B+1)} = 0$. We have:

$$\begin{aligned}
\Delta d_r &= \sum_{j=1}^{B}(\bar{v}_{rj} - v_{r(j+1)})_+ - \sum_{j=1}^{B}(\bar{v}_{rj} - v_{rj})_+ \\
&\leq \sum_{j=1}^{B}[(\bar{v}_{rj} - v_{rj})_+ + (v_{rj} - v_{r(j+1)})] - \sum_{j=1}^{B}(\bar{v}_{rj} - v_{rj})_+ \\
&= \sum_{j=1}^{B}(v_{rj} - v_{r(j+1)}) = v_{r1} - v_{r(B+1)} = v_{r1} = \Delta TL.
\end{aligned}$$

Let s be the queue from which OPT takes the packet with the kth largest value for transmission (of course $r = s$ is possible, and then $\{v_{sj}\}$ is the sequence after the first change). We have:

$$\begin{aligned}
\Delta d_s &= \sum_{j=k+1}^{B}(\bar{v}_{sj} - v_{s(j-1)})_+ - \sum_{j=k}^{B}(\bar{v}_{sj} - v_{sj})_+ \\
&\leq \sum_{j=k+1}^{B}(\bar{v}_{sj} - v_{sj})_+ - \sum_{j=k+1}^{B}(\bar{v}_{sj} - v_{sj})_+ - (\bar{v}_{sk} - v_{sk}) \\
&\leq -(\Delta OPT - \Delta TL),
\end{aligned}$$

where the last inequality results from $v_{sk} \leq \Delta TL$. Putting it all together we get:

$$\Delta OPT + \Delta\Phi = \Delta OPT + \Delta d_r + \Delta d_s \leq \Delta OPT + \Delta TL - (\Delta OPT - \Delta TL) = 2 \cdot \Delta TL$$

□

Claims 3 and 4 imply that the inequality holds when time t is finished. This completes the proof of Lemma 2. □

Theorem 1 follows directly from Lemma 2. $\square$

We now return to our original FIFO model and analyze the performance of GS. Before we proceed we wish to elaborate on the intuition behind our generic algorithm. Algorithm GS uses the simulation of algorithm TL to decide at each time unit which queue to use. This enables us to compare our algorithm's throughput with TL's throughput for each queue *separately*. Informally, this means that we can forget about the scheduling problem, lose a competitive factor of 2 since we use TL that is 2-competitive, and focus on our performance in each queue separately. We are now ready to state the main theorem of the paper.

Theorem 2. *Let GS^A denote the algorithm obtained by running algorithm GS with the asynchronous single queue admission control strategy A (preemptive or non-preemptive). Then $C_{GS^A} \leq 2 \cdot C_A$.*

Proof. We begin by introducing some new definitions and notations. Given the input sequence σ, denote by σ_i $(i = 1, \ldots, m)$ the sequence of packets arriving at queue i. For a given input sequence σ, define τ_i^k to be the time unit at which algorithm TL transmits a packet from queue i for the k-th time ($\tau_i^k = \infty$ if queue i is used less than k times). We now define a more compact representation of σ_i, denoted by $\hat{\sigma}_i$, which relies on TL operation. We consider only time units in which queue i was used for transmission, and define $\hat{\sigma}_i(t) = (\sigma_i(\tau_i^{t-1}+1), \ldots, \sigma_i(\tau_i^t))$, where we concatenate packets arriving between time units τ_i^{t-1} and τ_i^t and assign them all to the latter time unit. Consider an algorithm ALG, that exercises an independent asynchronous admission control policy in each queue (denote by A_i the policy used in queue i) and makes the same scheduling decisions as TL. Then, we can decouple admission control and scheduling and obtain : $ALG(\sigma) = \sum_{i=1}^m A_i(\hat{\sigma}_i)$. Specifically, we have: $GS^A(\sigma) = \sum_{i=1}^m A(\hat{\sigma}_i)$ and $TL(\sigma) = \sum_{i=1}^m TL(\hat{\sigma}_i)$, where we denote by TL both the algorithm for m queues and the restriction to a single queue.

We can now prove the desired competitive ratio:

$$OPT(\sigma) \leq 2 \cdot TL(\sigma) = 2\sum_{i=1}^m TL(\hat{\sigma}_i) \leq 2\sum_{i=1}^m OPT(\hat{\sigma}_i) \leq 2\sum_{i=1}^m C_A \cdot A(\hat{\sigma}_i) = 2 \cdot C_A \cdot GS^A(\sigma),$$

where the first inequality follows from Theorem 1, the third inequality follows from the fact that the optimal solution is at least as good as TL for $\hat{\sigma}_i$ and the forth inequality follows from the fact that the optimal solution is the same for the preemptive and the non-preemptive models. $\square$

We now show how to combine our generic technique with known admission control algorithms for a single queue, in order to construct specific preemptive and non-preemptive algorithms for our general m queues FIFO model. These examples demonstrate both the flexibility and the strength of our generic technique. All the following theorems are derived directly from Theorem 2.

General preemptive model: Kesselman *et al.* [9] proved that algorithm $GREEDY$ is 2-competitive in the single queue preemptive model. There follows:

Theorem 3. *Algorithm GS^{GREEDY} in the general preemptive model is 4-competitive.*

General non-preemptive model: Andelman *et al.* [2] recently presented a non-preemptive admission control algorithm for a single queue called **Exponential-Interval Round Robin** (abbreviated $EIRR$), which is $(e\lceil \ln \alpha \rceil)$-competitive, where α denotes the ratio between the largest value in the packets sequence σ and the smallest one . Therefore:

Theorem 4. *Algorithm GS^{EIRR} for the general non-preemptive model is ($2e\lceil \ln \alpha \rceil$)-competitive .*

2-value preemptive model: In this special case, studied in [10, 12], the values of the packets are restricted to two values, 1 and α. Lotker and Patt-Shamir [12] presented their mf (abbreviation for **mark&flush**) single queue preemptive admission control algorithm for the problem whose competitive ratio is approximately 1.3. Combined with GS we obtain:

Theorem 5. *Algorithm GS^{mf} for the 2-value preemptive model is approximately 2.6-competitive.*

2-value non-preemptive model: Andelman *et al.* [2] presented a single queue non-preemptive algorithm called **Ratio Partition** (abbreviated RP) for this case, with competitive ratio $2 - \frac{1}{\alpha}$, where α denotes the ratio between the largest value in the packets sequence σ and the smallest one. We obtain:

Theorem 6. *Algorithm GS^{RP} for the 2-value non-preemptive model is ($4 - \frac{2}{\alpha}$)-competitive.*

Unit value packets: In this special case all packets have unit values and the goal is to maximize the number of transmitted packets. This model correspond to networks lacking QoS capabilities, most notably IP networks.

Theorem 7. *Every reasonable online algorithm is 2-competitive in the unit value model.*

Proof. Note that the admission control algorithm $GREEDY$ is 1-competitive in the unit value model. Combined with Theorem 2 we obtain that algorithm GS^{GREEDY} is 2-competitive. Moreover, since all packets have unit values algorithm TL (which dictates GS scheduling decisions) can use any non-empty queue at each time unit, hence every reasonable algorithm is 2-competitive. $\square$

4 Randomized algorithm for unit value packets

We present the following randomized algorithm for the unit value model.

Algorithm RandomSchedule (RS):

1. The algorithm uses m auxiliary queues, each of size B. These queues contain real numbers from the range $(0, 1)$, where each number is labelled as either marked or unmarked. Initially these queues are empty. To avoid confusion between the auxiliary queues and the switch queues holding the packets, denote the former by $Q_1, \ldots, Q_m$ and the latter by $q_1, \ldots, q_m$.

2. Consider the packets arrival phase in each time unit. Suppose a new packet arrives at queue q_i. The algorithm chooses uniformly at random a real number from the range $(0, 1)$, that is inserted to queue Q_i and labelled as unmarked. If queue Q_i was full when the packet arrived, the number at the head of the queue is deleted prior to the insertion of the new number.

3. During the transmission phase in every time unit, we check whether queues $Q_1, \ldots, Q_m$ contain any unmarked number. If there are unmarked numbers, let Q_i be the queue containing the largest unmarked number. We change the label of the largest number to 'marked' and select queue q_i for transmission in this time unit. Otherwise (no unmarked numbers), we transmit a packet from any non-empty queue, if such exists.

Theorem 8. *For every sequence σ, $\frac{OPT(\sigma)}{E[RS(\sigma)]} \leq \frac{e}{e-1} + o(1) \approx 1.58$.*

Proof. We begin by introducing a translation of our problem to the problem of finding a maximum matching in a bipartite graph. We then prove the competitive ratio of algorithm RS by a reduction to the online algorithm for bipartite matching shown in [8]. We note that in the unit value model, there is no admission control question, since there is no reason to prefer one packet over the other. Therefore, we deal with the scheduling problem alone.

Given a sequence σ, we translate it to the bipartite graph $G^\sigma = (U, V, E)$, which is defined as follows.

- Let T denote the latest time unit in σ in which a packet arrives. We define the set of *time nodes* as $U = \{u_1, \ldots, u_{T+mB}\}$.
- Let P be the total number of packets specified in σ. We define the set of *packet nodes* as $V = \{v_1, \ldots, v_P\}$.
- Let P_i^t denote the set of the last B packets that arrive to queue q_i until time t (inclusive). Define $P^t = \bigcup_{i=1}^m P_i^t$. We define the set of edges in G^σ as follows: $E = \{(u_t, v_p) | p \in P^t\}$.

Before we proceed we introduce some new definitions.

Definition 2. A *schedule* S for a sequence of arriving packets σ is a set of pairs of the form (t, q_i), where queue q_i is scheduled for transmission at time t. The *size of the schedule*, denoted $|S|$, is the size of the set.

Definition 3. A schedule S for a sequence σ is called *legal* if for every pair (t, q_i), queue q_i is not empty at time t.

The following lemmas connect bipartite matching to our problem.

Lemma 5. *Every legal schedule S for the sequence σ can be mapped to a matching M in G^σ such that $|S| = |M|$.*

Proof. Let S be a legal schedule for σ. We construct the desired matching M incrementally while moving ahead in time. For each pair $(t, q_i) \in S$, we connect node u_t to node v_j, where $j = \min\{1 \leq k \leq P \,|\, v_k \in P_i^t, v_k \notin M\}$. A simple induction proves that for each time t and queue q_i the number of unmatched nodes in P_i^t is equal to the number of packets residing in queue q_i at time t according to S. Hence, every transmitted packet can be mapped to an edge in M. Clearly, by the construction $|S| = |M|$. □

Lemma 6. *Every matching M in G^σ can be translated in polynomial time to a legal schedule S for σ such that $|S| = |M|$.*

Proof. Let M be a matching in G^σ. We construct a legal schedule S for σ incrementally, while going over the nodes in U, starting from u_1. Let node u_t be connected in M to node $v_j \in P_i^t$. Then we add the pair (t, q_i) to the schedule S. A simple induction shows that for every i and t, the number of nodes from P_i^t that are included in M is at most the number of packets residing in queue q_i at time t according to S. Therefore, we can always translate an edge in M to a packet transmission in S, and our obtained schedule is legal. Clearly, this translation takes polynomial time and by the construction $|S| = |M|$. $\square$

The following corollaries directly result from Lemmas 5 and 6.

Corollary 7. *For any sequence σ, the size of the optimal schedule for σ is equal to the size of a maximum matching in G^σ.*

Corollary 8. *For any sequence σ, an optimal schedule can be found (off-line) in polynomial time.*

Consider algorithm RS as an algorithm for finding a matching in G^σ. The algorithm essentially maintains an order on the nodes in P^t, and connects node u_t to the first node from P^t (according to the maintained order) that has not been used yet. In fact, algorithm RS operates on G^σ exactly as the algorithm presented in [8] for the online maximum bipartite matching problem. Hence, the ratio between the size of the maximum matching in G^σ and the size of the matching constructed by algorithm RS is at most $\frac{e}{e-1} + o(1)$. Clearly, by the algorithm's operation (step 3), for every $1 \leq i \leq m$, the number of packets in queue q_i is at least the number of unmarked elements in Q_i. Therefore, there is always an available packet in q_i when the algorithm chooses it for transmission (step 3 in RS). In fact, it is worthwhile to note that the number of transmitted packets can be larger than the size of the constructed matching. Hence, according to corollary 7, RS is $(\frac{e}{e-1})$-competitive $\square$

5 Lower bounds for the unit value model

In the following theorems we prove deterministic and randomized lower bounds for the unit value case.

Theorem 9. *Every deterministic online algorithm for the unit value case has competitive ratio at least $2 - 1/m$.*

Proof. Fix any online algorithm A. We consider the case of unit size queues, i.e. $B = 1$. The adversary constructs the following sequence σ:

- At time $t = 0$, m packets arrive, one for each queue.
- Immediately before any time $t = 1, \ldots, m-1$, A has at least one full queue. At time $t = 1, \ldots, m-1$ the adversary generates a packet destined to queue i_t, where i_t is an index of one of the full queues in A. Clearly, A can not accept a single packet from this sequence. At time t ($t = 0, \ldots, m-2$) OPT empties queue i_{t+1}, so it can accept all packets in the sequence.
- At the end of time unit $t = m-1$, all queues in OPT are full except one, and all queues in A are empty. From this time on, no packets arrive, hence only packets currently stored will be transmitted.

Clearly: $\frac{OPT(\sigma)}{A(\sigma)} = \frac{2m-1}{m} = 2 - \frac{1}{m}$. $\square$

In the above lower bound we set $B = 1$. The next theorem shows a lower bound for any specific value of B.

Theorem 10. *For any specific value of B, every deterministic online algorithm for the unit value case has competitive ratio at least $1.366 - \Theta(1/m)$.*

Proof. Consider any specific value for B and fix any online algorithm A. We prove the theorem for any number of queues. We distinguish between two possible cases:

$\boldsymbol{B \leq m}$: Let us assume that B divides m (note that since the adversary can reduce the number of active queues, this assumption is w.l.o.g). The sequence σ produced by the adversary is as follows:

- At time $t = 0$, B packets arrive at each queue.
- The sequence consists of $(k \cdot \frac{m}{B} - 1)$ B-phases, each composed of B consecutive time units. In each B-phase packets arrive only at the last time unit. Denote by i_j the most loaded queue in A before the last time unit of B-phase j. At the last time unit of B-phase j, B new packets arrive at queue i_j.
- After the B-phases are finished, no additional packets arrive.

At the beginning of any time unit $1 \leq t \leq mB$ note that A holds at least $mB - t$ packets in its queues. Therefore, A has at least one queue with total load at least $B - \lfloor \frac{t}{m} \rfloor$. For B-phase j ($j = 1, \ldots (k \cdot \frac{m}{B} - 1)$) OPT empties queue i_j during the first B time units, and hence it accepts all arriving packets at the end of the phase. We now analyze the respective throughput of A and OPT:

$$\begin{aligned} OPT(\sigma) &= mB + km - B = m(B + k) - B \\ A(\sigma) &= mB + (\frac{m}{B} - 1) \cdot 0 + \frac{m}{B} \cdot 1 + \ldots \frac{m}{B} \cdot (k - 1) = m\left(B + \frac{k(k-1)}{2B}\right). \end{aligned}$$

Therefore, we get:

$$\frac{OPT(\sigma)}{A(\sigma)} = \frac{B + k - \frac{B}{m}}{B + \frac{k^2}{2B} - \frac{k}{2B}} \geq \frac{B + k - \frac{2B}{m}}{B + \frac{k^2}{2B}} = \frac{B + k}{B + \frac{k^2}{2B}} - \Theta\left(\frac{1}{m}\right),$$

we take $k = \alpha B$, where $\alpha = -1 + \sqrt{3}$ maximizes the expression. We obtain:

$$\frac{OPT(\sigma)}{A(\sigma)} \geq \frac{\sqrt{3}}{1 + \frac{1}{2}(-1 + \sqrt{3})^2} - \Theta\left(\frac{1}{m}\right) = \frac{1}{2}(1 + \sqrt{3}) - \Theta\left(\frac{1}{m}\right) \approx 1.366 - \Theta\left(\frac{1}{m}\right).$$

We note that although k can be a real number, either $\lfloor k \rfloor$ or $\lceil k \rceil$ obtain our desired ratio.

$\boldsymbol{B > m}$: The adversary generates a sequence σ similar to the one used in the previous case, only now it is composed of k B-phases. From the same considerations as before we get:

$$\frac{OPT(\sigma)}{A(\sigma)} = \frac{mB + kB}{mB + \lfloor \frac{B}{m} \rfloor + \ldots + \lfloor \frac{kB}{m} \rfloor} \geq \frac{m + k}{m + \frac{k(k+1)}{2m}} \geq \frac{m + k}{m + \frac{k^2}{2m}} - \Theta\left(\frac{1}{m}\right) \geq 1.366 - \Theta\left(\frac{1}{m}\right),$$

where the last inequality follows from the same calculation as in the previous case. $\square$

Theorem 11. *Every randomized online algorithm for the unit value case has competitive ratio at least* $1.46 - \Theta(1/m)$.

Proof. We prove the lower bound for unit size queues, i.e. $B = 1$. We provide a probability distribution on sequences of inputs. We prove the lower bound for any deterministic algorithm. Since any randomized algorithm is a probability distribution on deterministic ones this also provides a lower bound for any randomized algorithm (see also [5], chapter 8 for Yao's theorem). The adversary constructs the following sequence σ:

- At time $t = 0$, m packets arrive, one for each queue.
- For every time $t = 1, \ldots, r$ a packet arrives to a queue that is randomly selected according to the uniform distribution.
- After time r no additional packets arrive.

Clearly, OPT can accept all packets, because it always transmits a packet from the queue to which the next packet will arrive. Hence, its throughput is exactly $m + r$. On the other hand, the behavior of any online algorithm A can be described as a Markov chain. The Markov chain has m states. State i for $0 \leq i \leq m-1$ corresponds to a total of i packets in the queues. Clearly, at the beginning of time $t = 1$ the algorithm is in state $m-1$. Let X_t denote the random variable that indicates the state of the online deterministic algorithm at time t, and let ΔX_t denote the indicator random variable with value 1 iff the algorithm changes its state during time t (clearly, $X_{t+1} = X_t - \Delta X_t$). We further denote by p_i^t the probability of the algorithm to be in state i at time t. At any time the probability of moving from state i to state $i-1$ is exactly i/m and the probability of staying at state i is $1 - i/m$. Therefore, $E[\Delta X_t] = Pr[\Delta X_t = 1] = \sum_{i=1}^{m-1} \frac{i}{m} p_i^t$. The expected state at time t, which corresponds to the expected number of packets in the queues, is: $E[X_t] = \sum_{i=1}^{m-1} i \cdot p_i^t = m \cdot E[\Delta X_t]$. As a result, $E[X_{t+1}] = E[X_t - \Delta X_t] = E[X_t] - E[\Delta X_t] = \frac{m-1}{m} E[x_t]$. Hence, after r time units the expected number of packets in the queues is $(m-1)(1-1/m)^r$. Since the online algorithm transmitted at most one packet at each time unit we conclude that the expected throughput of any online algorithm is at most $1 + r + (m-1)(1-1/m)^r$. By the appropriate (optimal) choice of r (i.e. taking $r = \alpha m$ where $\alpha = 1.146$) to maximize the ratio $\frac{m+r}{1+r+(m-1)(1-1/m)^r}$ we conclude that the ratio is at least $1.46 - \Theta(1/m)$. □

References

[1] W. A. Aiello, Y. Mansour, S. Rajagopolan, and A. Rosen. Competitive queue policies for differentiated services. In *Proceedings of the IEEE INFOCOM 2000*, pages 431–440.

[2] 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.

[3] A. Bar-Noy, A. Freund, S. Landa, and J. Naor. Competitive on-line switching policies. In *Proc. 13th ACM-SIAM Symp. on Discrete Algorithms*, pages 525–534, 2002.

[4] A. Birman, H. R. Gail, S. L. Hantler, Z. Rosberg, and M. Sidi. An optimal service policy for buffer systems. *Journal of the Association Computing Machinery (JACM)*, 42(3):641–657, 1995.

[5] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

[6] M. Chrobak, J. Csirik, C. Imreh, J. Noga, J. Sgall, and G. J. Woeginger. The buffer minimization problem for multiprocessor scheduling with conflicts. In *Proc. 28th International Colloquium on Automata, Languages, and Programming*, pages 862–874, 2001.

[7] E. L. Hahne, A. Kesselman, and Y. Mansour. Competitive buffer management for shared-memory switches. In *Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 53–58, 2001.

[8] R. Karp, U. Vazirani, and V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of 22nd Annual ACM Symposium on Theory of Computing*, pages 352–358, may 1990.

[9] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko. Buffer overflow management in QoS switches. In *Proc. 33rd ACM Symp. on Theory of Computing*, pages 520–529, 2001.

[10] A. Kesselman and Y. Mansour. Loss-bounded analysis for differentiated services. In *Proc. 12th ACM-SIAM Symp. on Discrete Algorithms*, pages 591–600, 2001.

[11] H. Koga. Balanced scheduling toward loss-free packet queuing and delay fairness. In *Proc. 12th Annual International Symposium on Algorithms and Computation*, pages 61–73, 2001.

[12] Z. Lotker and B. Patt-Shamir. Nearly optimal fifo buffer management for diffserv. In *Proc. 21st ACM Symp. on Principles of Distrib. Computing*, pages 134–143, 2002.

[13] M. May, J. C. Bolot, A. Jean-Marie, and C. Diot. Simple performance models of differentiated services for the internet. In *Proceedings of the IEEE INFOCOM 1999*, pages 1385–1394.