

# Dynamic Queue Length Thresholds for Shared-Memory Packet Switches

Abhijit K. Choudhury, *Member, IEEE*, and Ellen L. Hahne, *Member, IEEE*

**Abstract**—In shared-memory packet switches, buffer management schemes can improve overall loss performance, as well as fairness, by regulating the sharing of memory among the different output port queues. Of the conventional schemes, static threshold (ST) is simple but does not adapt to changing traffic conditions, while pushout (PO) is highly adaptive but difficult to implement. We propose a novel scheme called dynamic threshold (DT) that combines the simplicity of ST and the adaptivity of PO. The key idea is that the maximum permissible length, for any individual queue at any instant of time, is proportional to the unused buffering in the switch. A queue whose length equals or exceeds the current threshold value may accept no more arrivals. An analysis of the DT algorithm shows that a small amount of buffer space is (intentionally) left unallocated, and that the remaining buffer space becomes equally distributed among the active output queues. We use computer simulation to compare the loss performance of DT, ST, and PO. DT control is shown to be more robust to uncertainties and changes in traffic conditions than ST control.

**Index Terms**— Adaptive thresholds, asynchronous transfer mode, buffer allocation, dynamic thresholds, memory management, pushout, queue length thresholds, shared-memory switch.

## I. INTRODUCTION

MOST OF THE packet switch architectures that have been proposed in the literature use some buffering to accommodate packets whose service has been delayed due to contention for some resource within the switch. The location of these buffers and the buffer management policy directly affect the performance of such a switch. Switches based on input queueing, output queueing, and completely shared buffering have been proposed and studied [1]–[3]. It was concluded that output queueing and completely shared buffering achieve optimal throughput-delay performance.

However, output-queued shared-memory packet switches with no buffer management procedures may not perform well under overload conditions [4]. The problem is that a single output port can take over most of the memory, preventing packets destined for less utilized ports from gaining access. This causes the total switch throughput to drop. One solution to this problem is to place restrictions on the amount of buffering

a port can use. This makes buffers available to the less utilized ports and increases the total switch throughput.

We will now recall some well-known techniques for regulating port queue lengths, and then propose a new method. To simplify the presentation, we will describe these mechanisms in the context of asynchronous transfer mode (ATM) switches (i.e., fixed-length cell switches) with true output queueing. However, the same ideas can be applied to input-queued switches with virtual output queueing, to routers that internally segment packets into cells, and to variable-length packet switches and routers.

Two styles of buffer sharing restrictions appear in the literature. One type places limits on the maximum or minimum amount of buffering that should be available to any individual queue [4]–[7]. Our paper considers a version of this approach called the *static threshold* (ST) scheme. In this method, an arriving cell is admitted only if the queue length at its destination output port is smaller than a given threshold. The ST strategy is very simple to implement. It requires only queue length counters, which are likely to be needed for network management purposes anyway, and a comparator. Although traffic priority classes are beyond the scope of this paper, note that it is trivial to extend the ST scheme to multiple space priority (i.e., loss priority) classes—one can simply use different thresholds for different classes. However, we will see in Section IV that the performance of this scheme suffers from the fact that it is not adaptive. When so many queues are active at once that the sum of their thresholds exceeds the buffer capacity, then it is possible for the buffer to fill up completely even though all queues are obeying their threshold constraints. This allows some queues to become starved for space, which can lead to underutilization of the switch. At other times, when very few output queues are active, these queues are needlessly denied access to the idle buffer space “beyond” the sum of their thresholds. This creates higher cell loss rates and lower throughputs for these active queues than they would experience if they had access to extra buffer space.

In the second style of buffer management, arriving cells are allowed to enter the buffer as long as there is space and when the buffer fills up, an incoming cell is allowed to enter by selectively overwriting another cell that is already in the buffer [8]–[12]. In this paper we consider a version, simply called *pushout* (PO), in which a cell that arrives to find the buffer full pushes out the cell at the head of the longest queue. While the incoming cell usurps the physical space of the discarded cell, the incoming cell does not take over the discarded cell’s position in its logical output port

Manuscript received March 5, 1997; revised October 10, 1997; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor H. J. Chao. This paper was presented in part at INFOCOM’96, San Francisco, CA, March 26–28, 1996.

The authors are with Bell Laboratories, Lucent Technologies, Holmdel, NJ 07733 USA (e-mail: akc@bell-labs.com; hahne@bell-labs.com).

Publisher Item Identifier S 1063-6692(98)02671-5.

queue. Indeed, the pushing and pushed cells may belong to different output port queues. Rather, the arriving cell joins its own logical queue. The PO approach has many performance benefits. PO is fair—it allows smaller queues to increase in length at the expense of longer queues. PO is efficient—no output queue is ever starved for space, and no space is ever held idle while some queue desires more; thus, overall system throughput should be high. PO is naturally adaptive—when lots of queues are active, their rivalry keeps their queue lengths short; when only one queue is active, it is allowed to become long. It is difficult to implement PO, however, in high-speed switches. When the shared memory is full, writing a cell into a queue involves the extra step of first pushing out (in effect, reading out) another cell from a different queue, which could be any queue in the system. This puts a substantial amount of extra work in the critical path. Moreover, PO requires that the switch monitor not only the individual queue lengths but also the identity of the longest (or nearly longest [13]) queue. This operation consumes more time, and the switch is not likely to be doing it already for some other purpose. Furthermore, it is fairly difficult to implement PO for traffic with multiple space priority (i.e., loss priority) classes. Pushing out a low priority cell—locating such a cell in the middle of a queue, excising the cell, then mending the break in the queue—is not a trivial task [14].

A buffer management scheme with the simplicity of ST and the adaptivity of PO would be desirable. It would be better if this adaptivity were achieved without explicitly monitoring the fluctuating cell arrival rate for each individual queue, as is done in [15] and [16]. Therefore, in this paper we propose a novel scheme called *dynamic threshold* (DT) to fairly regulate the sharing of memory among different output queues. The approach is conceptually similar to bottleneck flow control [17] and to the bandwidth balancing mechanism in distributed queue dual bus (DQDB) networks [18]. The key idea is that the output queue length threshold, at any instant of time, is proportional to the current amount of unused buffering in the switch. Cell arrivals for an output port are blocked whenever the output port's queue length equals or exceeds the current threshold value. The DT method deliberately holds a small amount of buffer space in reserve, but distributes the remaining buffer space equally among the active output queues. The DT technique will be described and analyzed in more detail in Sections II and III.

Unlike the ST scheme, the DT method is adaptive and, hence, should be more efficient. Less buffer space should be wasted, and starvation incidents should be fewer and less severe. One would not expect the DT procedure to perform quite as well as PO, however, for the following two reasons. First, while both methods are adaptive, PO adapts faster—PO can retrieve cell buffers from a long output queue immediately, whereas DT can only reduce a queue's length by blocking new arrivals and waiting for the queue to drain naturally. Second, PO allows all of the memory to be used at all times, while DT holds some space in reserve at all times. For these two reasons, PO should perform somewhat better than DT. DT, however, is easier to implement. As with ST, queue length counters and a comparator are needed. The only additional requirement,

as explained in Section II, is a shift register. Moreover, the extension of DT's to multiple space priorities, as described in other papers [19], [20], is relatively straightforward and simple to implement. A much more detailed performance comparison is presented in Section IV. Conclusions appear in Section V.

## II. DYNAMIC THRESHOLD SCHEME

When only one switch output port is very active, we would like it to have access to as much of the shared buffer memory as possible. When there are many contending queues, however, we want to divide the memory fairly among them. All queues with sufficient traffic to warrant thresholding should obtain the same amount of space, called the *control threshold*. The control threshold value is determined by monitoring the total amount of unused buffer space.

Each output queue attempts to limit its length to some function  $f$  of the unused buffer space; output queues with less demand than this can have all the space they wish. At time  $t$ , let  $T(t)$  be the control threshold and let  $Q^i(t)$  be the length of queue  $i$ . Let  $Q(t)$  be the sum of all of the queue lengths, i.e., the total occupancy of the shared memory. Then, if  $B$  is the total buffer space

$$T(t) = f(B - Q(t)) = f\left(B - \sum_i Q^i(t)\right). \quad (1)$$

An arriving cell for queue  $i$  will be blocked at time  $t$  if  $Q^i(t) \geq T(t)$ . All cells going to this queue will be blocked until the queue length drains below the control threshold and/or the threshold rises above the queue length. The simplest scheme is to set the control threshold to a multiple  $\alpha$  of the unused buffer space

$$T(t) = \alpha \cdot (B - Q(t)) = \alpha \cdot \left(B - \sum_i Q^i(t)\right). \quad (2)$$

If  $\alpha$  is a power of two (either positive or negative), then the threshold computation is extremely easy to implement—only a shift register is required. For this reason, we only consider  $\alpha$  values that are powers of two in this paper.

The DT scheme adapts to changes in traffic conditions. Whenever the load changes, the system will go through a transient. For example, when a lightly loaded output port suddenly becomes very active, its queue will grow, the total buffer occupancy will increase, the control threshold will decrease, and queues exceeding the threshold will have their arrivals blocked temporarily while they drain, freeing up more cell buffers for the newly active queue. As we shall see in the next section, eventually all the very active queues, both old and new, will stabilize at equal lengths.<sup>1</sup>

When referring to steady-state values, we simply drop the time parameter  $t$  from the variable names. For instance,  $Q$  denotes the steady-state value of  $Q(t)$  and  $T$  denotes the

<sup>1</sup> Different queues will stabilize at different lengths if different values of  $\alpha$  are used. If each queue  $i$  uses its own value  $\alpha^i$  to compute its threshold, then the steady-state lengths of the controlled queues will be in the same relative proportions as their  $\alpha^i$  parameters. In particular, formula (4) below generalizes to  $Q^i = \alpha^i \cdot (B - \Omega) / (1 + \sum_{j \in C} \alpha^j)$ , where  $C$  is the set of controlled queues.

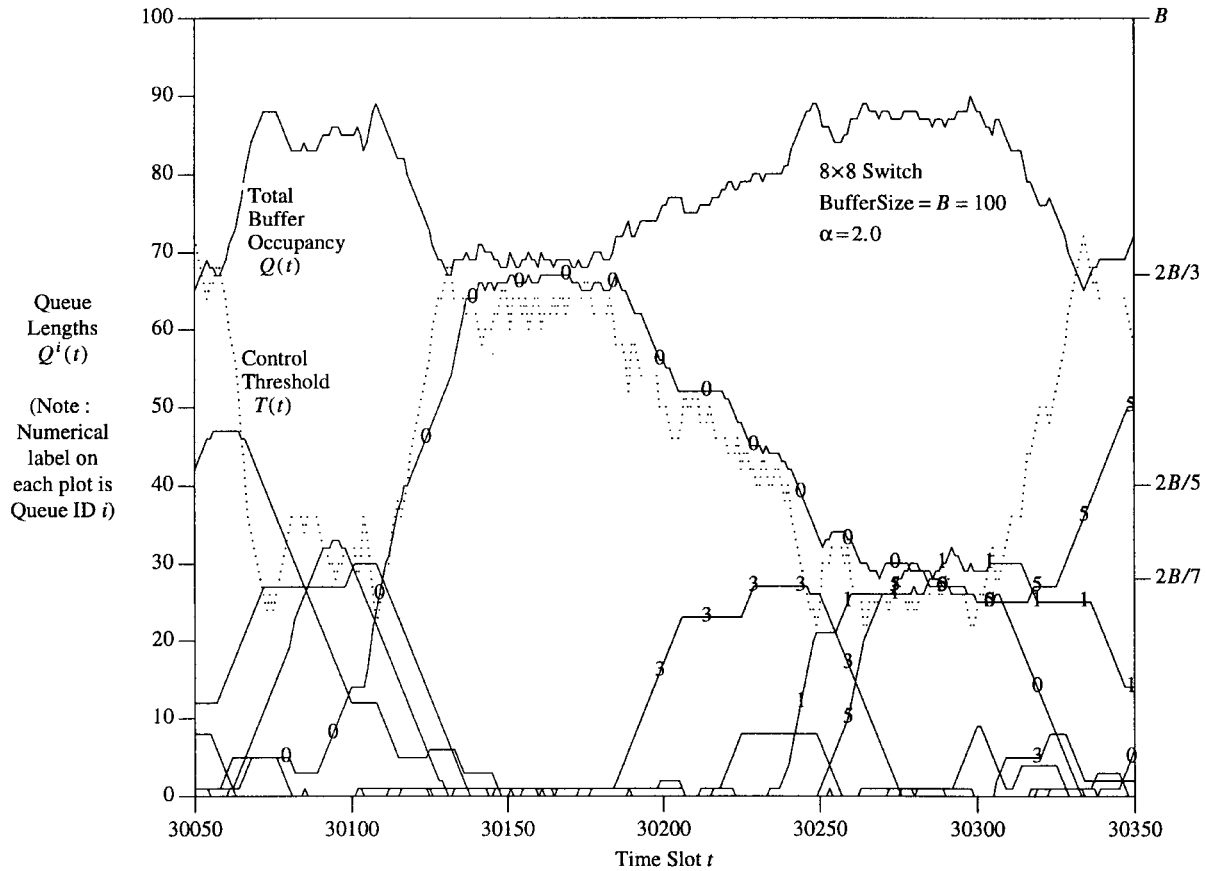


Fig. 1. Transient plot of queue lengths versus time.

steady-state value of  $T(t)$ . If there are  $S$  very active queues, then the total buffer occupancy  $Q$  in steady state will be

$$Q = S \cdot T + \Omega \quad (3)$$

where  $\Omega$  is the space occupied by the uncontrolled queues, i.e., those with lengths below the control threshold. The steady-state length of each controlled queue can be found by substituting (3) into (2) and solving for  $T$ .

$$Q^i = T = \frac{\alpha \cdot (B - \Omega)}{1 + \alpha \cdot S}. \quad (4)$$

The amount of memory  $\Theta$  held in reserve by the algorithm is as follows:

$$\Theta = \frac{(B - \Omega)}{1 + \alpha \cdot S}. \quad (5)$$

If  $\alpha = 2$ , for instance, the control algorithm tries to regulate each queue length to be twice the spare buffer capacity. So, a single queue with no competition is allowed to take  $2/3$  of the entire shared memory, and  $1/3$  of the memory is held back. When two long queues are active, each queue gets  $2B/5$ , and  $B/5$  is unallocated. If  $S$  then increases from two to ten, the long queues will drain and the newly active queues will grow until all ten stabilize at  $2B/21$ , with  $B/21$  left unallocated.

Our DT scheme deliberately wastes a small amount of buffer space. This “wastage” actually serves two useful functions. The first advantage of maintaining some spare space at all times is that it provides a cushion during transient periods

when an output queue first becomes active. This cushion, which will be illustrated in Section III (Fig. 3), reduces cell loss during such transients. Secondly, when an output queue has such a load increase and begins taking over some of the spare buffer space, this action signals the allocation mechanism that the load conditions have changed and that a threshold adjustment is now required. Without this built-in spare capacity, the cell arrival rates and/or loss rates of the individual output queues would have to be monitored to determine when load conditions had changed. Such a monitoring scheme [15], [16] would be a complex undertaking.

Note from (5) that the memory left unallocated in steady-state  $\Theta$  would be negligible if a very large value of the DT parameter  $\alpha$  were used. This would seem to be a good thing. However, if  $\alpha$  is too large, then the threshold  $T(t)$  tends to be too high and the control is ineffective—a single queue can consume too much memory for too long, causing other newly active queues to suffer too much cell loss. In Section IV-C (Figs. 9, 11, 13, and 15) we will demonstrate via simulation that extremely small and extremely large values of  $\alpha$  can cause excessive cell loss. Fortunately, though, the loss performance is fairly insensitive to the setting of  $\alpha$  as long as that value is moderate.

Fig. 1, which was generated by simulation, shows the DT scheme in action. The simulated system is an  $8 \times 8$  switch with a buffer size  $B = 100$  cells. The sources are of ON-OFF type, with geometric on and off periods. The figure plots various queue lengths versus time over a period of 300 time slots.

Included are the queue lengths of the eight output queues  $Q^i(t)$ ,  $i = 0, \dots, 7$ , the total buffer occupancy  $Q(t) = \sum_i Q^i(t)$ , and the control threshold  $T(t)$ . The numerical labels on the plots in the figure identify the various output queues. The DT parameter  $\alpha = 2.0$ . Near time 30 070,  $T(t)$  hovers around 30 due to the interaction of certain queues which then start to drain around  $t = 30\,100$ . This causes the total buffer occupancy  $Q(t)$  to decrease, raising the value of  $T(t)$ . Meanwhile, queue 0 has started to build and is allowed to increase because  $T(t)$  is increasing. Around  $t = 30\,140$ , queue 0 occupies about two-thirds of the total buffer, at which point the DT starts blocking arrivals to it. This is as predicted by (4) for  $S = 1$ . Around  $t = 30\,180$ , queue 3 starts to build up, pulling  $T(t)$  down, and this causes queue 0 to decline. A little later queues 1 and 5 also start building up, pulling  $T(t)$  even lower. Around  $t = 30\,270$ , queue 3 has drained out and queues 0, 1, and 5 are the only long queues. In accordance with (4) for  $S = 3$ ,  $T(t)$  settles around  $2B/7$ , permitting each of the three queues a length of roughly 28 cells. Note that there are several other queues that are very small during this period and are not controlled by the DT. After a while both queues 0 and 1 drain, allowing  $T(t)$  to rise again, and queue 5, which still has arrivals, is able to grow.

### III. TRANSIENT ANALYSIS

We now analyze the transient behavior of the DT scheme to demonstrate how the queue lengths converge to the fair steady-state allocation given by (4). While the simulations of Sections II and IV model discrete cells in discrete time, the analysis in this section assumes that time and queue lengths are continuous variables, that queue arrivals and departures are fluids, and that the threshold computations are instantaneous and ongoing.<sup>2</sup>

Consider a scenario where  $N$  active queues have lengths equal to their steady-state allocations when, at  $t = 0$ ,  $M$  empty dormant queues become active simultaneously.

$$T(0) = \frac{\alpha \cdot B}{1 + \alpha \cdot N} \quad (6)$$

$$Q^i(0) = \begin{cases} \frac{\alpha \cdot B}{1 + \alpha \cdot N}, & \text{for } 0 \leq i < N \\ 0, & \text{for } N \leq i < N + M. \end{cases} \quad (7)$$

We assume that each of these  $N + M$  queues now receives a smooth offered load of rate  $r > 1$ . (Flow rates are normalized to the service rate of an output port.) Using (2) and observing that a queue length's rate of change is bounded below by

<sup>2</sup>Of course, these assumptions are not strictly true. For one thing, queue lengths must be integers, so they are subject to roundoff errors, and the threshold must be an integral multiple of  $\alpha$ , so it cannot change smoothly either. Secondly, cells can only enter or leave queues at discrete time instants, as specified by a periodic schedule over the switch's input and output ports. Therefore, cells arriving simultaneously on different input ports may "see" somewhat different values of  $Q$  and, hence,  $T$ . Thirdly, the computation of  $T$  may require a substantial amount of time. It is possible to construct worst-case scenarios in which these real-world constraints result in significant errors, unfairness, and oscillations in the queue lengths. However, the ill effects can be mitigated by careful implementation—by interleaving input ports and output ports in the schedule and by computing  $T$  as often and as quickly as possible. Moreover, these artifacts can always be reduced by using a smaller value of  $\alpha$ .

$-1$  and bounded above by  $r - 1$ , we arrive at the following differential equations for  $t > 0$ :

$$\dot{T}(t) = -\alpha \cdot \sum_{i=0}^{N+M-1} \dot{Q}^i(t) \quad (8)$$

$$\dot{Q}^i(t) = \begin{cases} -1, & \text{if } Q^i(t) > T(t) \\ \max[-1, \min[\dot{T}(t), (r-1)]] , & \text{if } Q^i(t) = T(t) \\ r-1, & \text{if } Q^i(t) < T(t). \end{cases} \quad (9)$$

By evaluating (9) for  $t = 0^+$ , substituting that into (8), proving by contradiction that  $\dot{T}(0^+) \leq 0 < r - 1$ , and using that fact to further simplify the formula for  $\dot{T}(0^+)$ , we conclude that

$$\dot{T}(0^+) = -\alpha \cdot N \cdot \max[-1, \dot{T}(0^+)] - \alpha \cdot M \cdot (r - 1). \quad (10)$$

We now break the analysis into two cases, according to the value of the load  $r$ . As we shall see,  $r$  determines whether  $\dot{T}(0^+)$  is greater than or less than  $-1$ .

*Case 1:* Assume that

$$r \leq 1 + \frac{1 + \alpha \cdot N}{\alpha \cdot M}. \quad (11)$$

Using proof by contradiction, it follows from (10) and (11) that  $\dot{T}(0^+) \geq -1$ . This means that it is possible for the  $N$  old queues to track the dropping threshold by blocking a portion of their arrivals. Hence, (10) and (9) reduce to

$$\dot{T}(0^+) = -\frac{\alpha \cdot M \cdot (r - 1)}{1 + \alpha \cdot N} \quad (12)$$

$$\dot{Q}^i(0^+) = \begin{cases} -\frac{\alpha \cdot M \cdot (r - 1)}{1 + \alpha \cdot N}, & \text{for } 0 \leq i < N \\ r - 1, & \text{for } N \leq i < N + M. \end{cases} \quad (13)$$

By the same reasoning that led to (12) and (13), these differential equations will *continue* to hold as long as  $Q^i(t) = T(t)$  for  $0 \leq i < N$ , and  $Q^i(t) < T(t)$  for  $N \leq i < N + M$ . Solving the differential equations from initial conditions (6) and (7) yields

$$T(t) = \frac{\alpha}{1 + \alpha \cdot N} \cdot [B - M \cdot (r - 1) \cdot t] \quad (14)$$

$$Q^i(t) = \begin{cases} \frac{\alpha}{1 + \alpha \cdot N} \cdot [B - M \cdot (r - 1) \cdot t], & \text{for } 0 \leq i < N \\ (r - 1) \cdot t, & \text{for } N \leq i < N + M. \end{cases} \quad (15)$$

The trajectories of the queue lengths and the threshold are shown in Fig. 2. The solution (14), (15) is valid until the time  $t_1$  when the  $M$  new queues hit the threshold. Equating (14) and the second case of (15) for  $t = t_1$  and solving for  $t_1$  yields

$$t_1 = \frac{\alpha \cdot B}{[1 + \alpha \cdot (N + M)] \cdot (r - 1)}. \quad (16)$$

At  $t = t_1$ , all  $N + M$  queues reach the steady-state allocation predicted by (4)

$$Q^i(t_1) = T(t_1) = \frac{\alpha \cdot B}{1 + \alpha \cdot (N + M)}, \quad \text{for } 0 \leq i < N + M. \quad (17)$$

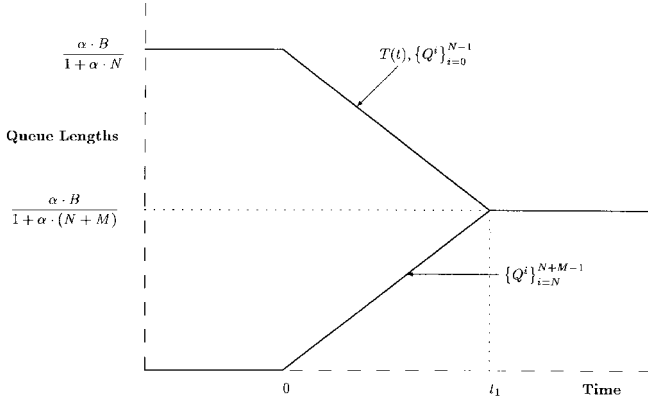


Fig. 2. Case 1:  $r \leq 1 + \frac{1+\alpha \cdot N}{\alpha \cdot M}$ .

Now let us consider  $t > t_1$ . By evaluating (9) for  $t = t_1^+$ , substituting that into (8), proving by contradiction that  $-1 < \dot{T}(t_1^+) < r - 1$ , and using that fact to further simplify the formulas, we conclude that

$$\dot{Q}^i(t_1^+) = \dot{T}(t_1^+) = 0, \quad \text{for } 0 \leq i < N + M. \quad (18)$$

The queue lengths and threshold have now reached equilibrium. For  $t > t_1$ , each queue maintains a constant length by being served at rate 1, admitting arrivals at rate 1, and rejecting excess arrivals at rate  $r - 1$ . This completes Case 1.

Case 2: Assume that

$$r > 1 + \frac{1 + \alpha \cdot N}{\alpha \cdot M}. \quad (19)$$

By first substituting (19) into (10), one can prove by contradiction that  $\dot{T}(0^+) < -1$ . This means that it is impossible for the  $N$  old queues to track the dropping threshold; even though they will block *all* their arrivals, they can only decline at rate  $-1$ . Hence, (10) and (9) reduce to

$$\dot{T}(0^+) = \alpha \cdot N - \alpha \cdot M \cdot (r - 1) \quad (20)$$

$$\dot{Q}^i(0^+) = \begin{cases} -1, & \text{for } 0 \leq i < N \\ r - 1, & \text{for } N \leq i < N + M. \end{cases} \quad (21)$$

The same reasoning that led to (20) and (21) can be used to show that these differential equations will *continue* to hold as long as  $Q^i(t) > T(t)$  for  $0 \leq i < N$ , and  $Q^i(t) < T(t)$  for  $N \leq i < N + M$ . Solving these differential equations from initial conditions (6) and (7) yields

$$T(t) = \frac{\alpha \cdot B}{1 + \alpha \cdot N} - \alpha \cdot [M \cdot (r - 1) - N] \cdot t \quad (22)$$

$$Q^i(t) = \begin{cases} \frac{\alpha \cdot B}{1 + \alpha \cdot N} - t, & \text{for } 0 \leq i < N \\ (r - 1) \cdot t, & \text{for } N \leq i < N + M. \end{cases} \quad (23)$$

The trajectories of the queue lengths and the threshold are shown in Fig. 3. The solution (22), (23) is valid for  $0 < t < t_2$ , where  $t_2$  is the time when the  $M$  new queues hit the threshold. In Fig. 3, the time interval  $(0, t_2)$  is called Phase 1. Equating

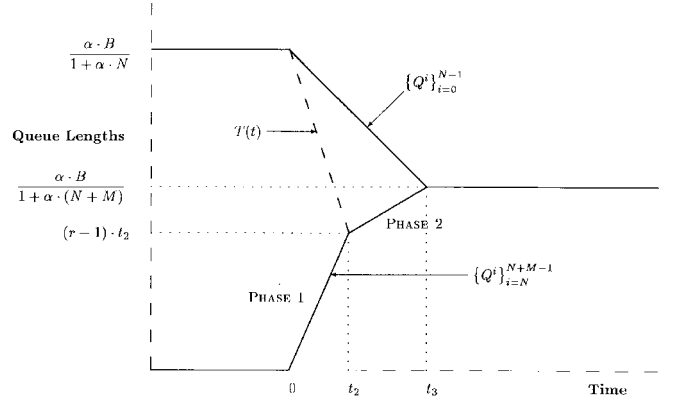


Fig. 3. Case 2:  $r > 1 + \frac{1+\alpha \cdot N}{\alpha \cdot M}$ .

the second case of (23) with (22) for  $t = t_2$  and solving yields

$$t_2 = \frac{\alpha \cdot B}{(1 + \alpha \cdot N) \cdot [(1 + \alpha \cdot M) \cdot (r - 1) - \alpha \cdot N]} \quad (24)$$

$$Q^i(t_2) = \begin{cases} \frac{\alpha \cdot B \cdot [(1 + \alpha \cdot M) \cdot (r - 1) - \alpha \cdot N - 1]}{(1 + \alpha \cdot N) \cdot [(1 + \alpha \cdot M) \cdot (r - 1) - \alpha \cdot N]}, & \text{for } 0 \leq i < N \\ \frac{\alpha \cdot B \cdot (r - 1)}{(1 + \alpha \cdot N) \cdot [(1 + \alpha \cdot M) \cdot (r - 1) - \alpha \cdot N]}, & \text{for } N \leq i < N + M \end{cases} \quad (25)$$

$$T(t_2) = \frac{\alpha \cdot B \cdot (r - 1)}{(1 + \alpha \cdot N) \cdot [(1 + \alpha \cdot M) \cdot (r - 1) - \alpha \cdot N]}. \quad (26)$$

Now the system enters Phase 2. By evaluating (9) for  $t = t_2^+$ , then substituting that into (8), one can prove by contradiction that  $\dot{T}(t_2^+) > 0 > -1$ . Then, using (19) and another proof by contradiction, one can show that  $\dot{T}(t_2^+) < r - 1$ . This means that it is possible for the  $M$  new queues to track the rising threshold by blocking a portion of their arrivals. Hence, (8) and (9) reduce to

$$\dot{Q}^i(t_2^+) = \begin{cases} -1, & \text{for } 0 \leq i < N \\ \frac{\alpha \cdot N}{1 + \alpha \cdot M}, & \text{for } N \leq i < N + M \end{cases} \quad (27)$$

$$\dot{T}(t_2^+) = \frac{\alpha \cdot N}{1 + \alpha \cdot M} \quad (28)$$

By the same reasoning that led to (27) and (28), these differential equations will *continue* to hold as long as  $Q^i(t) > T(t)$  for  $0 \leq i < N$ , and  $Q^i(t) = T(t)$  for  $N \leq i < N + M$ . Solving the differential equations from initial conditions (25) and (26) yields

$$Q^i(t) = \begin{cases} \frac{\alpha \cdot B}{1 + \alpha \cdot N} - t, & \text{for } 0 \leq i < N \\ \frac{\alpha}{1 + \alpha \cdot M} \cdot \left[ \frac{B}{1 + \alpha \cdot N} + N \cdot t \right], & \text{for } N \leq i < N + M \end{cases} \quad (29)$$

$$T(t) = \frac{\alpha}{1 + \alpha \cdot M} \cdot \left[ \frac{B}{1 + \alpha \cdot N} + N \cdot t \right]. \quad (30)$$

The solution (29), (30) is valid until the time  $t_3$  when the  $N$

old queues hit the threshold. Equating (30) and the first case of (29) for  $t = t_3$  and solving for  $t_3$  yields

$$t_3 = \frac{\alpha^2 \cdot M \cdot B}{(1 + \alpha \cdot N) \cdot [1 + \alpha \cdot (N + M)]}. \quad (31)$$

At  $t = t_3$ , all  $N + M$  queues reach the steady-state allocation predicted by (4).

$$Q^i(t_3) = T(t_3) = \frac{\alpha \cdot B}{1 + \alpha \cdot (N + M)}, \quad \text{for } 0 \leq i < N + M. \quad (32)$$

As before, we can evaluate (8) and (9) at  $t = t_3^+$  to show that the system stabilizes at this point. For  $t > t_3$ , each queue rejects just enough arriving traffic to maintain a constant length. This completes Case 2.

Note that Case 2 confirms our earlier claim that the transient performance for newly active queues can be poor if  $\alpha$  is too large. According to (24), for fixed  $r$ , as  $\alpha$  tends to infinity,  $t_2$  tends to zero. This means that Phase 1, the fast-growth phase for new queues, vanishes. The new queues are almost immediately in Phase 2, the slow-growth phase, where they are constrained by the threshold.

#### IV. PERFORMANCE

In this section we use simulation to compare the cell loss performance of the DT and ST strategies. To put the performance of these two schemes in perspective, we consider the performance of PO as well as the switch performance with No Control (NC) on queue lengths. One would expect the cell loss probability with NC to be an upper bound on loss for a properly tuned threshold scheme, and the losses with PO to be a lower bound on what is achievable with thresholds [8]–[12]. In all the plots in this paper, points are shown with 95% confidence intervals. (For many data points, the confidence intervals are so tiny that they are barely visible.)

##### A. Switch and Traffic Model

The architecture that we use for our simulation study is a single-stage switch with input and output ports running at identical rates. The switch uses output queueing and shared memory, i.e., each output port of the fabric has a logical queue, but these queues all share the same fabric memory. This memory can buffer  $B$  cells.

Each input line is connected to a single bursty source. The source alternates between active and idle periods. Cells arrive in consecutive time slots throughout each active period. The duration of the active period is geometrically distributed with parameter  $\alpha$ , while the idle period follows a shifted geometric distribution with parameter  $\beta$ ; we assume that there is at least one cell in each active burst, but the duration of an idle period can be zero. The burst lengths are assumed to be statistically independent. Given  $\alpha$  and  $\beta$ , then the mean burst length  $L_b$ , the mean idle time  $L_{\text{idle}}$ , and the normalized offered load  $\rho$  are given as

$$L_b = \frac{1}{\alpha} \quad L_{\text{idle}} = \frac{1 - \beta}{\beta} \quad \rho = \frac{L_b}{L_{\text{idle}} + L_b}. \quad (33)$$

For a given mean burst length  $L_b$ , the offered load  $\rho$  is varied by changing the mean idle duration between bursts.

In this paper we consider three traffic scenarios. In the uniform traffic case each burst from a source is equally likely to be destined to any of the output ports. The nonuniform traffic scenario includes two classes of output ports—moderately loaded and heavily loaded. The third traffic pattern is a hot-spot scenario, with a single overloaded output port and moderate loads at the other ports. For the nonuniform and hot-spot cases, bursts are more likely to be destined to a heavily loaded or overloaded output port than to a moderately loaded one. The burst length distribution is the same for all output ports, and all input ports carry identical loads. For the uniform load experiments, the metric we use is the overall cell loss probability. For nonuniform and hot-spot experiments, our metric is the loss probability for those cells destined to the moderately loaded output ports; we see these as the “good” ports that deserve to be protected from the other “bad” ports.

##### B. Robustness of the Four Schemes

In this subsection, we first consider a nominal switch configuration with dimensions  $512 \times 512$ , buffer size  $B = 6500$  cells, load  $\rho = 0.55$ , mean burst length  $L_b = 20$  cells, and uniform loading of the output ports.<sup>3</sup> We tuned the ST setting  $T$  and the DT factor  $\alpha$  to the values that minimize cell loss probability under the nominal load conditions: viz.,  $T = 120$ ,  $\alpha = 1.0$ .<sup>4</sup> For this load, the cell loss probability with NC is  $2.0 \times 10^{-2}$ . All three queue length control schemes do much better than that. The ST approach has only half that much loss, at  $1.0 \times 10^{-2}$ . DT does even better, at  $8.5 \times 10^{-3}$ . As expected, PO edges out all of the other schemes, with a cell loss probability of  $7.0 \times 10^{-3}$ .

Next we conduct four one-dimensional parameter variations around our nominal load conditions, without retuning  $T$  and  $\alpha$ , to see how robust the schemes are and to check their relative performance. In the first experiment we make one output queue a hot spot of varying intensity. In the next we add heavy but nonsaturating loads to a varying number of output ports. In the third and fourth investigations we vary the load  $\rho$  and the mean burst length  $L_b$ , respectively. Let us describe these experiments one at a time.

First consider the effect of a single overloaded output port on the cell loss for the 511 other ports, as shown in Fig. 4. Such an overload might happen because of outright errors or malfunctions in the ATM call admission and policing controls, or from a combination of loose controls and statistical bad

<sup>3</sup>The choice of this nominal scenario was a compromise between realism (the need for an accurate model) and reality (the need for measurable cell loss probabilities with tight confidence intervals in a finite amount of simulation time). Simulating lower cell loss probabilities would have been desirable but would have taken an impractically long time. When trying to draw conclusions for other switch sizes and traffic loads, the reader should bear in mind that the ratio of the buffer size to the average burst length is a critical parameter. We confirmed by simulation the intuitive notion that if the buffer size and the burst length (and, if applicable, the ST setting) are scaled up or down by the same amount, then the cell loss probability remains approximately the same.

<sup>4</sup>Since the nominal load in these simulations is uniform over all output ports, it is appropriate for ST to use the same  $T$  value at every queue, for DT to use the same  $\alpha$  value at every queue, and for PO to push out from the longest queue.

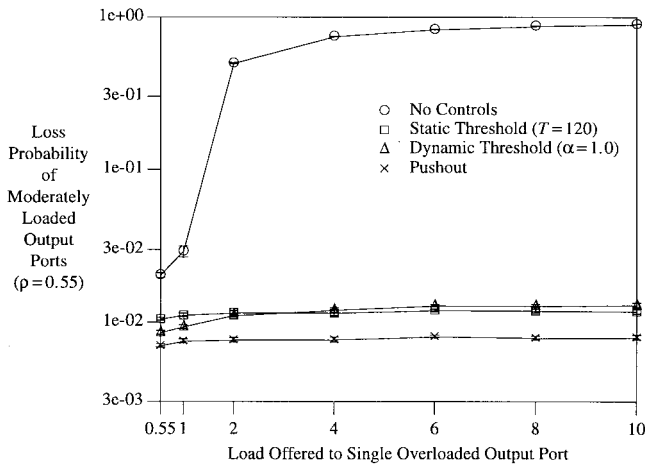


Fig. 4. Cell loss probability of moderately loaded output ports ( $\rho = 0.55$ ) versus load offered to a single overloaded output port, for various schemes.

luck. The figure plots the cell loss probability for the “good” ports, which are loaded at  $\rho = 0.55$ , as a function of the load to the “bad” port, as its load is varied from 0.55 to 10.00. This plot shows why some queue length control is necessary if there is any possibility of such an overload. The NC strategy is a disaster here, because the high loss probability of the overloaded port, which is unavoidable, is spread to the good ports. For example, when the hot spot has a load of 10.0 and, therefore, a cell loss probability of at least 90%, the good ports also suffer 90% losses. In other words, the total system throughput is only 10% of what it should be. Fortunately, all three queue length control schemes do *much* better than NC for this scenario, effectively fixing the problem. ST and DT have comparable performance. (Although it appears from the figure that ST is marginally better for extremely intense overloads, using a slightly different setting of  $\alpha$ , viz.,  $\alpha = 0.5$ , would have given DT the advantage at every point.) PO does somewhat better than ST and DT, having only about two-thirds as much loss. Note that all three control schemes give the good ports a loss performance that is nearly independent of the *magnitude* of the overload to the bad port.

Next we consider nonuniform traffic scenarios in which most of the 512 output ports are moderately loaded at  $\rho = 0.55$ , but some are heavily loaded at  $\rho = 0.95$ . Fig. 5 plots the cell loss probability for the moderately loaded ports as the number of heavily loaded ports varies from 0 to 128. The figure shows that, with all four strategies, as one would expect, the loss for the moderately loaded ports increases as more ports become heavily loaded. The ranking of the four schemes stays the same throughout—PO performs better than DT, which performs better than ST, which performs better than NC. However, the gap between DT and ST widens and the gap between ST and NC narrows as the number of heavily loaded ports increases. We offer the following (admittedly oversimplified) explanation based on (4) to help the reader understand why DT outperforms ST in this kind of situation. In (4), think of  $S$  as the number of heavily loaded ports, whose long queue lengths  $Q^i$  often hit the control threshold  $T$ , and think of  $\Omega$  as the total buffering consumed by the moderately loaded ports, whose queues are usually too short

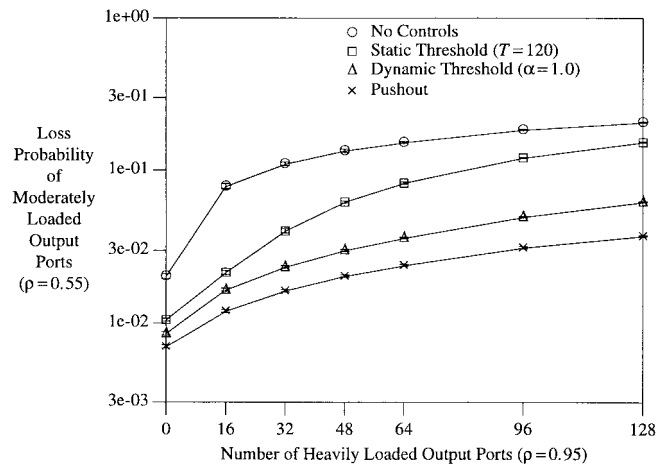


Fig. 5. Cell loss probability of moderately loaded output ports ( $\rho = 0.55$ ) versus number of heavily loaded output ports ( $\rho = 0.95$ ), for various schemes.

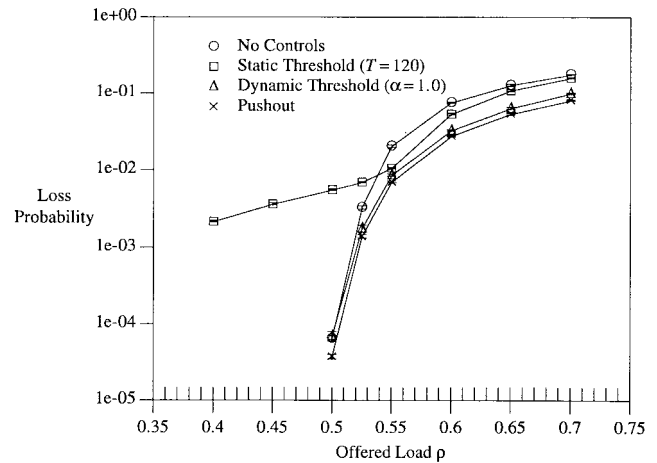


Fig. 6. Cell loss probability versus offered load  $\rho$ , for various schemes.

to be controlled. Equations (4) and (5) show that the controlled queues adapt their lengths as a function of  $S$  and  $\Omega$ , leaving a little spare buffering in case one of the moderately loaded ports gets the urge to expand a bit. With ST, on the other hand, queue lengths are allowed to reach  $T = 120$  regardless of the number of long queues. In the extreme case where 128 ports are heavily loaded, the ST scheme even permits the “bad” ports to squeeze out the “good” ports entirely, since  $B = 6500$  cells and  $120 \times 128 > 6500$ .

Now we consider uniform traffic, varying the load  $\rho$  over a wide range while fixing the mean burst length  $L_b$  at 20 cells. Fig. 6 shows the cell loss probability as a function of  $\rho$  for the four strategies. Recall that the ST setting  $T$  and the DT factor  $\alpha$  were tuned for the nominal load level  $\rho = 0.55$ . At this point, both threshold schemes perform well—while their losses are somewhat higher than PO’s, they are noticeably lower than NC’s. DT maintains this performance relative to NC and PO for higher loads, though for loads around  $\rho = 0.5$  DT’s performance becomes indistinguishable from NC’s. Unfortunately, ST does not perform much better than NC for higher loads, and for lower loads ST is dramatically worse—orders of magnitude worse—than any of the other

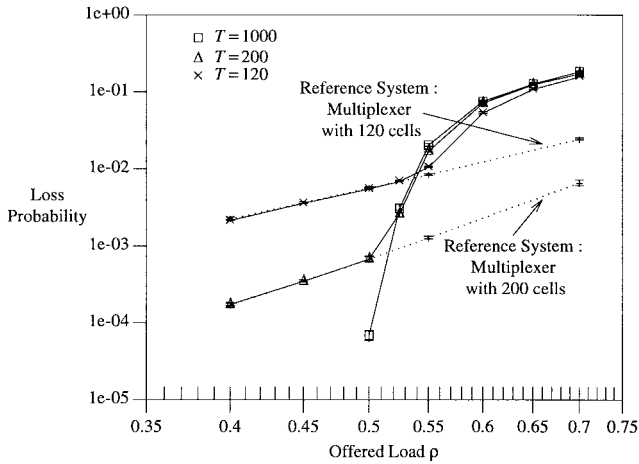


Fig. 7. Cell loss probability versus offered load  $\rho$ , for various ST settings  $T$ .

three strategies, including NC. As we shall see in Fig. 14, the reason for ST's abysmal performance at low loads is that the threshold setting  $T = 120$  is too low. With this threshold setting, essentially all the cell loss at low loads comes from individual queues hitting the threshold, and essentially no cell loss is due to the buffer becoming full.

Fig. 7 graphically illustrates these facts. The solid curves show loss versus load for our shared-memory switch under ST control. Each curve corresponds to a different setting of the threshold  $T$ . Notice how much better the low-load performance is when  $T$  is very large. The dotted curves are similar plots for a reference system comprising a multiple-input single-output switch, i.e., a multiplexer. The multiplexer models an individual output port queue of the switch in isolation. The buffer size in this multiplexer model corresponds to the ST value  $T$  in the full switch model. Simulation results for multiplexers with two different buffer sizes appear as the two dotted curves in the figure. When comparing a solid curve with its dotted counterpart, the dotted curve shows roughly how much of the solid curve's loss is attributable to the ST. As you can see, at low loads the threshold accounts for practically all the loss. If we could simulate extremely low load and loss levels, we would expect to see similar tails for the other three schemes in Fig. 6. For PO and NC we would expect the tail to correspond to a multiplexer of capacity  $B$ , since these two strategies allow an individual queue to take over the whole shared memory. For DT we would expect the tail to correspond to a multiplexer of capacity  $(\alpha \cdot B)/(1 + \alpha)$ , since this is the maximum length that an individual queue can ever attain with the DT scheme.

Finally we vary the mean burst length  $L_b$  over a wide range while fixing the load  $\rho$  at 0.55. (As  $L_b$  increases, the mean time between bursts is also increased, so that  $\rho$  stays constant.) Fig. 8 shows the cell loss probability as a function of  $L_b$  for the four strategies. The ST setting  $T$  and the DT factor  $\alpha$  were tuned for the nominal burst size  $L_b = 20$  cells. The shapes and relative positions of these curves are so similar to Fig. 6 that we will not repeat the details. As before, DT is much more robust than ST to this kind of traffic variation. ST does dramatically worse than NC for small burst sizes. As we shall see in Fig. 16, when the average burst size is

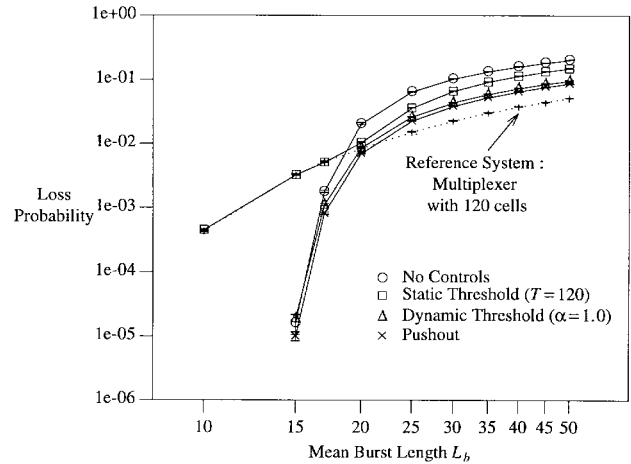


Fig. 8. Cell loss probability versus mean burst length  $L_b$ , for various schemes.

small, an ST setting of 120 cells is too low. Under these conditions, essentially all of scheme ST's loss comes from individual queues hitting the threshold, and practically no cell loss is due to the buffer becoming full. This claim is supported by the dotted curve, which shows the loss for a reference system comprising a single-queue multiplexer with 120 cells of buffering.

### C. Tuning the Two Threshold Schemes

Now let us take a more detailed look at the two threshold schemes. In this subsection we *will* retune the ST value  $T$  and the DT factor  $\alpha$  as we vary the load conditions around their nominal values. We want to see how the four schemes compare under different load conditions when optimal retuning of the control parameters is possible. More importantly, we want to see how close the loss is to this optimal value if you are restricted to one control parameter setting and *cannot* retune.

The figures for this subsection (Figs. 9–16) all have the same format. Each figure concentrates on either DT or ST. The  $y$ -axis shows the cell loss probability; in the case of nonuniform and hot-spot loads, only the loss for the “good” output ports is shown. The  $x$ -axis gives the control parameter setting, either  $\alpha$  or  $T$ . (Notice that the  $\alpha$  values we simulated are powers of two; these values can be implemented conveniently with a shift register.) Each figure includes three to four solid curves, where each curve shows the loss performance of the scheme for one specific traffic condition. Each curve gives the cell loss probability as a function of the control parameter setting for the given traffic. For comparison, the performance of PO is shown by three to four dashed lines on each plot, for the same three to four different traffic conditions as the threshold schemes. Since PO has no tunable parameters, the dashed lines are flat, not curved. Rather than using similar lines to depict the performance of NC, we note instead that, at the right-hand endpoint of any solid curve, the setting of  $T$  or  $\alpha$  is so high that the control is completely ineffective; the performance at these right-hand endpoints is the NC performance.

First, in Figs. 9 and 10 we consider various loads offered to a single overloaded output port. These plots, the least



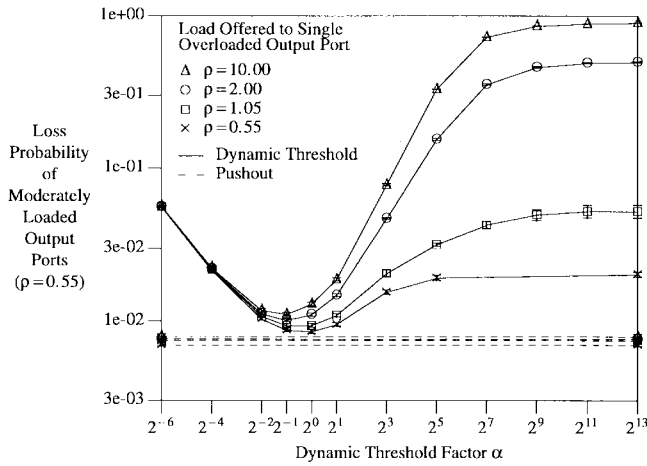


Fig. 9. Cell loss probability of moderately loaded output ports ( $\rho = 0.55$ ) versus DT factor  $\alpha$ , for various loads offered to a single overloaded output port.

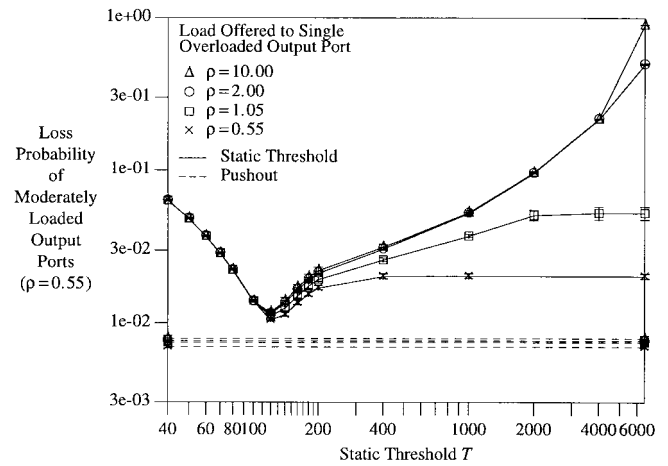


Fig. 10. Cell loss probability of moderately loaded output ports ( $\rho = 0.55$ ) versus ST setting  $T$ , for various loads offered to a single overloaded output port.

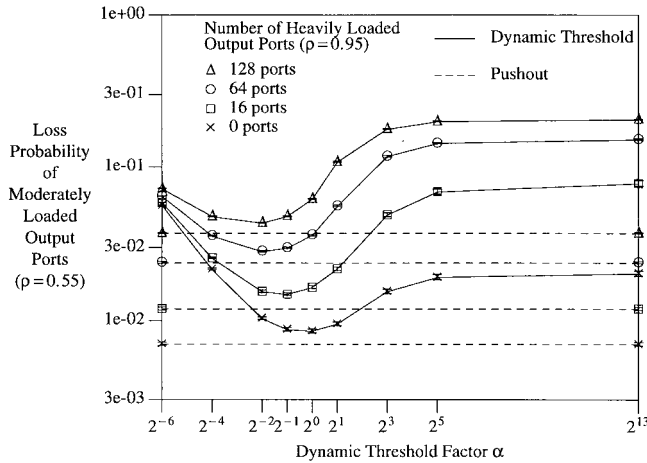


Fig. 11. Cell loss probability of moderately loaded output ports ( $\rho = 0.55$ ) versus DT factor  $\alpha$ , for various numbers of heavily loaded output ports ( $\rho = 0.95$ ).

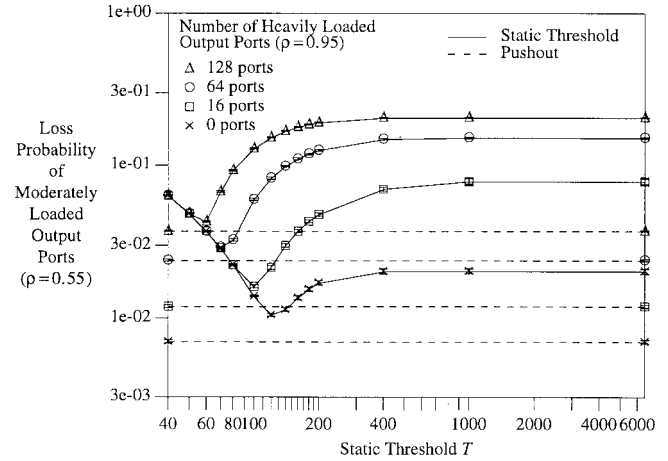


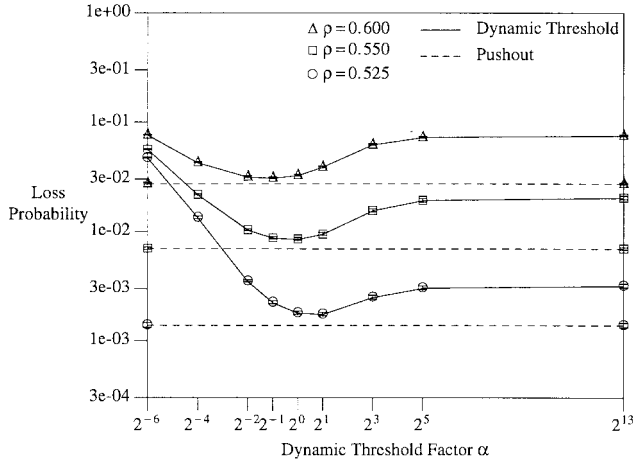
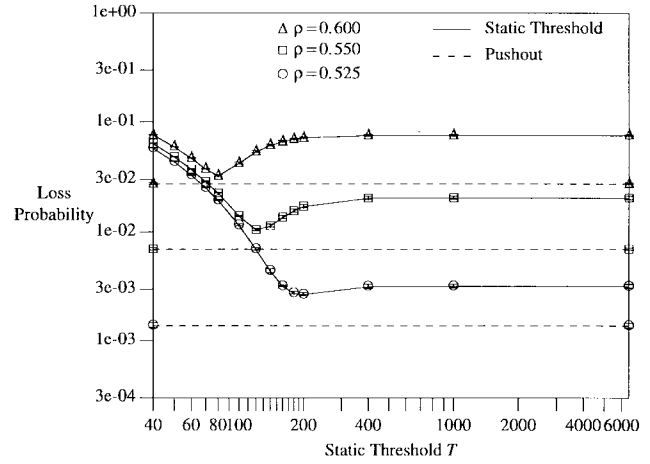
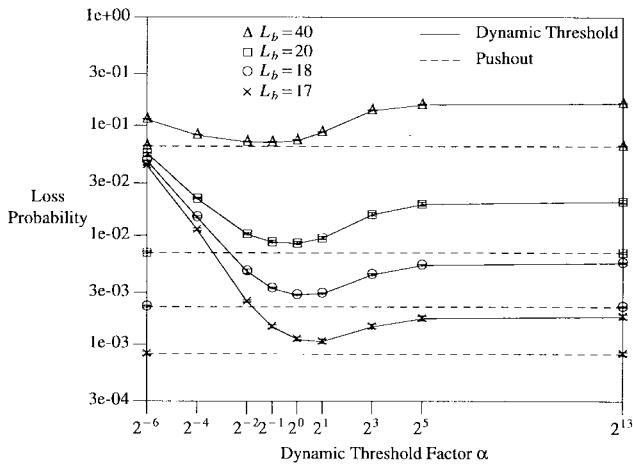
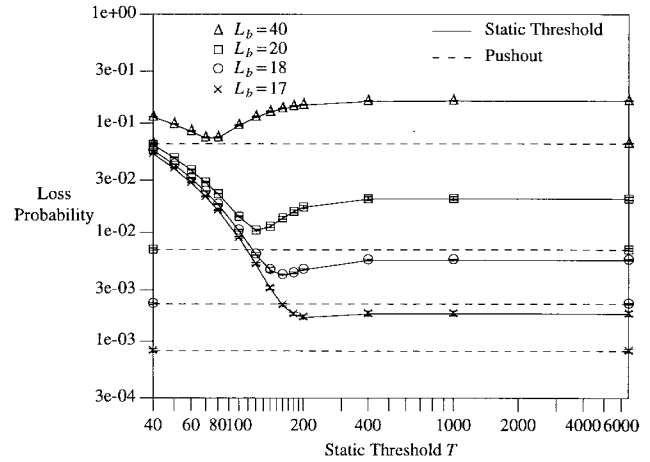
Fig. 12. Cell loss probability of moderately loaded output ports ( $\rho = 0.55$ ) versus ST setting  $T$ , for various numbers of heavily loaded output ports ( $\rho = 0.95$ ).

interesting of the bunch, are companion plots to Fig. 4. The performance of both ST and DT varies widely according to the setting of the control parameter  $T$  or  $\alpha$ . However, when optimally tuned for conditions, the losses for ST and DT are fairly close. Properly tuned, both DT and ST are dramatically better than NC—almost two orders of magnitude better for an overload factor of 10.00. The losses for PO are somewhat less than those of tuned ST and tuned DT. The most important point to be made with these figures is that both ST and DT handle single hot-spot overloads of any magnitude very well with  $\alpha$  set at 0.5 or 1.0 and  $T$  set at 120. Once set at these values, the parameters do not need to be retuned for hot spots of differing intensity. Of all of the traffic patterns we will consider, only for this hot-spot type of load will we find that ST is as robust as DT.

Next, in Figs. 11 and 12 we consider various numbers of heavily loaded output ports. These are companion plots to Fig. 5. The performance of both ST and DT varies widely according to the setting of the control parameter  $T$  or  $\alpha$ . However, when optimally tuned for conditions, the losses for ST and DT are fairly close. Properly tuned, both DT and ST

are much better than NC, but somewhat worse than PO. Now suppose the control parameter setting is fixed while the number of heavily loaded ports varies. For the four traffic patterns that we studied, the optimal setting for the DT factor  $\alpha$  ranges from 0.25 to 1.0, and fixing  $\alpha$  anywhere in that range does a reasonably good job in all four cases. However, for the same four traffic patterns, the optimal ST setting  $T$  ranges from 60 to 120, and there is *no* fixed parameter value that does a good job for all four cases. In this sense, DT is more robust than ST to changes in the number of heavily loaded ports. Of course, traffic changes such as these will cause changes in the cell loss probability no matter what control policy is used. With DT, however, the control parameter need not be retuned in order to attain performance close to that attainable with retuning. When the traffic changes, the performance of ST can be made close to DT, but only if ST's control parameter is retuned; if not retuned, the performance of ST will be far worse than it could be with retuning, and far worse than DT with or without retuning.

Now, in Figs. 13 and 14 we consider various load levels  $\rho$ , assuming uniform traffic conditions. These are companion

Fig. 13. Cell loss probability versus DT factor  $\alpha$ , for various loads  $\rho$ .Fig. 14. Cell loss probability versus ST setting  $T$ , for various loads  $\rho$ .Fig. 15. Cell loss probability versus DT factor  $\alpha$ , for various mean burst lengths  $L_b$ .Fig. 16. Cell loss probability versus ST setting  $T$ , for various mean burst lengths  $L_b$ .

plots to Fig. 6. When optimally tuned for conditions, ST is somewhat worse than DT, which is somewhat worse than PO. The difference between ST and DT is more pronounced, though, if the control parameter setting is tuned for optimal performance at some load level and fixed, and then the actual load  $\rho$  turns out to be other than expected. For the three load levels that we studied, the optimal setting for the DT factor  $\alpha$  ranges from 0.5 to 2.0, and fixing  $\alpha$  anywhere in that range does reasonably well for all three cases. However, for the same three load levels, the optimal ST setting  $T$  ranges from 80 to 200, and there is *no* fixed parameter value that does well for all three cases. In this sense, DT is more robust than ST to changes in  $\rho$ .

Finally, in Figs. 15 and 16 we consider various mean burst lengths  $L_b$ , assuming uniform traffic conditions. These are companion plots to Fig. 8. When optimally tuned for conditions, ST is somewhat worse than DT, which is somewhat worse than PO. Once again, the big difference between ST and DT becomes apparent when the control parameter setting is tuned for optimal performance at some mean burst length level and fixed, and then the actual mean burst length  $L_b$  changes. For the four burst lengths that we studied, the optimal setting for the DT factor  $\alpha$  ranges from 0.5 to 2.0, and any  $\alpha$  in that

range is acceptable in all four cases. However, for the four burst lengths, the optimal ST setting  $T$  ranges from 80 to 200, and there is *no* fixed parameter value that is acceptable for all four cases. We conclude that DT is more robust than ST to changes in  $L_b$ .

## V. CONCLUSIONS

In shared-memory packet switches, queue length controls can promote fair and efficient use of the packet buffer memory. The simplest control, ST, works well as long as the threshold is tuned properly for the load conditions. However, we showed that ST is not robust to many sorts of load changes unless the threshold setting is retuned. To get good performance under uniform traffic loads, one has to be especially careful not to set the threshold too low; for nonuniform and hot-spot loads, one has to be particularly careful not to set the threshold too high. It is impossible to find one threshold value that works well over a broad range of load conditions. The attraction of ST is its ease of implementation. At the other extreme, the PO scheme is very fair, space efficient, adaptive, and robust. Implementing PO, however, is far more complex than ST (especially for multiple space priorities).

As a compromise, we proposed a DT scheme that automatically adapts to changing load conditions according to a simple formula: at any instant of time, the output queue length threshold (beyond which new arrivals are blocked) is proportional to the current amount of unused buffering in the switch. Our analysis of the DT method showed that it (intentionally) holds a small amount of buffer space in reserve, and that it distributes the remaining buffer space equally among the active output queues. Like the ST control, the DT control has a single parameter, viz., the proportionality constant  $\alpha$ . For the switch size and traffic types that we studied, an appropriate value of  $\alpha$  is 1.0.

We studied the performance of ST and DT via simulation. First we optimized their control parameters for nominal load conditions and found that DT performed only slightly better than ST. Then we varied the traffic in several ways, without retuning the control parameters. First, both ST and DT were shown to handle intense overloads of single output ports very well. Next, we considered nonuniform traffic scenarios in which most output ports were moderately loaded, but a few were heavily loaded; taking the loss of the moderately loaded ports as our performance metric, we found that DT was more robust than ST to changes in the number of heavily loaded ports. Finally, DT was found to be *much* more robust than ST to changes in the load level and the burst length, for uniform loads. Thus, the DT scheme appears to be a reasonable compromise for queue length control in high-speed shared-memory packet switches. DT is nearly as simple to implement as ST and, because DT is adaptive, it offers most of the performance benefits of PO.

Although this paper considered the DT scheme in the context of ATM switches with true output queueing, the same idea could be applied to input-queued switches with virtual output queueing, to routers that internally segment packets into cells, and to variable-length packet switches and routers.

#### ACKNOWLEDGMENT

The authors would like to thank F. Bonomi, F. Chiussi, K. Hedlund, J. Kneuer, D. Rancich, and A. Wong for countless helpful discussions of the concept, design, implementation, application, and tuning of DT's.

#### REFERENCES

- [1] P. Gonet, J. P. Coudreuse, and M. Servel, "Implementing asynchronous transfer mode concepts: Main results of the prelude experiment," in *Proc. IEEE GLOBECOM'87*, Nov. 1987, pp. 1871–1875.
- [2] M. G. Hluchyj and M. J. Karol, "Queueing in high-performance packet switching," *IEEE J. Select. Areas Commun.*, vol. 6, pp. 1587–1597, Dec. 1988.
- [3] T. Kozaki, Y. Sakurai, O. Matsubara, M. Mizukami, M. Uchida, Y. Sato, and K. Asano, "32 × 32 shared buffer type ATM switch VLSI's for B-ISDN," in *Proc. IEEE ICC'91*, Denver, CO, June 1991, pp. 711–715.
- [4] M. I. Irland, "Buffer management in a packet switch," *IEEE Trans. Commun.*, vol. COM-26, pp. 328–337, Mar. 1978.
- [5] 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.
- [6] G. Latouche, "Exponential servers sharing a finite storage: Comparison of space allocation policies," *IEEE Trans. Commun.*, vol. COM-28, pp. 992–1003, June 1980.
- [7] G. J. Foschini and B. Gopinath, "Sharing memory optimally," *IEEE Trans. Commun.*, vol. COM-31, pp. 352–360, Mar. 1983.

- [8] A. K. Thareja and A. K. Agarwala, "On the design of optimal policy for sharing finite buffers," *IEEE Trans. Commun.*, vol. COM-32, pp. 737–740, June 1984.
- [9] S. X. Wei, E. J. Coyle, and M. T. Hsiao, "An optimal buffer management policy for high-performance packet switching," in *Proc. IEEE GLOBECOM'91*, Phoenix, AZ, Dec. 1991, pp. 924–928.
- [10] A. K. Choudhury and E. L. Hahne, "Space priority management in a shared memory ATM switch," in *Proc. IEEE GLOBECOM'93*, vol. 3, Houston, TX, Dec. 1993, pp. 1375–1383.
- [11] ———, "A simulation study of space priorities in a shared memory ATM switch," *J. High Speed Networks*, vol. 3, pp. 491–512, Nov. 1994.
- [12] L. Georgiadis, I. Cidon, R. Guérin, and A. Khamisy, "Optimal buffer sharing," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1229–1240, Sept. 1995.
- [13] Y. S. Lin and C. B. Shung, "Quasi-pushout cell discarding," *IEEE Commun. Lett.*, vol. 1, pp. 146–148, Sept. 1997.
- [14] A. K. Choudhury and E. L. Hahne, "New implementation of multipriority pushout for shared memory ATM switches," *Comput. Commun.*, vol. 19, pp. 245–256, Mar. 1996.
- [15] A. K. Thareja and S. K. Tripathi, "Buffer sharing in dynamic load environment," in *Proc. IEEE INFOCOM'84*, San Francisco, CA, Apr. 1984, pp. 369–380.
- [16] D. Tipper and M. K. Sundareshan, "Adaptive policies for optimal buffer management in dynamic load environments," in *Proc. IEEE INFOCOM'88*, New Orleans, LA, Mar. 1988, pp. 535–544.
- [17] J. M. Jaffe, "Bottleneck flow control," *IEEE Trans. Commun.*, vol. COM-29, pp. 954–962, July 1981.
- [18] E. L. Hahne, A. K. Choudhury, and N. F. Maxemchuk, "Distributed-queue-dual-bus networks with and without bandwidth balancing," *IEEE Trans. Commun.*, vol. 40, pp. 1192–1204, July 1992.
- [19] A. K. Choudhury and E. L. Hahne, "Dynamic thresholds for multiple loss priorities," in *Proc. IEEE ATM'97 Workshop*, Lisbon, Portugal, May 1997, pp. 272–281.
- [20] ———, "Dynamic queue length thresholds for multipriority traffic," in *Proc. ITC 15*, vol. 2a, V. Ramaswami and P. Wirth, Eds. Washington, DC: Elsevier, 1997.



**Abhijit K. Choudhury** (S'88–M'91) received the B.Tech. degree from the Indian Institute of Technology, Kanpur, India, in 1986, the M.S. degree from the State University of New York, Stony Brook, in 1987, and the Ph.D. degree from the University of Southern California, Los Angeles, in 1991, all in electrical engineering.

Since 1991, he has been with Bell Laboratories, Murray Hill, NJ, and Holmdel, NJ. He has worked on the design and analysis of fast packet switches, access and routing protocols in LAN's and MAN's, and intellectual property protection in electronic publishing. His current research focuses on buffer management in shared-memory switch architectures. He holds four patents and has published over 20 journal and conference papers.

Dr. Choudhury has served as Guest Editor for a special issue of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS on advances in ATM switching systems for B-ISDN. He has served on the Technical Program Committees of IEEE INFOCOM, the IEEE ATM Workshop, and the IEEE LAN/MAN Workshop. He is a Member of the IEEE Communications Society, the IEEE Computer Society, and Eta Kappa Nu.



**Ellen L. Hahne** (S'76–M'77) received the B.S. degree in electrical engineering from Rice University, Houston, TX, in 1978, and the S.M., E.E., and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1981, 1984, and 1987, respectively.

Since 1978 she has been with Bell Laboratories, Holmdel, NJ, and Murray Hill, NJ. Traffic regulation is her primary interest, and she has conducted research and design projects on scheduling and flow control in WAN's, access protocols for dual-bus and rectangular-mesh MAN's, resource management in ATM switches, congestion control for voice telephony networks, and dynamic routing in automated factories.

Dr. Hahne received the Presidential Scholar award from the U.S. Government in 1974 and the Distinguished Member of Technical Staff award from Bell Laboratories in 1993.