Efficient Buffer Sharing in Shared Memory ATM Systems With Space Priority Traffic

Rajarshi Roy and Shivendra S. Panwar, *Senior Member, IEEE*

***Abstract*—In this letter we study the problem of the optimal design of buffer management policies within the class of *pushout* and *expelling* policies for a shared memory asynchronous transfer mode (ATM) switch or demultiplexer fed by traffic containing two different space priorities. A numerical study of the optimal policies for small buffer sizes is used to help design heuristics applicable to large buffer sizes. Simulation studies for large buffer systems are then presented.**

***Index Terms*—ATM, buffer management, Markov decision theory, sample path techniques, shared memory switch.**

I. INTRODUCTION

MOST of the asynchronous transfer mode (ATM) switch architectures that have been proposed in the literature use some buffering to queue cells whose service has been delayed due to contention for resources within the switch. The location of these buffers and the buffer management policy affects the switch performance [6]. Some buffer management is necessary to ensure that an output-queued shared-memory switch or demultiplexer, both of which have multiple output ports each with its own queue, does not perform poorly when some of the output queues get overloaded.

Let us describe the system model and the two different classes of buffer management schemes we will consider. The shared memory switch (Fig. 1) is modeled by a multiserver queue with a bounded buffer. The entire buffer is partitioned into two parts: the main buffer of size B_M and the temporary buffer of size B_T. Time is slotted and the transmission of a cell takes one time slot. During one time slot at most B_T cells may arrive to the system and they are placed in the temporary buffer. The cell loss priority (CLP) bit indicates whether an ATM cell is of high or low priority. A cost is incurred for each cell that is dropped from the system. The cost is higher for the high priority cells. Once the cells are accepted to the main buffer, first-in–first-out (FIFO) order is to be maintained within each of the logical output queues, because cells in the same logical queue may belong to the same virtual circuit. For the same reason, in a switch model reordering of cells is allowed when they are transferred from the temporary buffer to the main buffer but not in the demultiplexer model. The objective is to minimize the long run average weighted cost that is incurred from the lost cells. Depending on the available control we have over the dropping of cells from the temporary or main buffer and the placement of cells in the main buffer, we distinguish two classes of policies, *pushout* and *expelling*, which are defined in later sections. The main contribution of this paper is to show that an *expelling* policy helps to increase the admissible load region compared to *pushout* policies.

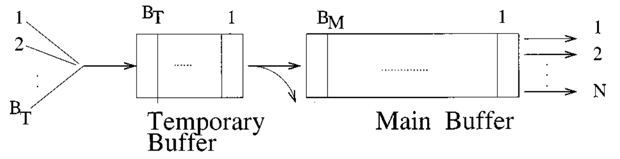

Fig. 1. The system model.

There has been a considerable amount of prior work in this area [1]–[5]. Our contribution differs from previous work in that it focuses on multiport devices, packets with priorities and considers broad policy classes.

II. PUSHOUT POLICIES

The class of *pushout* policies G_P is defined by the following rules: 1) a cell can be expelled from the main buffer only by another cell in the temporary buffer when the main buffer is full; 2) a cell from the temporary buffer can be discarded only if the main buffer is full; and 3) a cell can never be dropped if there is room for it in the main buffer.

The priority of a class is reflected by the cost C_h and C_l that is incurred by the dropping of a cell of high and low priority, respectively. We want to find out the policy π_1 amongst the class of *pushout* policies G_P so that

$$E_{\pi_1}\left(\sum_{t=0}^{\infty}(C_h \cdot D_h(t) + C_l \cdot D_l(t))\,|\,\text{any initial state } = x\right) \le E_{\pi_2}\left(\sum_{t=0}^{\infty}(C_h \cdot D_h(t)+C_l \cdot D_l(t))\,|\,\text{any initial state}=x\right) \quad \forall \pi_2 \in G_P.$$

Here, $D_h(t)$ and $D_l(t)$ are the number of high and low priority cells dropped from the system up to the end of slot $[t-1, t]$, respectively. C_h and C_l are positive real numbers, $C_h \ge C_l$. $E_{\pi_i}(\cdot)$ is the expectation when policy π_i is used. One can write with this value function the dynamic programming equations if the arrival statistics are completely characterized [9].

The optimal class of policies G_{P_1} is defined as the following. Suppose we are given the number of buffer positions that will

Manuscript received May 21, 2000. The associate editor coordinating the review of this letter and approving it for publication was Prof. D. Petr. This work was supported by the New York State Center for Advanced Technology in Telecommunications (CATT), Polytechnic University, Brooklyn, NY. This work was presented at the 33rd Conference on Information Science and Systems, March 1999.

R. Roy is with the CATT at Polytechnic University, Brooklyn, NY 11201 USA (e-mail: rroy@photon.poly.edu).

S. S. Panwar is with the Department of Electrical and Computer Engineering, Polytechnic University, Brooklyn, NY 11201 (e-mail: panwar@catt.poly.edu).

Publisher Item Identifier S 1089-7798(02)04493-9.

be allocated to a particular logical queue after the drop/pushout decision at a particular slot time in the main buffer is strictly less than the number of cells it had in the entire system just before the decision epoch t, i.e., at t^- a given number of cells are to be dropped/pushed out from each logical queue. Then the rules to be followed are as follows. 1) Append the cells that are in the temporary buffer and belong to logical queue n to the end of the main buffer in the allocated position, high priority cells first (if switch model) or in FIFO order (if demultiplexer model). 2) If the buffers allocated to logical queue n in the main buffer is full, and there are cells of that logical queue in the temporary buffer, push out the low-priority cells starting from those closest to the head of that logical queue. 3) If only high priority cells remain in a queue, discard all the remaining cells of that logical queue which are in the temporary buffer. This entire procedure defines the *squeeze-out* class of policies G_{P_1}.

The proof of optimality of the squeeze-out class is not given here due to space restrictions [8]. Note that we defined the optimal pushout policy structure but not the number of buffer positions allocated to each logical queue for a given system state.

III. Expelling Policies

The class of *expelling* policies G^E has as members all policies that append the new cells from the temporary buffer and do not reorder them, i.e., FIFO service order must be maintained. This class of policies is allowed to drop cells from the main buffer even when it is not full. We have proved that the optimal policy within the class G^E belongs to a subset of that class G^{EO}. The class G^{EO} is defined as the following. 1) Cells from each logical queue are moved from the temporary buffer to the portion of the main buffer allocated for it in that slot, either high priority cells first, or in the FIFO order, depending whether it is a switch model or a demultiplexer model, respectively. If they do not fit then low priority cells are expelled starting from those closest to the head of the queue. 2a) If the cell at the head of the queue is of high priority then it is served. 2b) If the cell at the head of the queue is of low priority then either that cell is served or all the low priority cells from the head of the queue until the high priority cell closest to the head of the queue are expelled and that high priority cell is served.

The proof of optimality is not given here due to space restrictions [8]. In this case, we have defined the optimal policy with the exception of: 1) the buffer allocation for each logical queue for each state and 2) the decision of when to serve the head-of-line low priority cell or high priority cell.

IV. Simulation Study for a Large System

In our numerical study we consider a two-ported shared memory switch. The logical queues are fed by a Bernoulli arrival process. In [7] we reported that loss probability calculations using value iteration [10]. We showed that the expelling policy does allow us to improve the performance of the high priority class compared to pushout policies at the cost of some degradation in the performance of the low priority class. Motivated by that we next present the performance of these policies for a 8×8 shared memory switch with large buffers driven by on–off traffic.

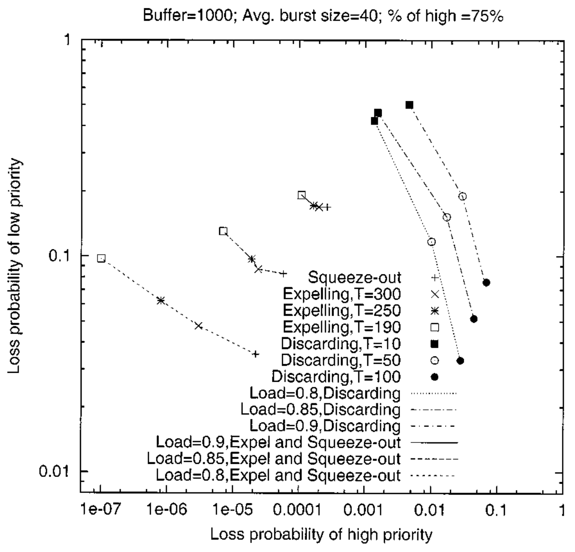

Fig. 2. High priority loss versus low priority loss.

In our simulation study we have a 8×8 switch with a shared 1000-cell buffer with each input line connected to a two-stage on-off source, with geometrically distributed on and off periods. Each burst is destined to one of the outputs. The idle period can be zero but each burst contains at least one cell. Cells were marked high or low priority probabilistically. For the squeeze-out scheme we use "squeeze-out the cell from the queue with a low priority cell that exceeded its threshold the most" policy. As we are yet to figure out a way to optimally assign the thresholds for each logical queue we use identical thresholds and our policy coincides with 'squeeze-out the cell from the longest queue with a low priority cell' policy.

We apply an heuristic expelling policy on each logical queue which expels low priority cells from the head of the logical queues when the number of high priority cells of that queue exceeded a threshold. We are able to achieve orders of magnitude improvement in the high priority loss performance at the cost of low priority loss performance. We also compared the performance of blocking or discarding policies as well. Here, once a cell is accepted it is always served. We set a threshold for total number of cells acceptable for each logical queue. The threshold on the number of cells in each logical queue is kept constant at 400 for the experiments in Fig. 2. Also, for each logical queue, we set another smaller threshold beyond which arriving low priority cells are discarded. This threshold is varied to achieve different loss performance. In Fig. 2 the letter 'T' beside the word "Expelling" indicates the expelling threshold and the same beside the word "Discarding" indicates the low priority cell blocking threshold in a discarding policy.

The merit of the expelling policy over the squeeze-out policy is the fact that the performance of low priority traffic can be traded off for the performance of high priority traffic. For example, assume a required maximum high priority loss of 10^{-5} and low priority loss of 0.7×10^{-2}. Under a squeeze-out policy for a buffer-size of 1000, an average burst length of 40, a load of 0.80 and with 75% of the total traffic of high priority, we do not achieve the required loss performance for the high priority cell but the low priority loss performance is achieved. If we want to

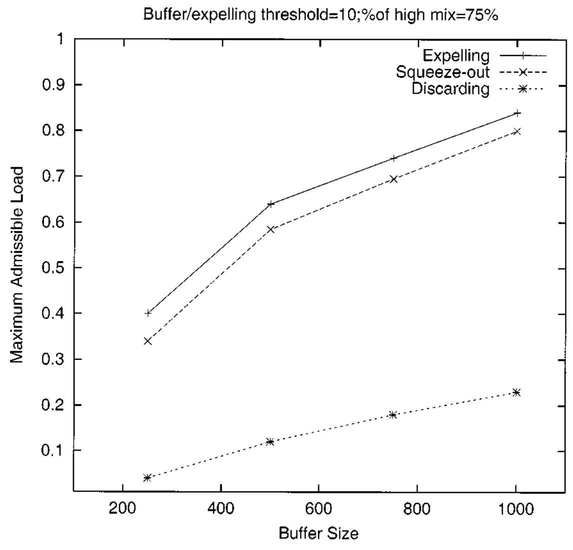

Fig. 3. Buffer size versus maximum admissible load.

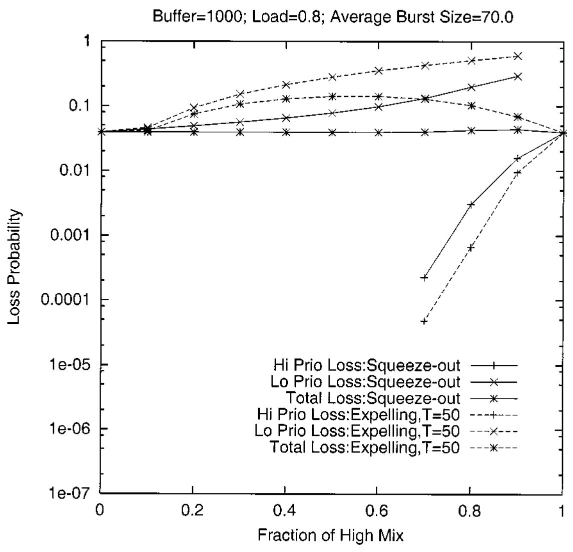

Fig. 4. Loss variation with traffic mix.

restrict ourselves to the this type of policy we have to increase the buffer size. Alternatively, we can use the expelling policy to achieve our goal. This is demonstrated in Fig. 2. This figure also shows that discarding (blocking) type of policies perform much worse in general. Apart from that, the policy of changing the low priority discarding threshold for the purpose of achieving varying loss requirements is not as flexible as the application of the expelling policy. Fig. 2 also demonstrates that in the low loss probability region the improvement due to the expelling policy is greater. Here, for the purpose of generating simulation data in a reasonable amount of time, we did not explore a more typical loss probability region (e.g., for which probability of loss for high priority traffic is less than 10^{-9} and probability of loss for low priority traffic is 10^{-6}). But we conjecture from the trend shown by the curves close to that region, which is the proposed operating region for ATM switches, the relative improvement of the expelling policy will be even more pronounced. The loss performance under all the schemes are very sensitive to the load and burst-size which is shown in [8, Figs. 2–5].

In Fig. 3 we have demonstrated the fact that under the expelling policy one can have a greater admissible load than a pushout policy of the "squeeze-out" class. Here the ratio between buffer size and the expelling threshold and that of buffer size and the threshold for low priority cell blocking in the discarding policy is always kept constant at 10 and 20, respectively. The average value of the burst size is 70, and the fraction of high priority traffic is 0.75. We performed a simulation based search to find out the maximum admissible load under both policies so that loss probability for high priority traffic is less than 10^{-3} and for low priority traffic is less than 0.4×10^{-1}.

Under the expelling policy, as the mix of high priority cell increases, high priority cells are less likely to get a low priority cell to pushout and that causes the high priority loss to increase. Low priority loss also increases because the expelling action gets triggered more often and the lack of low priority cells in the system increases the probability that a low priority cell will get pushed out (Fig. 4).

V. Conclusion

In this letter we present some properties of the optimum *pushout* and *expelling* policies. Our future goal is to develop heuristics for finding the various thresholds. Earlier numerical results for small buffer sizes and i.i.d. arrival process input indicated that the *expelling* policy can improve the high priority loss performance at the cost of low priority loss performance,thus allowing for more flexibility in meeting the cell loss constraints. Similar results are obtained from the simulation study for larger buffer sizes in this letter.

References

[1] 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, pp. 992–1003, July 1980.

[2] G. J. Foschini and B. Gopinath, "Sharing memory optimally," *IEEE Trans. Commun.*, vol. COM-31, pp. 352–360, Mar. 1983.

[3] S. X. Wei, E. J. Coyle, and M. T. Hsiao, "An optimal buffer management policy for high performance packet switching," in *IEEE GLOBECOM'91*, vol. 2, Dec. 1991, pp. 924–928.

[4] I. Cidon, L. Georgiadis, R. Guerin, and A. Khamisy, "Optimal buffer sharing," *IEEE J. Select. Areas Commun.*, vol. SAC-13, pp. 1229–1240, Sept. 1995.

[5] L. Tassiulas, Y. C. Hung, and S. S. Panwar, "Optimal buffer control during congestion in an ATM network node," *IEEE/ACM Trans. Networking*, vol. 2, pp. 374–386, Aug. 1994.

[6] M. J. Karol, M. J. Hluchyj, and S. P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.*, vol. COM-35, pp. 1347–1356, Dec. 1987.

[7] R. Roy and S. S. Panwar, "Optimal space priority policies for shared memory ATM system," in *Proc. 35th Annu. Allerton Conf. on Communication, Control and Computing*, Monticello, IL, Sept. 29–Oct. 1 1997, pp. 604–613.

[8] ——, "Optimal space priority policies for shared memory ATM system: Extended version," CATT, Polytechnic Univ., Brooklyn, NY, Tech. Rep., 2000.

[9] S. M. Ross, *Introduction to Stochastic Dynamic Programming*. New York: Academic, 1983.

[10] A. R. Odoni, "On finding the maximal gain for Markov decision processes," *Oper. Res.*, vol. 17, no. 5, pp. 857–860, Sept.–Oct. 1969.