Competitive Buffer Management for Shared-Memory Switches

Ellen L. Hahne* Alexander Kesselman† Yishay Mansour‡

ABSTRACT

We consider buffer management policies for shared memory packet switches supporting Quality of Service (QoS). There are two interesting dimensions in which the setting may different. The first is the packet size, whether all the packets of the same fixed size or do packets have variable length. The second is the value of the packets, do all the packets have the same value or do different packets have different values.

The goal of the buffer management policy is to maximize the total value of packets transmitted. Our main result is to show that the well-known Longest Queue Drop (LQD) policy is 2-competitive and at least $\sqrt{2}$-competitive for the case of fixed size and value packets. We also show a 4/3 general lower bound on the competitiveness in this case. We extend the results to the case of variable size fixed value packets, and derive a slightly worse bound. For the case of variable value we derive randomized policy whose competitive ratio is logarithmic on the ratio of the maximal to minimal value.

1. INTRODUCTION

In this paper we consider shared memory switches. Although memory sharing can provide a better queuing performance, by taking advantage of the statistical multiplexing, it requires carefully design of the buffer management policy in order to guarantee a fair and robust operation. 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.

*AT&T Labs Research, *email: hahne@research.att.com*.

†CheckPoint Software Tech. and Department of Computer Science, Tel-Aviv University, *email: alx@math.tau.ac.il*.

‡Department of Computer Science, Tel-Aviv University, *email: mansour@math.tau.ac.il*.

Shared-memory packet switches with complete sharing (i.e., a packet is accepted if there is enough available space) perform poorly under overload conditions [7]. 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 this phenomena, for example by restricting the amount of buffering an output port can use. This would hopefully make buffers available to the less utilized ports and increases the total switch throughput. For a good survey of shared-memory buffer management policies the reader can refer to [2].

The buffer management policies are traditionally classified to two categories: preemptive and non-preemptive, according to whether they utilize the preempt action. It is worth to note that non-preemptive policies with simple threshold-type controls may be run in parallel at the output port queue processors. The tradeoff here is between ease of implementation and hardware (where non-preemptive 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.

The class of preemptive policies, which are the subject of our work, has been studied with various queuing systems. A delayed resolution policy (DRP) was proposed by Thareja and Agrawala in [15]. 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. [16] propose to drop from the longest queue (LQD) in the switch, when the memory is full. A comparison of preemptive policies in shared memory has been provided in [11].

The main class of non-preemptive scheduling policies that was studied by Irland [7], is called static threshold schemes. In static threshold schemes 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 [8] each port always has access to a minimum allocated space. Another class of policies includes dynamic threshold schemes. In the Dynamic Threshold Policy due to Choudhury and Hahne [6], the queue length thresholds of the ports, at any instant in time, are proportional to the current amount of unused buffering in the switch multiplied by a parameter α. Packet arrivals for an output port are blocked whenever the output port queue length equals or exceeds the current threshold value.

Our model is the following. Each packet has a value and size, and is labeled with the destination output port on which it needs to be transmitted. The goal of the buffer management policy is to maximize the sum of the values of the packets transmitted, subject to the memory capacity constraint.

We use competitive analysis [14], [5], to study our policies. In competitive analysis the online policy is compared with an optimal offline policy, that knows the entire input sequence in advance. The advantage of competitive analysis is that a uniform performance guarantee is provided over all input instances.

We consider fixed as well as variable value and size packets. The fixed value corresponds to the Best Effort model, while the variable value, where each packet has an intrinsic value, is implicit in the recent DiffServ model [4], as well as other QoS models. Another dimension is the packet size. Fixed packet size may correspond to ATM cells while variable packet size may correspond to IP packets.

We show that the well known Longest Queue Drop (LQD) policy [16] for fixed value and size packets is 2-competitive. In addition we show a general lower bound on the competitive ratio for fixed size and value packets of 4/3, and a lower bound of $\sqrt{2}$ on the competitive ratio of the LQD policy. We extend the results to variable packet size and demonstrate that LQD achieves slightly worse competitive ratio of $2 + \frac{NC_{max}}{M-NC_{max}}$, where M is the shared memory capacity, N is the number of output ports and C_{max} is the maximum packet size. However, note that for a large enough memory ($M \gg NC_{max}$) this is an almost 2-competitive ratio. Finally, we consider variable value packets model and present a Randomized Partition policy that achieves the competitive ratio of $4 \log \alpha$, where α is the ratio between the maximal and the minimal values. In particular, we prove that for an important case of two possible packet values it is at most 4 competitive. For variable size and value packets we prove that the competitive ratio of the Randomized Partition policy is $4 \log \alpha + \frac{2NC_{max} \log \alpha}{M-NC_{max}}$.

Most recent research efforts in competitive throughput analysis have been focused on the management of a single FIFO buffer. The work of [1] concentrated on the case where one cannot preempt a packet already in the buffer. They give tight bounds on the competitive factor of various policies for the special case where there are two different weights. In [12], the question of video smoothing is studied, and an upper bound of 4 is derived on the competitive ratio of the greedy policy using the FIFO model; a result latter improved to 2 [9], where they also introduced a new bounded packet delay model. The work of [10] studies the competitive analysis of the lost value rather then the throughput value. Translating the loss bound to the throughput bound considered here, for the variant where only two packet weights are possible, one can show that the competitive ratio approaches one as the ratio of the two weights tends to infinity.

The rest of the paper is organized as follows. The model description appears in Section 2. Analysis of LQD for fixed value packets is presented in Section 3. In Section 4 we introduce the Randomized Partition policy for the case of variable packet values. We conclude with Section 5.

2. MODEL DESCRIPTION

We consider a $N \times N$ switch (i.e. N inputs and N outputs) with shared memory of size M. A buffer management policy determines how the shared memory will be used by individual output ports of the switch. More specifically, when a packet arrives the policy decides whether to *accept* or *reject* it, and the accepted packet can be later *preempted*. Conceptually we think of all the packets destined to a given output port as organized in a First-In-First-Out (FIFO) queue. First we consider fixed size packets whose transmission takes exactly one time slot. At the start of each time slot the first packet from every non-empty queue is sent to the corresponding output port.

It would be convenient to argue about the number of packets preceding a packet p at time t to its destination port (which equals the position of p in the FIFO order at time slot t), we denote this by $l(p,t)$.

Definition 2.1. *For a packet p let $s(p)$ to be its size and $v(p)$ be its value. For variable size packets we assume that $C_{min} \leq s(p) \leq C_{max}$, and for variable value packets we assume that $s(p) \leq v(p) \leq \alpha \cdot s(p)$, where α is a parameter.*

In the case of variable size packets we need to define how the packets are sent. We assume that at each time slot a single byte is transmitted. For simplicity we assume that once a packet has started to be transmitted, it can not be preempted. (This restriction applies only to the online policy, since, without loss of generality, we assume that the optimal offline policy never preempts packets.)

For the case of variable size packets the parameter α is used to bound the value per byte of different packets. In the analysis it would be convenient to assign each byte of a packet p a value of $v(p)/s(p)$, and sum the value of the bytes transmitted.

The aim of the buffer management policy is to maximize the value of the packets sent. As we explained before, WLOG, we assume that the optimal offline policy OPT does not preempt packets. Let $V_A(S)$ and $V_{OPT}(S)$ be the value of packets transmitted by an online policy A and an optimal offline policy OPT, respectively. The throughput-competitive ratio is defined as follows.

Definition 2.2. *An online buffer management policy A is c-throughput-competitive iff for every sequence of packets S, $V_{OPT}(S) \leq c \cdot V_A(S)$.*

3. FIXED VALUE PACKETS

In this section we consider fixed value packets. Since all the packets have the same value, the goal of the buffer management policy is simply to maximize the number of bytes transmitted.

3.1 Fixed Size Packets

In this sub-section we consider fixed size and value packets. First we show that no online policy can achieve throughput-competitive ratio better than 4/3. Then we demonstrate that the competitive ratio of Longest Queue drop (LQD) is at most 2 and at least $\sqrt{2}$.

THEOREM 3.1. *The competitive ratio of any online policy A is at least* $4/3$.

PROOF. Suppose that $N \geq 2M$ [1]. Consider the following scenario in which there are only two active output ports. At time $t = 0$ the memory is empty and two bursts arrive. The bursts contain M packets each and are destined to output ports 1 and 2, respectively. (We need that $N \geq 2M$ in order to generate such bursts.) By "burst" we mean a set of k packets that arrive at the same time slot from k different input ports and are all destined to the same output port. It must be the case that the online policy A retains in its memory at most $M/2$ packets destined to either port 1 or 2. WLOG, assume that it is port 1. During the following M time slots one packet destined to output port 2 arrives every time slot. Clearly, A is able to transmit at most $M + M/2$ packets in the time interval $\mathcal{I} = [t+1, t+M]$. On the other hand OPT would have initially buffered $M-1$ packets destined to output port 1 and transmitted $2M-1$ packets throughout the time interval $\mathcal{I}$.

At time $t+M+1$ we repeat the scenario and so on. For a sufficiently long run and sufficiently large M the ratio between the number of packets sent by OPT and A asymptotically approaches to 4/3, and the theorem follows. □

We now proceed to upper bound the competitive ratio of the LQD policy. First we define the policy.

Longest Queue Drop Policy (LQD): When a packet arrives, if it can be accepted without any preemptions, it is accepted. Otherwise we accept the packet and drop a packet from the tail of the longest queue. (This implies that if the packet is destined to the longest queue then it is rejected.)

We start by defining the set of packets that OPT sends and LQD does not send on the same output port at the same time slot.

DEFINITION 3.1. *A packet transmitted by OPT at time slot t on output port q is an* **extra** *packet if at time t the output port q of LQD was idle.*

[1]The assumption that $M \approx N$ is impractical, however, the scenario may be easily extended to more realistic setup with $M \gg N$.

Note that the number of packets sent by OPT is clearly bounded by the number of packets sent by LQD plus the number of extra packets. In order to prove that LQD achieves the competitive ratio of 2 we construct a matching between extra packets transmitted by OPT and packets sent by LQD. Notice that a packet can become an extra packet if at some time moment its position in OPT is greater that the length of the LQD queue corresponding to the same output port (it must not be necessarily the time slot at which the packet arrives). Each time there appears an OPT packet that can potentially become extra packet, we match it to an unmatched LQD packet. The intuition behind this is that we try to match potential extra packets as early as possible. Observe that in this way all extra packets are matched.

Matching: We iteratively construct a matching between OPT and LQD packets as follows:

a. Each time slot t for each unmatched OPT packet p in the queue to output port i:

    If $l(p,t)$ in OPT is greater than the length of the queue for output port i in LQD, then we match OPT packet p as follows:

        If p arrives at the current time slot and it is accepted by both OPT and LQD and the LQD memory is not full, then match OPT packet p to itself in LQD.

        Otherwise, match OPT packet p to any unmatched packet in LQD memory.

    Else do nothing.

b. If a matched LQD packet p is preempted, then the packet p' that is accepted in place of p replaces it in the matching.

We will show that such a matching is feasible by demonstrating two important properties. First, that we will always be able to find an unmatched packet in LQD to match to OPT packet p, when needed. Second, a more technical yet important property, is that for any pair of matched packets, the LQD packet is in a lower position in the memory then 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 may be destined to different output ports.

OBSERVATION 3.2. *If at time t an OPT packet p is matched to an LQD packet p', $p' \neq p$, then the LQD memory is full at time t.*

PROOF. It must be the case that at least one packet from the LQD queue corresponding to p's output port is dropped. This implies that the LQD memory is full. □

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

CLAIM 3.3. *Suppose at time t packet p in OPT, destined to output port i, is matched to packet p' in LQD. Then $l_{OPT}(p,t) \geq l_{LQD}(p',t)$.*

PROOF. Suppose that p and p' are matched at time $t' \leq t$. First note that if $l_{OPT}(p,t') \geq l_{LQD}(p',t')$ then $l_{OPT}(p,t) \geq l_{LQD}(p',t)$, since OPT does not perform preemptions, and we re-match packets after LQD's preemptions. It remains to prove that the claim holds at time t', which will be done by considering the three possible cases for matching.

(1) The first case is that p arrives at time t' and LQD memory is not full. This implies that $p' = p$, and by the definition of the matching we have $l_{OPT}(p,t') \geq l_{LQD}(p',t')$.

(2) The second case is that p at time t' is for the first time since its arrival at a higher location than the length of the queue of the ith output port in LQD. By Observation 3.2 the memory of LQD is full at time t' and some packets from the corresponding LQD queue q_i are dropped. Therefore q_i is the longest queue in LQD, and let its length be k. By the construction of the matching the latency of packets to be matched from the ith queue of OPT is at least $k+1$. Since the latency of all packets in LQD memory is at most k (k is the length of the longest queue). Thus the claim holds at time t'.

(3) The third case is when LQD preempts at time t' some packet $\hat{p}$, which is matched to p, and accepts p'. This implies that $l_{LQD}(p',t) \leq l_{LQD}(\hat{p},t)$, because LQD always preempts packets from the longest queue. Hence, the latency property remains satisfied over preemptions. □

LEMMA 3.4. *At any time t the number of unmatched OPT packets that must be matched is at most the number of unmatched LQD packets.*

PROOF. Consider a time moment t at which some OPT packets have to be matched. If all packets are matched to themselves then we are done. Otherwise, by Observation 3.2 the memory of LQD is full at time t. Consider the configuration at time t, after the output queues have been served and before the input queues have been served. Let the number of matched packets in OPT be x and the number of matched packets in LQD be y. By Claim 3.3 matched LQD packets are scheduled earlier than their OPT mates, therefore it follows that $x \geq y$. Thus, the number of unmatched packets in OPT memory at the end of time slot t is bounded by $M-x$, which is at most the number of unmatched packets in the memory of LQD, that is $M-y$. □

The following corollary states that our matching process always finds a unmatched LQD packet, when needed.

COROLLARY 3.5. *The matching process never fails.*

PROOF. There exist the three possible cases for matching between LQD packet p and OPT packet p'. (1) The first case is that p arrives at time t' when LQD memory is not full and p is matched to itself. (2) The second case is that p at time t' is for the first time since its arrival at a higher location than the length of the queue of the p's output port in LQD and p is matched to some packet in LQD. (3) The third case is when LQD preempts at time t' some packet $\hat{p}$, which is matched to p, and accepts p'. From the three cases of the matching, the only problematic one is when an OPT packet is matched to some unmatched packet in LQD. However by Lemma 3.4 it follows that there is at least one unmatched packet in LQD memory. □

The next Corollary claims that all the extra packets are matched.

COROLLARY 3.6. *All the extra packets are matched.*

PROOF. Consider an extra packet sent at time t. At this time it clearly satisfies condition (a) of the matching. Since by Corollary 3.5 the matching never fails, it had to have been matched before. □

Now we upper bound the competitive ratio of LQD.

THEOREM 3.7. *The competitive ratio of the Longest Queue Drop policy is at most 2.*

PROOF. For any sequence of packets S, packets transmitted by OPT may be divided to packets transmitted by both OPT and LQD and extra packets transmitted only by OPT. According to Corollary 3.6 all the extra packets are matched. This implies that the number of extra packets is bounded by the number of packets that LQD sends, and therefor, $V_{OPT}(S) \leq 2V_{LQD}(S)$. □

We now improve the lower bound on the competitive ratio specifically for the LQD policy, and show that it is at least $\sqrt{2}$ (compared to 4/3 for any online policy).

THEOREM 3.8. *The competitive ratio of the Longest Queue Drop policy is at least $\sqrt{2}$.*

PROOF. Let us assume that the switch memory equal $M = A^2/2 + A$ and the number of output ports is $3A$ [2]. We divide the switch output ports into 3 classes: Active, Overloaded, and Idle. The number of active output ports is A. Active output ports have an average load of 1, and this load is periodic and very bursty. The period length is A and it starts with a burst of A packets that arrive in a single time slot, then there are no arrivals for $A-1$ time slots, then just as the queue is about to be empty, the next period start with a burst of A packets. The bursts to successive

[2]The assumption that $M \approx N^2$ is impractical, however, the scenario may be easily extended to more realistic setup with $M \gg N$.

active output ports are evenly staggered in time, and each time slot only one active output port starts a period (and receives a burst). OPT will receive all the packets destined to active output ports. The total queue length, summed over all these active queues, will be constant over time and approximately equals $A^2/2$.

The number of overloaded output ports is also A. Overloaded output ports receive exactly 2 packets every time slot. The optimal policy admits only 1 of these 2 packets in each time slot, and maintains a negligible queue length (of 1 packet) for each overloaded output port. The rest of the output ports are idle.

The optimal policy ensures that both the active and overloaded output ports are completely utilized, i.e., sends a packet each time slot, and its throughput at each time slot would be exactly $2A$.

In contrast, LQD allows the overloaded output ports to take up a substantial amount of memory space. Assume that the maximum queue length in LQD is xA, this implies that the overload ports will have xA^2 queued packets. By definition of LQD the active output ports will accept only xA packets each time. Since active output ports are drained at a linear rate, the total memory space of the active output ports is approximately $x^2A^2/2$. Assuming that the memory size is roughly $A^2/2$ (which hold for large value of A) we get that $x = \sqrt{2} - 1$. Hence, the throughput of the active output ports is $(\sqrt{2}-1)A$, and the total system throughput of LQD is $(\sqrt{2}-1)A + A = \sqrt{2} \cdot A$. Therefore, the competitive ratio of LQD is at least $(2 \cdot A)/(\sqrt{2} \cdot A) = \sqrt{2}$. □

3.2 Variable Size Packets

In this section we consider variable size packets. We show that the competitive ratio of LQD is at most $2 + \frac{NC_{max}}{M - NC_{max}}$, which is close to 2 for sufficiently large M. Now instead of extra packets, we define extra bytes that OPT sends and LQD does not send on the same output port at the same time.

DEFINITION 3.2. *A byte transmitted by OPT at time slot t on output port l is an* **extra** *byte if at time t the output port l of LQD was idle.*

When discussing variable packet size, the queue length of LQD in time of overflow may vary up to C_{max} bytes. In spirit similar to the previous section, we construct a fractional matching between extra bytes transmitted by OPT and bytes sent by LQD so that each extra byte is matched to a $\frac{M-NC_{max}}{M}$ fraction of a byte transmitted by LQD.

Matching: We iteratively construct a matching between OPT and LQD bytes as follows:

a. Each time slot t for each unmatched byte b that belong to OPT packet p in the queue to output port i:

If $l(b,t)$ in OPT is greater than the length of the queue for output port i in LQD, then we match OPT byte b as follows:

If p arrives at the current time slot and it is accepted by both OPT and LQD and the LQD memory is not full, then match OPT byte b to $\frac{M-NC_{max}}{M}$ fraction of itself in LQD.

Otherwise, match OPT byte b to $\frac{M-NC_{max}}{M}$ fraction of eligible (with smaller latency) unmatched bytes in LQD memory.

Else do nothing.

b. If a matched LQD byte b is preempted, then its OPT mate is unmatched and it is re-matched at the end of the time slot by the matching procedure, if necessary.

The proof of the following theorem is similar to that of the fixed packet size and is omitted.

THEOREM 3.9. *The competitive ratio of Longest Queue Drop policy is at most $2 + \frac{NC_{max}}{M-NC_{max}}$.*

4. VARIABLE VALUE PACKETS

In this section we consider variable value packets and the goal of the scheduling policy is to maximize the value of packets transmitted.

4.1 Fixed Size Packets

In this sub-section we consider fixed size variable value packets. We present a randomized scheduling policy achieving a competitive ratio of $4 \log \alpha$, where α is the ratio between the maximum and the minimum values.

The main idea is to divide packets into classes of almost equal values. The randomized policy selects one of the classes and schedules only packets that belong to that class. In particular, we prove that this policy is 4 competitive for the case of two possible packet values.

Randomized Partition Policy (RP): Divide packets into K classes of values so that the ratio between the maximal and the minimal values within the class is bounded by R. At time of initialization, choose uniformly at random one of these classes and schedule only the packets whose value falls in this class using LQD. All other packets are simply dropped.

THEOREM 4.1. *The competitive ratio of the Randomized Partition Policy is at most $2RK$.*

PROOF. Let S_D denote the subsequence of packets that belong to the class chosen by the RP policy. The expected value of $V_{OPT}(S_D)$ is at least $V_{OPT}(S)/K$. By Theorem 3.7 the RP policy schedules at least half the number of packets as OPT does. In addition, each packet from the selected class that is scheduled by OPT is at most R times more valuable than a packet scheduled by the RP policy, therefore, $V_{RP(S)} \geq V_{OPT}(S)/(2KR)$, which completes the proof of the theorem. □

To show that in the case of two possible packet values the competitive ratio is 4, we simply substitute $K = 2$ and $R = 1$ to the competitive ratio of derived in Theorem 4.1. For the general case where a packet may be assigned a value from interval $[1, \alpha]$ the values are divided into $K = \log\alpha$ classes: $[1, 2)$, $[2, 4)$, …, $[\alpha/2, \alpha]$, where the ratio between the maximal and the minimal values within each class is $R = 2$. Therefore the competitive ratio is at most $4 \log \alpha$.

4.2 Variable Size Packets

In this section we consider variable size and value packets. We modify the Randomized Partition Policy to work with variable size packets and show that it is $4\log\alpha + \frac{2NC_{max}\log\alpha}{M-NC_{max}}$ competitive. The following theorem follows directly from Theorems 3.9 and 4.1.

THEOREM 4.2. *The competitive ratio of the Randomized Partition policy is at most* $2RK + \frac{RKNC_{max}}{M-NC_{max}}$ *when packets are divided into* K *classes of values so that the ratio between the maximal and the minimal values within each class is bounded by* R.

5. CONCLUDING REMARKS

In this paper we study the performance of shared memory management policies for packet switches using competitive analysis. We consider Best Effort and DiffServ (QoS) models in the context of ATM and IP networks. We establish the competitiveness of the well-known Longest Queue Drop (LQD) policy for Best Effort model and show that its performance is near optimal. Then we consider the DiffServ model which introduces variable values, and present a randomized policy whose competitive ratio depends logarithmically on α, which is the ratio between the maximal and the minimal values. We extend our results from fixed size packets to variable size packets obtaining only slightly worse results.

An open question is whether for DiffServ model (variable values) it is possible to achieve competitiveness which is independent of α, or whether the lower bound can be improved. Another interesting future direction is to investigate the behavior of a network of switching elements and try and derive competitive analysis results for that setting.

Acknowledgments. The authors are very grateful to William Aiello for starting us with the line and providing essential contribution to this reserach.

6. REFERENCES

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

[2] M. Arpaci and J. A. Copeland, "Buffer Management for Shared-Memory ATM Switches," *IEEE Communications Surveys*, First Quarter 2000.

[3] B. Awerbuch and Y. Azar and A. Fiat and S. Plotkin and O. Waarts, "On-line load balancing with applications to machine scheduling and virtual circuit routing," *Proc. of the 23rd ACM Symp. on Theory of Computing*, pp. 623-631, 1993.

[4] D. Black and S. Blake and M. Carlson and E. Davies and Z. Wang and W. Weiss, "An Architecture for Differentiated Services," *Internet RFC 2475*, December 1998.

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

[6] A. K. Choudhury and E. L. Hahne, "Dynamic Queue Length Thresholds for Shared-Memory Packet Switches," *IEEE/ACM Trans. Commun.*, vol. 6, no. 2, Apr. 1998, pp. 130-40.

[7] M. Irland, "Buffer Management in a Packet Switch," *IEEE Trans. Commun.*, vol. COM-26, no. 3, Mar. 1978, pp. 328-37.

[8] F. Kamoun and L. Kleinrock, "Analysis of Shared Finite Storage in a Computer Network Node Environment under General Traffic Conditions," *IEEE Trans. Commun.*, vol. COM-28, no. 7, July 1980, pp. 992-1003.

[9] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber and M. Sviridenko, "Buffer Overflow Management in QoS Switches," to appear in *STOC 2001*.

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

[11] H. Kroner, "Comperative Peformance Study of Space Priority Mechanisms for ATM Networks," *Proc. IEEE INFOCOM '90*, June 1990, pp. 1136-43.

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

[13] K. Nichols, V. Jacobson and L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet," *Internet draft*, July 1999.

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

[15] A. K. Thareja and A. K. Agrawala, "On the Design of Optimal Policy for Sharing Finite Buffers," *IEEE Trans. Commun.*, vol. COM-32, no. 6, June 1984, pp. 737-40.

[16] S. X. Wei, E. J. Coyle, and M. T. Hsiao, "An Optimal Buffer Management Policy for High-Performance Packet Switching," *Proc. IEEE GLOBECOM '91*, vol. 2, Dec. 1991, pp. 924-28.