Competitive Buffer Management for Shared-Memory Switches

3

WILLIAM AIELLO

University of British Columbia

ALEX KESSELMAN

Google Inc.

AND

YISHAY MANSOUR

Tel-Aviv University

Abstract. We consider buffer management policies for shared memory switches. We study the case of overloads resulting in packet loss, where the constraint is the limited shared memory capacity. The goal of the buffer management policy is that of maximizing the number of packets transmitted. The problem is online in nature, and thus we use competitive analysis to measure the performance of the buffer management policies. Our main result is to show that the well-known preemptive Longest Queue Drop (LQD) policy is at most 2-competitive and at least $\sqrt{2}$-competitive. We also demonstrate a general lower bound of $4/3$ on the performance of any deterministic online policy. Finally, we consider some other popular non-preemptive policies including Complete Partition, Complete Sharing, Static Threshold and Dynamic Threshold and derive almost tight bounds on their performance.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Packet-switching networks, store and forward networks*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Sequencing and scheduling*

General Terms: Algorithms, Performance, Theory

Additional Key Words and Phrases: Buffer management, competitive analysis, shared memory

---

A preliminary version of this article appeared in *Proceedings of the 13th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2001)*, (Munich, Germany, June 14–16), ACM, New York, 2001, pp. 53–58.

Authors' addresses: W. Aiello, Department of Computer Science, University of British Columbia, Vancouver, BC, Canada, e-mail: aiello@cs.ubs.ca; A. Kesselman, Google, Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043, e-mail: alx@google.com; Y. Mansour, School of Computer Science, Tel-Aviv University, P.O. Box 39040, Tel-Aviv 69978, Israel, e-mail: mansour@ca.tau.ac.il.

ACM Reference Format:
Aiello, W., Kesselman, A., and Mansour, Y. 2008. Competitive buffer management for shared-memory switches. ACM Trans. Algor. 5, 1, Article 3 (November 2008), 16 pages. DOI = 10.1145/1435375.1435378 http://doi.acm.org/10.1145/1435375.1435378

1. *Introduction*

In this article, we consider Shared Memory (SM) switch architecture. In SM switches output queues are dynamically allocated from a shared memory pool. Although memory sharing can provide a better queuing performance, by taking advantage of the statistical multiplexing, it requires careful design of the buffer management policy in order to guarantee a fair and robust operation. Irland [1978] demonstrated that SM switches with complete sharing (i.e., a packet is accepted if there is enough available space) perform poorly under overload conditions. The problem is that a single output port can take over most of the memory, preventing packets destined for other output ports from gaining access, which causes the total switch throughput to drop. An important task of a buffer management policy is to avoid starvation, for example by restricting the amount of buffering an output port can use, which makes buffer space available to the less utilized ports and hopefully increases the total switch throughput.

The buffer management policy has to decide for each incoming packet whether to *accept* or *reject* it. In addition, the buffer management policy may have the ability to *preempt* packets that have been already accepted. Clearly, the buffer management policy has to ensure that at any point of time the memory consumed does not exceed the total shared memory capacity. The buffer management policies are traditionally classified to two categories: preemptive and nonpreemptive, according to whether they utilize the preempt action. It is worth to note that nonpreemptive policies with simple threshold-type controls can be implemented in a distributed fashion. The tradeoff here is between ease of implementation and hardware (where nonpreemptive policies have an advantage) and higher performance (where preemptive policies have an advantage). Both types of policies have been widely considered in the networking literature. For a good survey of shared-memory buffer management policies, the reader can refer to Arpaci and Copeland [2000].

The main class of nonpreemptive scheduling policies, which are also called static threshold schemes, was considered by Irland [1978]. In sharing with maximum queue lengths (*SMXQ*) scheme, each output queue has a static bound on its length and a packet is accepted if there is a free space in the buffer and the corresponding bound is not violated. In some schemes, like sharing with a maximum queue and minimum allocation (*SMQMA*) due to Kamoun and Kleinrock [1980], each port always has access to a minimum allocated space. The main problem of the static threshold schemes is that they are not adaptive. When many queues are active and the sum of their thresholds exceeds the buffer capacity, the buffer may fill up completely, even though all queues are obeying the threshold constraints. Thus, some queues can be starved, which leads to underutilization of the switch. On the other hand, when very few queues are active, they are denied access to the idle buffer space beyond the sum of their thresholds. This creates higher packet loss rate for the active queues (see Choudhury and Hahne [1998]).

Another class of nonpreemptive policies includes dynamic threshold schemes. In the Dynamic Threshold policy due to Choudhury and Hahne [1998], the threshold

on a queue length at any instant of time is proportional to the current amount of unused buffering in the switch multiplied by some constant. Packet arrivals for an output port are blocked whenever the corresponding queue length equals or exceeds the current threshold value. The main idea is that the Dynamic Threshold policy deliberately holds a small amount of buffer space in reserve and divides the remaining buffer space equally among the active output queues.

The class of preemptive policies has been also studied extensively. A delayed resolution policy (*DRP*) was proposed by Thareja and Agrawala [1984]. The *DRP* does not discard an arriving packet if there is space in the common buffer. If a packet arrives and the common buffer is full, the arriving packet, or some other packet that was already accepted, is discarded. The decision to drop a packet from a certain queue of an output port can be made based on the state of the system or based on different priority classes. Wei et al. [1991] propose to drop from the longest queue in the switch, when the memory is full. A comparison of preemptive policies has been provided by Kroner [1990].

We consider a $N \times N$ switch with shared memory of size M. A buffer management policy is presented with packet arrivals and has to serve packets online, that is, without knowledge of future arrivals. The goal of the buffer management policy is to maximize the number of packets sent, subject to the memory capacity constraint. We use competitive analysis [Sleator and Tarjan 1985; Borodin and El-Yaniv 1998], to study our policies. In competitive analysis the online policy is compared with an optimal offline policy, which knows the entire input sequence in advance. The advantage of competitive analysis is that a uniform performance guarantee is provided over all input instances.

Our Results. We show that the well-known preemptive Longest Queue Drop (*LQD*) policy due to Wei et al. [1991] is 2-competitive. We also demonstrate a general lower bound of $4/3$ on the performance of any deterministic online policy, and a lower bound of $\sqrt{2}$ on the competitive ratio of the *LQD* policy. We further analyze some popular nonpreemptive policies including Complete Partition, Complete Sharing, Static Threshold and Dynamic Threshold. We establish that the competitive ratio of the Complete Partition and the Complete Sharing policies is $O(N)$ and this bound is approximately tight. Then, we demonstrate that a more sophisticated Static Threshold policy achieves a competitive ratio of of $O(\sqrt{N})$ and this bound is also nearly tight. Finally, we derive a lower bound of $\Omega(\sqrt{N/\log N})$ on the competitive ratio of the Dynamic Threshold policy under the default setting of parameters (to be defined later), where the memory size $M = \Theta(N)$.

Related Work. This work is most closely related to the work of Kesselman and Mansour [2004], which considers nonpreemptive buffer management policies for shared memory switches. The work of Kesselman and Mansour [2004] proposes a new buffer management policy called Harmonic whose competitive ratio is $O(\log N)$ and establishes a lower bound of $\Omega(\log N / \log\log N)$ on the performance of any online nonpreemptive deterministic policy. In contrast to Kesselman and Mansour [2004], in this article, we demonstrate that preemptive policies can achieve a *constant* competitive ratio.

Azar and Richter [2005] consider a weighted multi-queue switch problem with FIFO buffers and present a 4-competitive algorithm. An improved 3-competitive algorithm is given by Azar and Richter [2004]. Albers and Schmidt [2005] develop

a deterministic 1.89-competitive algorithm for the case of unit-value packets. In a recent paper, Azar and Litichevskey [2006] derived a 1.58-competitive algorithm for switches with large buffers.

Kesselman and Rosén [2006] study CIOQ switches with FIFO buffers. For the case of packets with unit values, they present a switch policy that is 3-competitive for any speedup. For the case of packets with variable values, they propose two switch policies achieving a competitive ratio of $4S$ and $8\min(k, 2\log\beta)$, where S is the speedup of the switch, k is the number of distinct packet values and β is the ratio between the largest and the smallest value. Azar and Richter [2006] obtain a 8-competitive algorithm for CIOQ switches with FIFO buffers for the latter case.

The problem of throughput maximization in the context of a single buffer has been explored extensively in recent years (see Epstein and Stee [2004] for a good survey). Aiello et al. [2005] concentrate on the case where one cannot preempt packets already in the buffer and consider the special case of two different packet values. This model is extended to the case of variable value packets by Andelman et al. [2003]. Kesselman et al. [2004] study preemptive policies for FIFO buffers and introduce a new bounded-delay model.

The rest of the article is organized as follows. The model description appears in Section 2. Section 3 contains analysis of the *LQD* policy. Analysis of alternative policies appears in Section 4. We conclude with Section 5.

2. *Model Description*

We consider a $N \times N$ switch with shared memory of size M (see Figure 1). Packets, of equal size, arrive at input ports, and each packet is labeled with the output port on which it has to leave the switch. All the packets destined to a given output port are organized in a First-In-First-Out (FIFO) queue.

Time is slotted. We divide each time step into two phases. The first phase is the *transmission* phase during which the first packet from every non-empty output queue is sent on the corresponding output link. The second phase is the *arrival* phase during which at most one new packet may arrive at each input port. However, our upper bounds will also hold for a bursty arrivals process, where multiple packets may arrive simultaneously. During the arrival phase the input ports are served in a fixed order from input port 1 to input port N. The buffer management policy processes packets in the same order and the decision of whether to admit the packet arriving at input port i depends only on the state of the buffer after processing the arrivals at input ports $1, \ldots, i-1$.

A buffer management policy determines how the shared memory is used by individual output ports of the switch. More specifically, when a packet arrives the buffer management policy decides whether to *accept* or *reject* it. The accepted packet can be later *preempted*, that is, dropped from the buffer. We consider memoryless policies, whose decisions depend solely on the instantaneous state of the buffer.

Next we introduce a few useful definitions. Let *ALG* be the buffer management policy. It would be convenient to argue about the position of a packet p in the FIFO order, that is, the number of packets preceding p in the queue.

Definition 2.1. We denote the *position* of a packet p in the FIFO order at the end of time slot t by $pos^t_{ALG}(p)$.

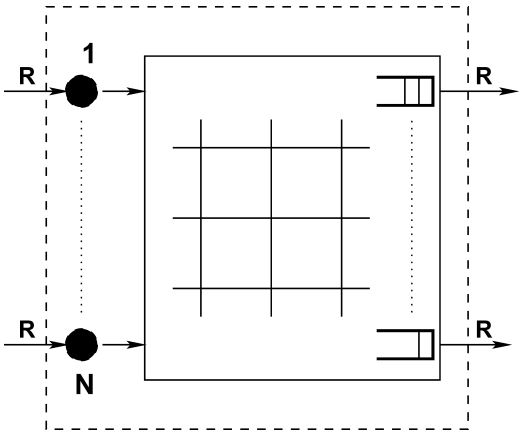

FIG. 1. An example of SM switch.

Definition 2.2. We denote by $L^t_{ALG}(q)$ the length of output queue q at the end of time slot t.

The buffer management policy accrues a unit value for each packet successfully transmitted and gains no value on dropped packets. The aim of the buffer management policy is that of maximizing the total number of packets sent. Let σ be a sequence of packets arriving at the inputs of the switch. Let $V_{ALG}(\sigma)$ and $V_{OPT}(\sigma)$ be the number of packets transmitted by an online policy *ALG* and an optimal offline policy *OPT*, respectively. The throughput-competitive ratio is defined as follows.

Definition 2.3. An online buffer management policy *ALG* is c throughput-competitive iff for every sequence of packets σ, $V_{OPT}(\sigma) \leq c \cdot V_{ALG}(\sigma) + a$, where a is a constant independent of σ.

Finally, we define a burst of packets, which will be extensively used in our lower bound constructions.

Definition 2.4. We denote by *burst* a set of packets that arrive at the same or at consequent time slots from different input ports and are all destined to the same output port.

3. *Analysis of Preemptive Policies*

In this section, we first present a general lower bound and then analyze the *LQD* policy.

3.1. GENERAL LOWER BOUND.

We show that no deterministic online policy can achieve throughput-competitive ratio better than $4/3$.

THEOREM 3.1. *The competitive ratio of any deterministic online policy is at least* $4/3$ *for sufficiently large* N.

PROOF. Suppose that the buffer is controlled by a deterministic online policy *ALG*. Consider the following scenario in which there are only two active output ports. At time $t = 0$ the memory is empty and during the following $\lceil 2M/(N-2) \rceil$ time slots two bursts arrive. Each burst contains $N/2$ packets destined to output

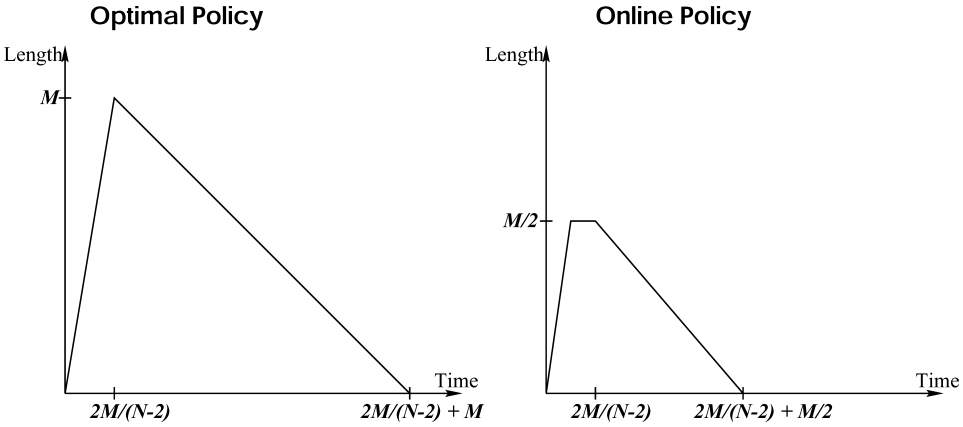

FIG. 2. Utilization of the queue of output port 1.

When a packet p arrives:

. If there is free space in the buffer, accept p.
. If p is destined to the longest queue, reject it.
. Otherwise drop a packet from the tail of the longest queue and accept p.

FIG. 3. Longest Queue Drop Policy (*LQD*).

ports 1 and 2, respectively. It must be the case that at time $t = \lceil 2M/(N-2) \rceil$, *ALG* retains in its memory at most $M/2$ packets destined to either output port 1 or 2. WLOG, assume that it is output port 1. During the following M time slots, one packet destined to output port 2 arrives every time slot.

Consider the time interval $\mathcal{I} = [0, \lceil 2M/(N-2) \rceil + M]$ (see Figure 2). On the one hand, *ALG* is able to transmit at most $\lceil M(3N+2)/(2(N-2)) \rceil$ packets: $\lceil 4M/(N-2) \rceil$ packets during the first $\lceil 2M/(N-2) \rceil$ time slots and $M/2$ and M packets from output ports 1 and 2 during the following M time slots. On the other hand, *OPT* buffers at time $t = \lceil 2M/(N-2) \rceil$, M packets destined to output port 1 and transmits $\lceil 2MN/(N-2) \rceil$ packets: $\lceil 4M/(N-2) \rceil$ packets during the first $\lceil 2M/(N-2) \rceil$ time slots and M and M packets from output ports 1 and 2 during the following M time slots.

At time $t + \lceil 2M/(N-2) \rceil + M + 1$ we repeat the scenario and so on. For a long run the ratio between the number of packets sent by *OPT* and that of *ALG*, that is $\frac{4N}{3N+2}$, asymptotically approaches to $4/3$, and the theorem follows. □

3.2. ANALYSIS OF *LQD* POLICY. We demonstrate that the competitive ratio of *LQD* is at most 2 and at least $\sqrt{2}$. The *LQD* policy is presented on Figure 3. Roughly speaking, *LQD* tries to balance the throughput of all output ports.

In what follows, we assume a given input packet sequence σ. To analyze the throughput of the *LQD* policy, we introduce some helpful definitions. The next definition concerns packets that *OPT* delivers during a time slot while *LQD* does not.

Definition 3.2. A packet transmitted by *OPT* at time slot t from output queue q is an *extra* packet if at this time the corresponding output queue of *LQD* is idle.

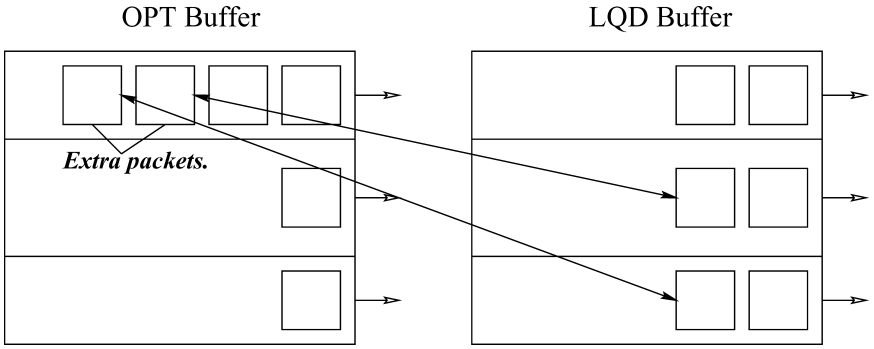

FIG. 4. A matching example.

Each time slot t do:

(1) If during the arrival phase a matched LQD packet p is preempted, then the packet p' that is accepted in place of p replaces it in the matching.

(2) At the end of the arrival phase match each unmatched OPT packet p in output queue q for which $pos^t_{OPT}(p) > L^t_{LQD}(q)$ as follows:

(a) If p arrives at the current time slot and it is accepted by both OPT and LQD, then match p to itself in the LQD memory.

(b) Otherwise, match p to an arbitrary unmatched packet in the LQD memory.

(*) Step 2a is applied to *all* packets before Step 2b.

FIG. 5. The matching routine for *LQD*.

Note that all extra packets sent by *OPT* should eventually appear in an output queue of *OPT* when the corresponding *LQD* queue is empty. Now we introduce a notion of *potential* extra packets.

Definition 3.3. We call a packet p in output queue q of *OPT* at time slot t a *potential extra packet* if $pos^t_{OPT}(p) > L^t_{LQD}(q)$.

In order to prove that *LQD* achieves the competitive ratio of 2, we will match each extra packet of *OPT* to a packet sent by *LQD*, in a way that each *LQD* packet is matched at most once (see Figure 4).

The matching routine presented in Figure 5 guarantees that all potential extra packets are matched to packets sent by *LQD* (this will be proved in what follows). Note that all extra packets became at some point of time prior to their transmission potential extra packets. The intuition is that we try to match potential extra packets as early as they appear, which guarantees that all extra packets are also matched at time they are sent by *OPT*. The matching routine is executed each time slot during the arrival phase, and adds some matchings according to the actions of *LQD* and *OPT*.

OBSERVATION 3.4. *All extra packets are matched before they are transmitted.*

We will show that the matching routine is feasible. First, we demonstrate a technical yet important property, that is, for any pair of matched packets, the *LQD*, packet is in a lower position in the memory than the *OPT* packet. This will guarantee that for any pair of matched packets the *LQD* packet is sent before the *OPT* packet. Observe that the two matched packets can be destined to different output ports.

OBSERVATION 3.5. *If at time slot t an OPT packet p from output queue q is matched to an LQD packet p', $p' \neq p$, then the LQD memory is full and some packets from output queue q of LQD are dropped.*

Since p is not matched by Step (2)(a), some packets from output queue q of *LQD* are dropped. This also implies that the *LQD* memory is full.

We now show that the position of the matched packets is smaller in *LQD* than in *OPT*.

CLAIM 3.6. *Suppose at time slot t a packet p in OPT in output queue q, is matched to packet p' in LQD. Then $pos^t_{OPT}(p) \geq pos^t_{LQD}(p')$.*

PROOF. Suppose that p and p' are matched at time $t' \leq t$. First, note that if $pos^{t'}_{OPT}(p) \geq pos^{t'}_{LQD}(p')$ then $pos^t_{OPT}(p) \geq pos^t_{LQD}(p')$, since both *OPT* and *LQD* transmit packets at the same speed. It remains to prove that the claim holds at time t', which will be done by considering the three possible cases for matching.

Step (1). The first case is when *LQD* preempts at time t' another packet p'', which is matched to p, and accepts p'. This implies that $pos^{t'}_{LQD}(p') \leq pos^{t'}_{LQD}(p'')$, because *LQD* always preempts packets from the longest queue. Thus, the position property remains satisfied through preemption.

Step (2)(a). The second case is when p arrives at time t' and is matched to itself. By the definition of the matching, we have that

$$pos^{t'}_{OPT}(p) > L^{t'}_{LQD}(q) \geq pos^{t'}_{LQD}(p).$$

Note that we consider that same output queue in *OPT* and in *LQD*.

Step (2)(b). The third case is when p is matched to an arbitrary packet in the *LQD* memory. By Observation 3.5, the memory of *LQD* is full at time t' and q is the longest queue in the memory of *LQD*. By the definition of the matching, $pos^{t'}_{OPT}(p) > L^{t'}_{LQD}(q)$. But the position of any packet in the *LQD* memory is at most $L^{t'}_{LQD}(q)$. We obtain that

$$pos^{t'}_{OPT}(p) > L^{t'}_{LQD}(q) \geq pos^{t'}_{LQD}(p'). \quad \square$$

Now we use the position property to show that we can always find an unmatched packet in *LQD* to match to a potential extra *OPT* packet, when needed.

LEMMA 3.7. *The matching process never fails.*

PROOF. Consider the switch configuration at the end of a time slot t. If all packets are matched by Step (1) or Step (2)(a) of the matching routine, we are done. Otherwise, we will show that the number of unmatched potential extra *OPT* packets is bounded by the number of unmatched *LQD* packets. By Observation 3.5, the memory of *LQD* is full. Let the number of matched packets in *OPT* be x and the number of matched packets in *LQD* be y. According to Claim 3.6, matched *LQD* packets are scheduled not later than their *OPT* mates, which implies that $x \geq y$. Thus, the number of unmatched packets in *OPT* is bounded by $M - x$, which is at most the number of unmatched packets in *LQD*, that is $M - y$. Therefore, Step (2)(b) of the matching routine always succeeds. $\square$

Now we are ready to derive an upper bound the competitive ratio of *LQD*.

THEOREM 3.8. *The competitive ratio of the Longest Queue Drop policy is at most* 2.

PROOF. The number of packets transmitted by *OPT* is bounded by the number of packets transmitted by *LQD* plus the number of extra packets. In accordance with Lemma 3.7, the matching routine is feasible. This implies that the number of extra packets is bounded by the number of packets sent by *LQD*. Therefore, $V_{OPT}(\sigma) \leq 2V_{LQD}(\sigma)$. □

We now derive a specific lower bound of $\sqrt{2}$ on the competitive ratio of the *LQD* policy (improving upon a general lower bound of 4/3).

THEOREM 3.9. *The competitive ratio of the Longest Queue Drop policy is at least* $\sqrt{2}$ *for* $M > (N/3)^2$.

PROOF. Let us consider a $3A \times 3A$ switch with memory equal $M = L(A-1)/2 + A$, where $L > A >> 1$. We assume that L is a multiple of A. We divide the output ports into three classes: Active, Overloaded, and Idle. Each class contains A output ports. The idle output ports are not used in the analysis, but we need extra input ports to produce packet arrivals for the active and overloaded output ports (recall that the switch is $N \times N$).

Consider the following arrival pattern. Active output ports have an average load of 1, and this load is periodic and very bursty. The period length is L and it starts with a burst of L packets that arrive over A input ports in L/A successive time slots. Then, there are no arrivals to that output port for $L - L/A$ time slots. Bursts to the various active output ports are evenly staggered in time, with a new one starting every L/A time slots, just as the previous burst completes. Thus, given enough buffering, A fully utilized input ports will keep the A active output ports completely busy. Overloaded output ports receive exactly 2 packets every time slot. (Recall that we have $2A$ available input ports that are not deployed in generating packets for active output ports. We use these input ports to generate packets for the overloaded output ports.)

Now we will analyze the performance of *OPT* and *LQD* for this system. First, let us describe the optimal policy. For each overloaded output port at each time slot, *OPT* admits only one of the two arriving packets. This maintains a queue of one packet for each overloaded output port. *OPT* also accepts all the packets destined to the active output ports. The queue length at an active output port will form a sawtooth waveform over time, with a fast linear rise at rate $A - 1$ for L/A time slots, then a slow linear fall at rate 1 for $L - L/A$ time slots (see Figure 6). The total queue length, summed over all active output ports, will be fixed and equal to $L(A-1)/2$. Hence, the total queue length over all output ports is $L(A-1)/2 + A$. This exactly equals the buffer size M. Thus, *OPT* ensures that both the active and overloaded output ports are completely utilized (i.e., they send a packet each time slot), so *OPT*'s throughput at each time slot is exactly $2A$. Note that the buffer is completely full all the time.

Now let us consider the performance of *LQD*. In contrast with *OPT*, *LQD* allows the overloaded output ports to take up a substantial amount of memory space. This keeps the active output ports from getting all the buffering they need for their bursty loads. With *LQD*, the maximum queue length in the system (i.e., the

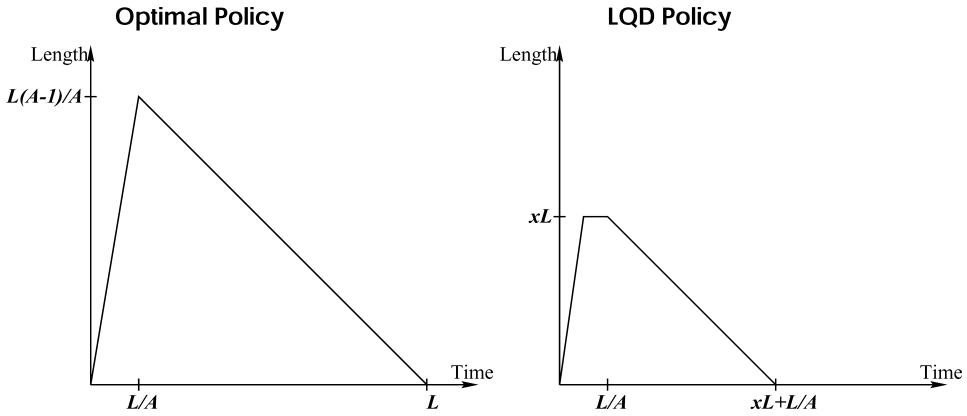

FIG. 6. Active output port utilization.

length of any overloaded port) is approximately constant.[1] The maximum queue length fluctuates near xL, with ripples of amplitude L/A. Since $1/A << x$, these ripples are a small fraction of the maximum queue length. We will show shortly that $x \approx (\sqrt{2}-1) \approx 0.41$. The queue length at each active output port will resemble a clipped sawtooth waveform over time, with a fast linear rise at rate $A-1$ to level xL, then a nearly flat plateau at xL for $(L/A) - xL/(A-1)$ time slots, then a slow linear fall at rate 1 for xL time slots, then steady value of 0 for the remaining $L - (L/A) - xL$ time slots (see Figure 6).

To solve for x, let us take a snapshot of the system at the end of a burst to an active output port. Each of the A overloaded output ports and the single active port that received the latest burst have length xL. Of the A active output ports, xA of them have positive queue lengths, evenly staggered and ranging from xL to 0 (viz., xL, $xL-(L/A)$, $xL-(2L/A)$, $xL-(3L/A)$,...). The other $(1-x)A$ active ports have no packets queued. The total memory space consumed by the active output ports is approximately $x^2AL/2$; to see this, approximate the sum with an integral, which amounts to taking the area of a right triangle with width xA and height xL. The total memory space consumed by the overloaded output ports is approximately xAL. We set the sum of these two quantities equal to the memory size $M = L(A-1)/2 + A$. Recalling that $L > A >> 1$ and solving for x yields: $x \approx (\sqrt{2}-1) \approx 0.41$.

Now let us figure the throughput of *LQD*. In every period of L time slots, the number of packets served at any active output port is the sum of the L/A packets served while that port's burst is arriving, plus the xL packets served from the queue after the burst has stopped. Since $1/A << x$, the throughput rate is approximately x. Thus, the total throughput of the active output ports is approximately $(\sqrt{2}-1)A$, and the total system average throughput of *LQD* is approximately $(\sqrt{2}-1)A + A = \sqrt{2} \cdot A$. Comparing this with *OPT*'s average throughput of $2A$, we obtain that the competitive ratio of *LQD* is at least $(2 \cdot A)/(\sqrt{2} \cdot A) = \sqrt{2}$. □

[1]The overall system is periodic with period L, but the maximum queue length repeats more frequently, with period L/A, corresponding to the length of successive bursts to successive active output ports.

4. *Analysis of Alternative Policies*

In this section, we consider alternative scheduling policies. We show that most of the policies have very poor competitive performance. One can expect such results because we compare *preemptive LQD* policy with *nonpreemptive* policies. It is worth to note that there is a correlation between the competitiveness of the policies derived here and their performance observed by Choudhury and Hahne [1998] (the better the competitive ratio, the higher the performance).

4.1. SIMPLE POLICIES.

In this section, we consider the Complete Partition and the Complete Sharing policies. The Complete Partition policy allocates to each output port exactly $1/N$ fraction of the buffer space. On the other hand, the Complete Sharing policy admits packets if there is some free space in the buffer allowing an output port to monopolize the whole memory. We demonstrate that the Complete Partition and Complete Sharing policies are $O(N)$-competitive and these bounds are almost tight.

We start with the analysis of Complete Partition. In the next theorem, we derive an upper bound on the competitive ratio of the Complete Partition policy.

THEOREM 4.1. *The competitive ratio of the Complete Partition policy is at most* N.

PROOF. Obviously, for each output port Complete Partition admits at least $1/N$ fraction of packets accepted by *OPT* since it is allocated a queue of size M/N and packets are transmitted at full speed. The theorem follows. □

The following theorem establishes a lower bound on the performance of the Complete Partition policy.

THEOREM 4.2. *The competitive ratio of the Complete Partition policy is at least* $N/2$ *for sufficiently large* N.

PROOF. Consider the following scenario. At time $t = 0$ the buffer is empty and during each of the following $\lceil M/(N-1) \rceil$ time slots arrives a burst of N packets destined to output port 1. Throughout the next M time slots there are no packet arrivals.

OPT, has its buffer full at time $t = \lceil M/(N-1) \rceil$ fully utilizing output port 1 during the interval $\mathcal{I} = [0, \lceil M/(N-1) \rceil + M]$. On the other hand, under the Complete Partition policy at time $t = \lceil M/(N-1) \rceil$ the queue of output port 1 contains $\lceil M/N \rceil$ packets and thus at most $\lceil M/(N-1) + M/N \rceil$ packets are transmitted throughout $\mathcal{I}$. At time $t = \lceil M/(N-1) \rceil + M + 1$ the scenario is repeated and so on. Therefore, the competitive ratio of the Complete Partition policy is at least

$$\frac{M/(N-1) + M}{M/(N-1) + M/N} = \frac{N^2}{2N-1}. \quad \square$$

Now we proceed to analyze the Complete Sharing policy. The next theorem shows an upper bound on the performance of the Complete Sharing policy.

THEOREM 4.3. *The competitive ratio of the Complete Sharing policy is at most* $N + 1$.

PROOF. We construct a matching between extra packets from *OPT* and packets sent by the Complete Sharing policy. Note that the number of potential extra *OPT*

When a packet p arrives:

. If there is no free space in the buffer, reject p.
. If the length of the queue p is destined to does not exceed the threshold, accept p.
. Otherwise, reject p.

FIG. 7. General Threshold Drop policy.

packet does not change if no packets are dropped by the Complete Sharing policy during the arrival phase. When Complete Sharing drops some packets, the length of the longest queue in its buffer is at least M/N. At the same time, the number of packets in the *OPT* buffer with position less than or equal to l $(1 \leq l \leq M)$ is bounded by lN while the total number of packets is at most M. Hence, one can match each potential extra *OPT* packet to a packet in the Complete Sharing buffer with smaller or equal position so that no Complete Sharing packet is matched more than N times. The theorem follows. □

In the following theorem, we demonstrate a lower bound on the competitive ratio of the Complete Sharing policy.

THEOREM 4.4. *The competitive ratio of the Complete Sharing policy is at least* N.

PROOF. Consider the following scenario. At time $t = 0$ the buffer is empty and during each of the following $\lceil M/(N-1) \rceil$ time slots arrives a burst of N packets destined to output port 1. Then, during a sufficiently long time interval $\mathcal{I} = [\lceil M/(N-1) \rceil + 1, T]$, every time slot arrives a packet for output port 1 followed by $N-1$ packets, one for each other output port.

Note that *OPT* will transmit at least $\lceil N(T - M/(N-1)) \rceil$ packets, by admitting N packets destined to all output ports every time slot $t > \lceil M/(N-1) \rceil$. On the other hand, under the Complete Sharing policy arrivals for all output ports, but 1, will be blocked. Hence, the throughput of the Complete Sharing policy is bounded by $T + M$. □

4.2. THRESHOLD-TYPE CONTROL POLICIES. In this section we study the Static Threshold and the Dynamic Threshold policies. In both of these policies a packet is admitted into the buffer if there is a free space and the corresponding threshold on the queue length is not exceeded (see Figure 7). The Static Threshold policy sets this threshold to the fraction f of the total memory while for Dynamic Threshold it equals the current amount of free memory multiplied by a parameter α.

We first consider the Static Threshold policy and demonstrate that its competitive ratio is at most $\sqrt{N} + 1$ and this bound is almost tight (up to a constant factor). Then, we proceed to the Dynamic Threshold policy and show that for $\alpha = 1$ its competitive ratio is bounded from below by $\Omega(\sqrt{N/\log N})$ for $M = \Theta(N)$. Note that one can still obtain better bounds for the Dynamic Threshold policy with properly tuned or variable α.

The next theorem shows that the competitive ratio of the Static Threshold policy is at least $\Omega(\max(1/f, fN))$.

THEOREM 4.5. *The competitive ratio of the Static Threshold policy is at least* $\Omega(\max(1/f, fN))$ *for sufficiently large* N.

PROOF. These bounds are obtained using scenarios that are analogous to that of Theorem 4.2 and Theorem 4.4, respectively.

To derive the first bound, consider the following scenario. At time $t = 0$ the buffer is empty and during each of the following $\lceil M/(N-1)\rceil$ time slots arrives a burst of N packets destined to output port 1. Throughout the next M time slots, there are no packet arrivals. *OPT* has its buffer full at time $t = \lceil M/(N-1)\rceil$ fully utilizing output port 1 during the interval $\mathcal{I} = [0, \lceil M/(N-1)\rceil + M]$. On the other hand, under the Static Threshold policy at time $t = \lceil M/(N-1)\rceil$ the queue of output port 1 contains $\lceil fM\rceil$ packets and thus at most $\lceil M/(N-1) + fM\rceil$ packets are transmitted throughout $\mathcal{I}$. At time $t = \lceil M/(N-1)\rceil + M + 1$ the scenario is repeated and so on. Therefore, the competitive ratio of the Static Threshold policy is at least

$$\frac{M/(N-1) + M}{M/(N-1) + fM},$$

which is $\Omega(1/f)$ for sufficiently large N.

To derive the second bound, consider the following scenario. At time $t = 0$ the buffer is empty and during each of the following $\lceil M/(N-1/f)\rceil$ time slots arrives a burst of $\lceil fM + M/(N-1/f)\rceil$ packets destined to each of output ports $1, \ldots, \lceil 1/f\rceil$. Then, during a sufficiently long time interval $\mathcal{I} = [\lceil M/(N-1/f)\rceil + 1, T]$, every time slot arrive $\lceil 1/f\rceil$ packets for output ports $1, \ldots, \lceil 1/f\rceil$ followed by $N - \lceil 1/f\rceil$ packets for output ports $\lceil 1/f\rceil + 1, \ldots, N$. Note that *OPT* will transmit at least $\lceil M/(f(N-1/f)) + N(T - M/(N-1/f))\rceil$ packets, by admitting N packets destined to all output ports every time slot $t > \lceil M/(N-1/f)\rceil$. On the other hand, under the Static Threshold policy arrivals for all output ports, but $1, \ldots, \lceil 1/f\rceil$, will be blocked. Hence, the throughput of the Complete Sharing policy is bounded by $M/(f(N-1/f)) + T/f + M$. Henceforth, the competitive ratio of the Static Threshold policy is at least

$$\frac{M/(f(N-1/f)) + N(T - M/(N-1/f))}{M/(f(N-1/f)) + T/f + M},$$

which is $\Omega(fN)$ for sufficiently large T. □

Observe that the minimal bound of $\Omega(\sqrt{N})$ is achieved when $f = 1/\sqrt{N}$. Now we show that for this value of f the Static Threshold policy is indeed $O(\sqrt{N})$-competitive.

THEOREM 4.6. *The competitive ratio of the Static Threshold policy with* $f = 1/\sqrt{N}$ *is at most* $\sqrt{N} + 1$.

PROOF. We construct a matching between extra *OPT* packets and packets sent by Static Threshold so that each Static Threshold packet is image of at most $\sqrt{N}$ packets in *OPT*. All potential extra *OPT* packets are matched immediately to packets in the Static Threshold buffer with smaller or equal position. We consider two cases.

—If the matching is done in time of Static Threshold buffer overflow, the number of packets in the buffer of Static Threshold with position at most l (for $1 \le l \le M/\sqrt{N}$) is at least $l\sqrt{N}$.

—If the matching is done when the buffer of Static Threshold is not full, all packets in *OPT* to be matched have position greater than $M/\sqrt{N}$ and at least one queue in Static Threshold has length of at least $M/\sqrt{N}$.

It follows that in both cases there exists a feasible matching of potential extra packets in *OPT* so that each Static Threshold packet is matched at most $\sqrt{N}$ times. □

Next we study the Dynamic Threshold policy and present a few lower bounds on its competitive ratio for $\alpha = 1$.

THEOREM 4.7. *The competitive ratio of the Dynamic Threshold policy for* $\alpha = 1$ *is at least* $\Omega(\sqrt{N/\log N})$ *for* $M = \Theta(N)$.

PROOF. Consider the following scenario. At time $t = 0$ the buffer is empty and during the following $\lceil M/N \rceil$ time slots arrive bursts of size $M/2^i$ ($i = 1, \ldots, \log M$) that are destined to output ports $1, \ldots, \log M$. Then, between $t = \lceil M/N \rceil$ and $t = \lceil M/N + \sqrt{N/\log M} \rceil$, every time slot arrive N packets destined to all output ports. At time $t = \lceil M/N + \sqrt{N/\log M} \rceil + 1$ we stop all arrivals.

Throughout time interval $[0, \lceil M/N \rceil - 1]$ *OPT* transmits at least M/N packets from output ports $1, \ldots, \log M$. Then, each time slot between $t = M/N$ and $t = \lceil M/N + \sqrt{N/\log M} \rceil$ it will accept and send N packets to all output ports. Therefore, during time interval $\mathcal{I} = [0, \lceil M/N + \sqrt{N/\log M} \rceil]$, the throughput of *OPT* is at least $\lceil M/N + N\sqrt{N/\log M} \rceil$. On the other hand, the Dynamic Threshold policy admits all packets arriving starting from $t = 0$ and up to $t = \lceil M/N \rceil$. At this time, Dynamic Threshold buffer contains at least $M - \lceil \frac{M}{N} \log M \rceil$ packets. During each of the following $\lceil \sqrt{N/\log M} \rceil$ time slots, there will appear at most $\log M$ additional free memory slots to accommodate new packets. Hence, starting from $t = \lceil M/N \rceil$ and up to $t = \lceil M/N + \sqrt{N/\log M} \rceil$, the Dynamic Threshold policy could accept at most

$$\left\lceil \left(2\frac{M}{N} \log M + \sqrt{N/\log M} \log M \right) \sqrt{N/\log M}/2 \right\rceil$$

additional packets. Notice that the number of packets destined to output ports $1, \ldots, \log M$ buffered before $t = \lceil M/N \rceil$ that is transmitted by the Dynamic Threshold policy is at most M. Thus, the total throughput of the Dynamic Threshold policy during $\mathcal{I}$ is bounded from above by

$$\left\lceil M + \left(2\frac{M}{N} \log M + \sqrt{N/\log M} \log M \right) \sqrt{N/\log M}/2 \right\rceil.$$

Therefore, the competitive ratio of the Dynamic Threshold policy is at most

$$\frac{M/N + N\sqrt{N/\log M}}{M + (2\frac{M}{N} \log M + \sqrt{N/\log M} \log M)\sqrt{N/\log M}/2},$$

which is $\Omega(\sqrt{N/\log N})$ for $M = \Theta(N)$. When all the queues are drained out, we repeat the scenario and so on. □

5. *Concluding Remarks*

In this article, we study the performance of buffer management policies for shared memory packet switches using competitive analysis. We show that preemptive policies are able to achieve a constant competitive ratio. That is in contrast to nonpreemptive policies, whose competitive ratio is bounded away from a constant. We also analyze some popular nonpreemptive policies and demonstrate that one can differentiate between them by means of competitive analysis.

An open problem is whether it is possible to achieve a constant competitive ratio for the case of variable value packets. Another open question is whether one can prove a nontrivial upper bound (better than N) on the competitive ratio of the Dynamic Threshold policy or prove a lower bound that does not depend on α. Finally, the gap between the upper and the lower bounds on the competitive ratio of the *LQD* policy is yet to be closed.

ACKNOWLEDGMENTS. We are very grateful to Ellen Hahne for many interesting discussions. We also would like to thank two anonymous referees for their helpful comments.

REFERENCES

AIELLO, W. A., MANSOUR, Y., RAJAGOPOLAN, S., AND ROSÉN, A. 2005. Competitive queue policies for differentiated services. *J. Algor. 55*, 2, 113–41.

ALBERS, S., AND SCHMIDT, M. 2005. On the performance of greedy algorithms in packet buffering. *SIAM J. Comput. 35*, 2, 278–304.

ANDELMAN, N., MANSOUR, Y., AND ZHU, A. 2003. Competitive queueing policies for qos switches. In *Proceedings of 14th ACM-SIAM Symposium on Discrete Algorithms (SODA 2003)*. ACM, New York, 761–770.

ARPACI, M., AND COPELAND, J. A. 2000. Buffer management for shared-memory atm switches. *IEEE Commun. Surv. 3*, 1, 2–10.

AZAR, Y., AND LITICHEVSKEY, M. 2006. Maximizing throughput in multi-queue switches. *Algorithmica 45*, 1, 69–90.

AZAR, Y., AND RICHTER, Y. 2004. The zero-one principle for switching networks. In *Proceedings of the 36th annual ACM symposium on Theory (STOC 2004)*. 64–71.

AZAR, Y., AND RICHTER, Y. 2005. Management of multi-queue switches in qos networks. *Algorithmica 43*, 1-2, 81–96.

AZAR, Y., AND RICHTER, Y. 2006. An improved algorithm for cioq switches. *ACM Trans. Algor. 2*, 2, 282–95.

BORODIN, A., AND EL-YANIV, R. 1998. Online Computation and Competitive Analysis. Cambridge University Press.

CHOUDHURY, A. K., AND HAHNE, E. L. 1998. Dynamic queue length thresholds for shared-memory packet switches. *IEEE/ACM Trans. Netw. 6*, 2 (Apr.), 130–140.

EPSTEIN, L., AND STEE, R. V. 2004. Sigact news online algorithms. *ACM SIGACT News 35*, 3 (Sept.), 58–66.

IRLAND, M. 1978. Buffer management in a packet switch. *IEEE Trans. Commun. COM-26*, 3 (Mar.), 328–37.

KAMOUN, F., AND KLEINROCK, L. 1980. Analysis of shared finite storage in a computer network node environment under general traffic conditions. *IEEE Trans. Commun. COM-28*, 7 (July), 992–1003.

KESSELMAN, A., AND MANSOUR, Y. 2004. Harmonic buffer management policy for shared memory switches. *TCS Special Issue on Online Algorithms in Memoriam of Steve Seiden 324*, 2-3, 161–82.

KESSELMAN, A., AND ROSÉN, A. 2006. Scheduling policies for cioq switches. *J. Algor. 60*, 1, 60–83.

KESSELMANM, A., LOTKER, Z., MANSOUR, Y., PATT-SHAMIR, B., SCHIEBER, B., AND SVIRIDENKO, M. 2004. Buffer overflow management in qos switches. *SIAM J. Comput. 33*, 3, 563–83.

KRONER, H. 1990. Comparative peformance study of space priority mechanisms for atm networks. In *Proceedings of IEEE INFOCOM 1990*. IEEE Computer Society Press, Los Alamitos, CA, 1136–1143.

SLEATOR, D., AND TARJAN, R. 1985. Amortized efficiency of list update and paging rules. *Commun. ACM 28*, 202–8.

THAREJA, A. K., AND AGRAWALA, A. K. 1984. On the design of optimal policy for sharing finite buffers. *IEEE Trans. Communications COM-32*, 6 (June), 737–740.

WEI, S. X., COYLE, E. J., AND HSIAO, M. T. 1991. An optimal buffer management policy for high-performance packet switching. In *Proceedings of IEEE GLOBECOM 1991*. IEEE Computer Society Press, Los Alamitos, CA, 924–928.

RECEIVED OCTOBER 2005; REVISED OCTOBER 2007; ACCEPTED OCTOBER 2007