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

Rajarshi Roy and Shivendra S. Panwar †
Center for Advanced Technology in Telecommunications
Polytechnic University
6 Metrotech Center
Brooklyn, NY 11201

March 11th, 1999

Abstract

In this paper we study the problem of the optimal design of buffer management policies for a shared memory ATM switch or demultiplexer fed by traffic containing two different space priorities. We present the properties of the optimal policy within the class of *pushout* and *expelling* policies that minimizes total weighted cell loss. A numerical study of the optimal policies for small buffer sizes based on the value iteration technique is used to help design heuristics applicable to large buffer sizes. Simulation studies for large buffer systems is then presented.

Keywords: Fast Packet Switching, Sample path techniques, Shared Memory Switch, Buffer Management, ATM.

1 Introduction

ATM (Asynchronous Transfer Mode) switch architectures use 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 fast packet 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. In such a case, without any control, the overloaded queue will consume most of the buffer spaces forcing the other lightly loaded queues to incur loss. In addition, cells marked as low priority may adversely effect the loss rate of high priority cells. This paper addresses both of these two related issues.

Let us next describe the system model and the two different classes of buffer management schemes we will consider. The shared memory switch (Fig. 1) is modelled 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 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 cells may belong to different traffic types. This assumption is consistent with the fact that a leaky bucket policed source may inject cells with different priority levels even if they belong to the same virtual circuit. A cost is incurred for each cell that is dropped from the system. The cost is higher for the high priority class. It is an N-ported, slotted system with cells destined to each output port or server constituting a logical queue. FIFO order is maintained within each of the logical queues once the cells are accepted to the main buffer because cells in the same logical queue may belong to the same virtual circuit. In the 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. This is because in a switch, due to the identical input line speeds, all the cells which came in the same time slot should belong to different virtual circuits, while that may not be the case in a demultiplexer model. Clearly, for a switch $N = B_T$, if each input line can inject at most one cell in every time slot. The value of B_T in case of a demultiplexer will be dependent on the input line speed. 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

*This research was supported by the New York State Center for Advanced Technology in Telecommunications, Polytechnic University.

†Tel: 718 260 3740, Fax: 718 260 3074, e-mail: panwar@kanchi.poly.edu

expelling. These classes will be defined in the next two 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 while satisfying the same cell level quality of service.

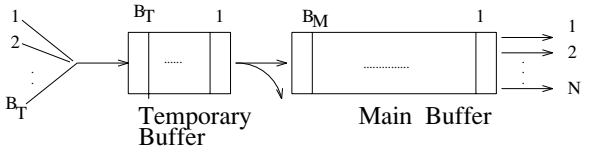

Figure 1: The system model

There has been a considerable amount of prior work in this area. The queueing analysis of different buffer sharing schemes and their relative merits was analyzed in [1]. The problem of designing optimal policies for the purpose of optimizing certain performance criteria is considered in [2]. They consider the problem of searching for optimal policies within the class where packets can only be blocked at the entry point of the buffer; dropping a packet once it is accepted in the switch is not allowed. In [3] *pushout* is allowed and arriving packets can push out packets from the longest logical queue. This policy turns out to be optimal only for a system with symmetric arrival and service processes. Cidon et al [4] considered a Poisson arrival, single loss priority and exponential service model for sharing memory in a switch with two output ports. They have established the optimality of a threshold based pushout scheme using Markov decision theory.

Hung et al in [5] provided optimal policies within the class of *discarding*, *pushout* and *expelling* policies using dynamic programming and sample path based techniques. They consider the case of general arrival models for the case of a single output port.

2 Pushout Policies

The class of *pushout* policies G_P is defined by the following rules: (a) A cell can be expelled from the main buffer only by another cell in the temporary buffer when the main buffer is full, (b) A cell from the temporary buffer can be discarded only if the main buffer is full, (c) A cell can never be dropped if there is room for it in the main buffer.

We will consider a two priority system. 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 among the class of *pushout* policies G_P so that
$E_{\pi_1}(\sum_{t=0}^{\infty}(C_h \cdot D_h(t)+C_l \cdot D_l(t))|\text{any initial state } = x) \leq E_{\pi_2}(\sum_{t=0}^{\infty}(C_h \cdot D_h(t)+C_l \cdot D_l(t))|\text{ any initial state} = x)$, $\forall \pi_2 \in G_P$. 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, $C_h \geq C_l$. $E_{\pi_i}(\cdot)$ is the expectation when policy π_i is used. One can write with this value function the stochastic dynamic programming equations if the arrival statistics are completely characterized [9].

The class of policies G_{P_1} is defined as the following: Suppose we are given the number of buffer positions that will be allocated to a particular logical queue after the drop/pushout decision 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 (a) 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), (b) 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, (c) 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}.

Theorem 1: When the systems start with the same initial conditions and under policies π_1 and π_2, where $\pi_1 \in G_{P_1}$ and $\pi_2 \in G_{P_1}^c \cap G_P$, and the arrivals are identical, then we have $D_h^{\pi_1}(t) \leq D_h^{\pi_2}(t)$ and $D_h^{\pi_1}(t)+D_l^{\pi_1}(t) \leq D_h^{\pi_2}(t)+D_l^{\pi_2}(t)$, $\forall t \in \{1,2,...\}$.

The proof of Theorem 1 is based on sample path method and is presented in [8]. It can be shown with modest algebraic manipulation that Theorem 1 is equivalent to proving the optimality of the class of policies G_{P_1}. Note that Theorem 1 defines the optimal pushout policy structure but not the number of buffer positions allocated to each logical queue for a given system state.

3 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.

In the following we will prove 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 mode, 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.

Theorem 2: For every policy $\pi \in G^E \cap G^{EO^c}$ there exists a policy $\pi_1 \in G^{EO}$ such that if the system starts from the same initial state under the two policies and the arrival process is identical under the two policies we have:
$D^{1h}(t) \leq D^h(t)$,
$D^{1h}(t) + D^{1l}(t) \leq D^h(t) + D^l(t)$, $t = 1, 2, ...$,
where $D^{1h}(t)$,$D^{1l}(t)$ is the number of cells lost of high and low priority, respectively, under π_1. $D^h(t)$ and $D^l(t)$ are similarly defined for π.

The proof of Theorem 2 is based on sample path method and is presented in [8]. In this case, Theorem 2 defines the optimal policy with the exception of (1) the buffer allocation for each logical queue for each state and (2) the decision on whether to serve the head-of-line low priority cell or high priority cell.

4 Numerical Study for a Small Buffer

We consider a two-ported shared memory switch modelled as a queue with bounded buffer and two servers each with a constant service rate of one cell/time slot. At any time slot at most two cells can arrive at the system. The arrival process is i.i.d. Bernoulli and the cells are destined to either of the two logical queues with probabilities b and $1-b$. Given the event that a cell belongs to logical queue 1, it can be of high or low priority with probability c and $1-c$, and given the event that it belongs to logical queue 2, it can be of high or low priority with probability d and $1-d$. Our system has six buffer places of which two are temporary buffer places. Small buffers were used because of the state space size limitations, when using the value iteration approach [10], which gives the optimal policy and the loss probabilities of different classes of cells. This is done by determining the optimum decisions left unspecified by Theorems 1 and 2.

The numerical results reveal that pushout from the longest queue policy when we have only either high or low priority cells in the buffer is not optimal for an unbalanced load. Rather, for a given load pattern the logical queues have thresholds k_1 and k_2, such that $k_1 + k_2 = B_M$. Thus if we need to drop cells we should check which logical queue has exceeded its threshold. If the new cells belong to that queue then they get dropped otherwise it pushes out cells from the other queue.

We study the expelling policy for the same system as described above. We confirmed that at certain states all the low priority cells at the head of a logical queue are dropped and the high priority cell is served. The decision of whether to serve the low priority cell or to drop them depends on the number of high priority cells of that logical queue in the system for most of the cases. However, we did see that there are some pairs of distinct states such that both have at least one low priority cell at the head of the queue and with the same number of high and low priority cells in the system, but the action chosen is different.

Loss probability calculations using value iteration shows that, as expected, for the same loss probability expelling policies can handle a higher load region than pushout policies. 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. These numerical results are reported extensively in [7]. Motivated by that we next investigate the performance of these policies for a 8x8 shared memory switch with a large buffer driven by on-off traffic.

5 Simulation Study with a Large Buffer

In our simulation study we modeled a 8x8 switch with each input line connected to a two-stage on-off source, and with geometrically distributed on and off periods. Each burst is destined to one of the outputs. Output queueing was employed and all logical output queues share the same buffer. The idle period can be zero but each burst contains at least one cell. Cells were marked high or low priority probabilistically. Simulation runs were set up so that the 95 percent confidence interval for cell losses is always less than 10 percent of the measured value.

For the pushout scheme we assign a threshold for each queue. If there are low priority cells in the main buffer, then among the queues which have them we select the one which has largest difference between its queue length and threshold for the squeeze-out process. If the differences are equal for some of the queues we pick one of those at random. If there is no low priority cell in the main buffer we do a similar sorting among all the queues. Incoming high priority cells then squeeze-out high priority cells. As we are yet to figure out a way to optimally assign the thresholds for each logical queue we use identical thresholds whose sum equals the buffer size. For the expelling scheme we use the same pushout policy while admitting the cells, but while serving the queue we apply an heuristic expelling policy on each logical queue where we expel low priority cells from the head of the logical queues when the number of high priority cells of that queue exceeds a threshold. The total loss is always greater than the squeeze-out policy in this scheme but remains of the same order. Here we are able to achieve orders of magnitude improvement in the high priority loss

performance at the cost of low priority loss performance. This is desirable since it increases the admissible load for certain high and low priority cell loss requirements. 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 the total number of acceptable cells for each logical queue. 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 performances. The threshold on the number of cells in each logical queue is kept constant at 400 for the experiments of Fig.2 and Fig.3. In Fig.2 and Fig.3 the letter 'T' beside the word "Expelling" indicates the expelling threshold and the same beside the word "Discarding" indicates low priority 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 value of probability of high priority loss of 10^{-5} and that of low priority loss of $0.7\text{x}10^{-2}$. 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 percent 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 (see Fig.3). If we want to restrict ourselves to the this type of policy we have to increase the buffer size or lower the load. Alternatively, we can use the expelling policy with $T = 250$ or 300 to achieve our goal. 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 blocking threshold for the purpose of achieving varying QOS is not as flexible as the application of the expelling policy. Figures 2 and 3 also demonstrate that in the low loss probability region the improvement due to the expelling policy is more prominent. Here, for the purpose of having 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 that in that region, which is the proposed operating region for ATM switches, the positive effect of the expelling policy will be even more pronounced.

In Fig.4 we demonstrate the fact that under the expelling policy one can have a greater admissible load than a pushout policy of the "squeeze-out" class and both do considerably better than discarding policies. 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 $0.4\text{x}10^{-1}$.

5.1 Variation of Performance with Traffic Parameters

The loss performance under all the schemes are very sensitive to the load and burst-size as is shown in Fig.5-7. We have noticed that if the ratio of burst-size to the buffer-size remains the same we achieve roughly the same loss probability for the same load as the buffer-size is varied under the squeeze-out policy. We have also noticed that if this ratio is kept the same and the ratio of expelling threshold to buffer-size is also kept same we get similar results under the expelling policy (Fig.8). We are investigating this rule of thumb type property further.

Under the expelling policy as the proportion of high priority cells increase, high priority cells are less likely to get a low priority cell to push out 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.7).

6 Conclusion

In this paper we present some properties of the optimum *pushout* and *expelling* policies. Our future goal is to develop heuristics for determining 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. Similar results are obtained from the simulation study for larger buffer sizes in this paper.

References

[1] F. Kamoun and L. Kleinrock, "Analysis of Shared Finite Storage in a Computer Network Node Environment Under General Traffic Conditions," *IEEE Trans. on Communications*, Vol. COM-28, No. 7, July 1980.

[2] G. J. Foschini and B. Gopinath, "Sharing Memory Optimally," *IEEE Trans. on Communications*, Vol. COM-31, No. 3, March 1983.

[3] S. X. Wei, E. J. Coyle and M. T. Hsiao, "An Optimal Buffer Management Policy for High Performance Packet Switching," *IEEE GLOBECOM'91*, Vol. 2, pp. 924-928, December, 1991.

[4] I. Cidon, L.Georgiadis, R. Guerin, A. Khamisy, "Optimal buffer sharing," *IEEE JSAC*, September 1995, vol 13, No. 7, pp. 1229-1240.

[5] L. Tassiulas, Y. C. Hung and S. S. Panwar, "Optimal Buffer Control During Congestion in an ATM Network Node," *IEEE/ACM Transactions on Networking*, August 1994, Vol. 2, No. 4, pp. 374-386.

[6] Mark J. Karol, Michael J. Hluchyj and Samuel P. Morgan, "Input Versus Output Queueing on a Space-Division Packet Switch," *IEEE Transactions on Communications*, December 1987, Vol. 35, No. 12, pp. 1347-1356.

[7] Rajarshi Roy and Shivendra S. Panwar, "Optimal Space Priority Policies for Shared Memory ATM System," *Proceedings of Thirty-Fifth Annual Allerton Conference on Communication, Control and Computing*, pp. 604-613, September-October, 1997.

[8] Rajarshi Roy and Shivendra S. Panwar, "Optimal Space Priority Policies for Shared Memory ATM System: Extended version", *CATT– Technical Report*, *Polytechnic University*, *Brooklyn*.

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

[10] A. R. Odoni, "On finding the maximal gain for Markov decision processes," *Operations Research*, 17(1969), 857-860.

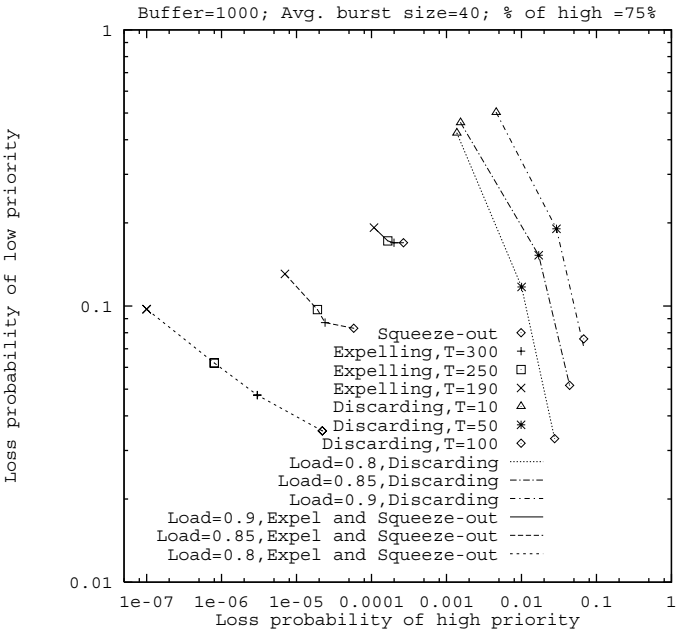

Figure 3: High priority loss versus low priority loss

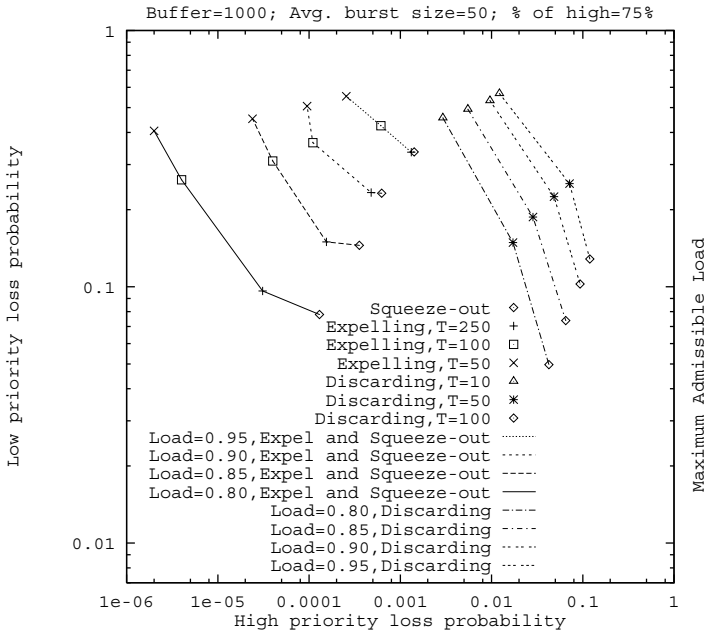

Figure 2: High priority loss versus low priority loss

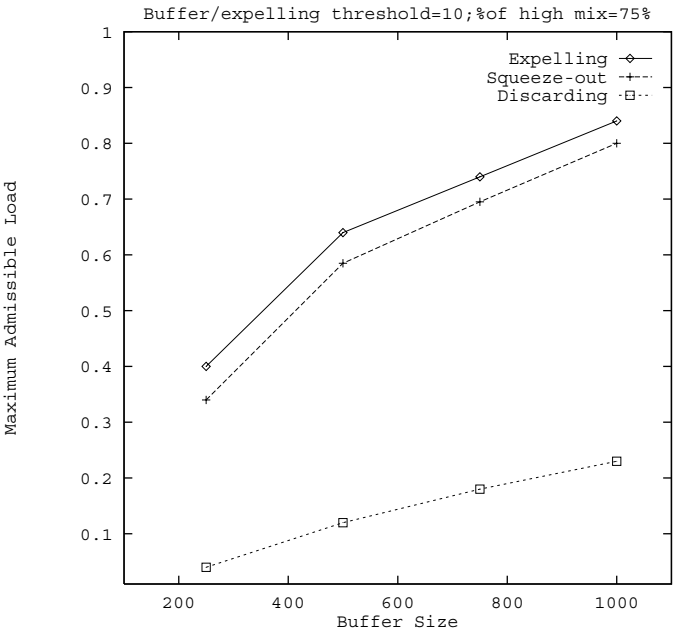

Figure 4: Buffer size versus maximum admissible load

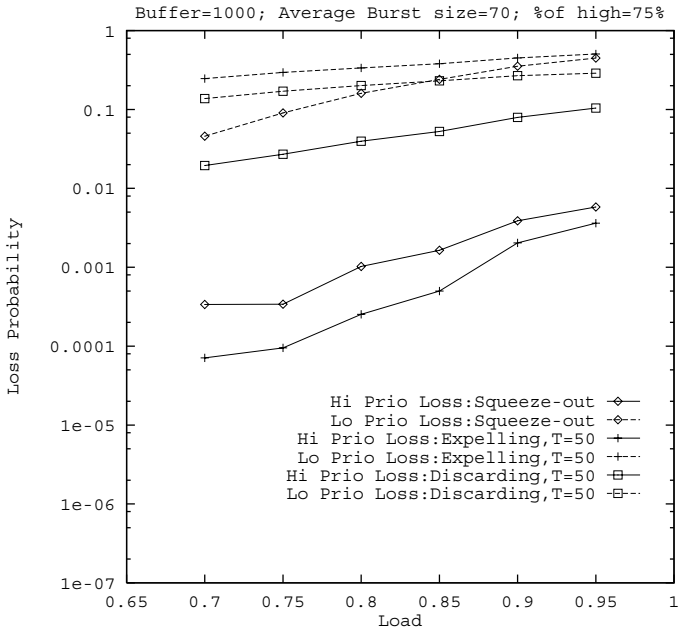

Figure 5: Loss variation with load

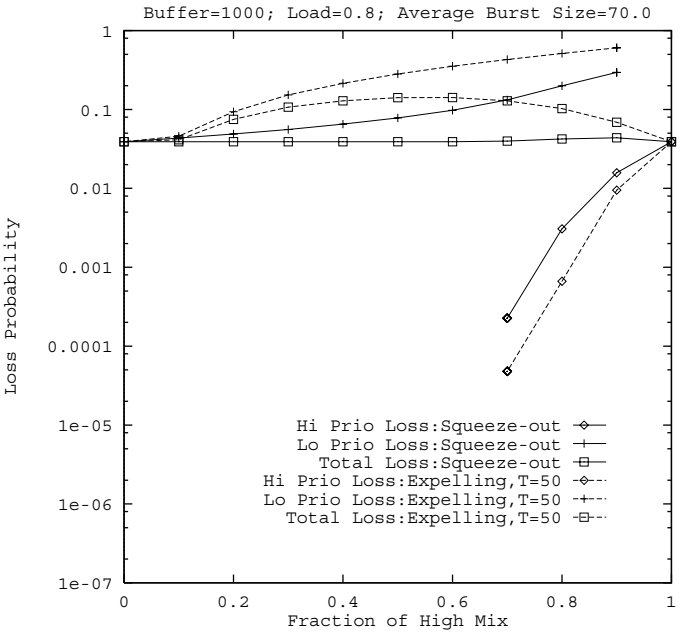

Figure 7: Loss variation with traffic mix

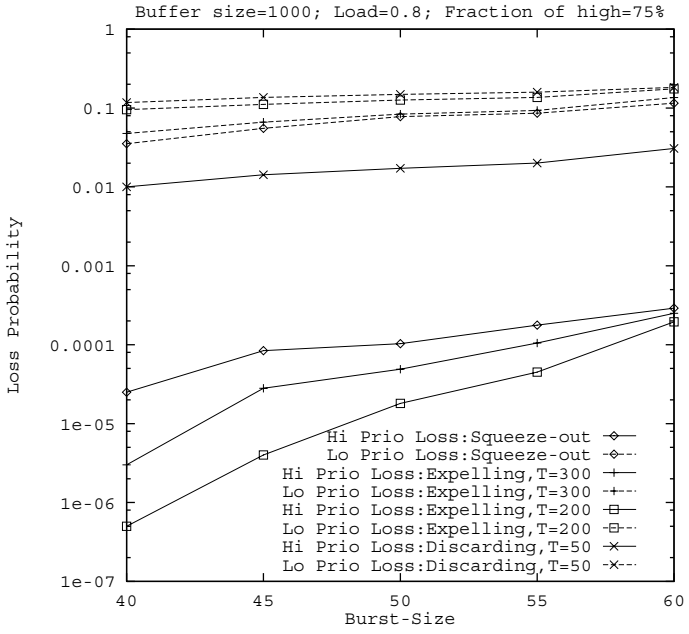

Figure 6: Loss variation with burst-size

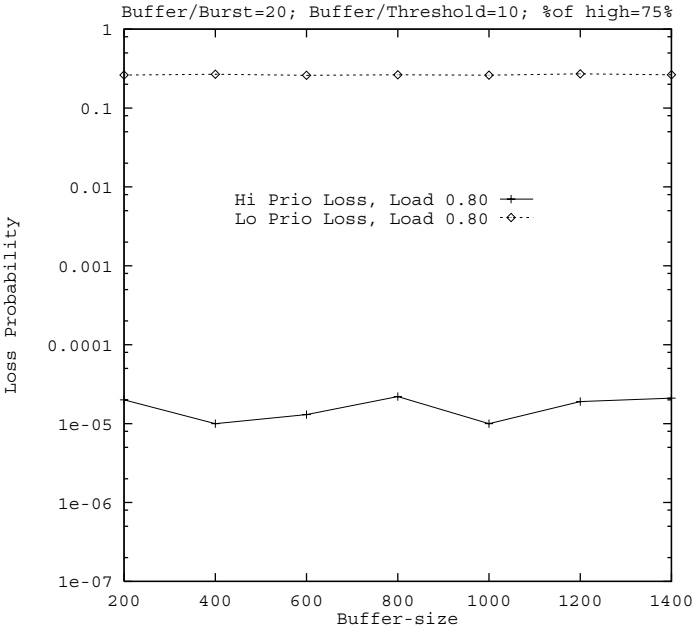

Figure 8: Verification of a conjecture