# MAINTAINING HIGH THROUGHPUT DURING OVERLOAD IN ATM SWITCHES

Jonathan S. Turner
Washington University
jst@cs.WUSTL.edu

*Abstract*

This paper analyzes two popular heuristics for ensuring packet integrity in ATM switching systems. In particular, we analyze the behavior of packet tail discarding, in order to understand how the packet level link efficiency is dependent on the rates of individual virtual circuits and the degree of the imposed overload. In addition, we study early packet discard and show that the queue capacity needed to achieve high efficiency under worst-case conditions grows with the number of virtual circuits and we determine the efficiency obtainable with more limited queue capacities. Using the insights from these analyses, extensions to early packet discard are proposed which achieve high efficiency with dramatically smaller queue capacities (independent of the number of virtual circuits).

## 1 Introduction

ATM networks are intended to carry a mix of information streams with widely different bandwidth characteristics. At one extreme, we have continuous applications that transmit at a steady, fixed rate, often with modest bandwidth, and at the other extreme, we have highly bursty applications that from time-to-time transmit very large blocks of information (hundreds of kilobytes or more) at high peak rates, but relatively low average rates.

During the late eighties, when the original ideas for ATM were first being worked out in detail, it was expected that all applications would provide a set of traffic descriptors, at the time a virtual circuit is established, which would describe the behavior of the virtual circuit with sufficient accuracy that they could be used to make routing and call acceptance decisions; and, that so long as the virtual circuits were all constrained to operate within the bounds defined by their traffic descriptors, congestion would be sufficiently rare and short-lived, that no additional mechanisms would be required for congestion control.

This approach remains viable for continuous rate streams and for variable rate streams with peak rates that are small relative to the link rate and modest peak-to-average rate ratios (such as real-time, coded video). However, it has become clear that it may not work well for many traditional data applications. The most compelling examples involve interactive information retrieval, where large data files (such as images) are retrieved across the network on demand from a human user. The inherent unpredictability of the human user, combined with the high peak transmission rates necessitated by providing acceptable interactive response (exacerbated by the inability of many ATM network adaptors to perform even peak rate pacing), forces the network designer and operator to accept either low network utilization or the occurrence of overload periods during which the traffic sent to specific network links, exceeds their capacities.

The fact that end-to-end transport protocols send information in packets containing many ATM cells makes the impact of overload periods worse than it would be otherwise, since a single lost cell can lead to the loss and retransmission of an entire transport-level packet. For protocols that use a go-back-$N$ retransmission strategy, a single lost cell can cause the retransmission of an amount of data equal to what would be sent during a complete network round-trip delay. What this means is that during overload periods, ATM networks can experience congestion collapse, where the throughput drops to zero, as the offered load increases. While the adaptive-windowing mechanisms in many transport protocols will eventually reduce the offered traffic to a level that the network can support, very little data is passed through

[0]This work was supported by the ARPA Computing Systems Technology Office, Ascom Timeplex, Bay Networks, Bell Northern Research, NEC, NTT, Samsung, Southwestern Bell and Tektronix.

the congested part of the network during the period it takes for these end-to-end mechanisms to respond.

A number of approaches have been proposed to address this problem, including fast-reservation protocols, such as described in [1]. Turner [5] has shown that fast reservation protocols can be implemented efficiently and Mahdavian [4] has shown that they can maintain high link efficiencies during overload with small buffers. The ATM Forum has recently standardized a rate-based flow control scheme to manage congestion in the presence of unpredictable bursty traffic [3]. This will be part of the service definition for the *Available Bit Rate* (ABR) service. However, like adaptive windowing mechanisms, the rate-based flow control mechanisms can require one or more network round-trip times before they effectively control congestion. Also, like adaptive windowing, rate-based flow control will not be universally applied, but will coexist with applications using the *Unspecified Bit Rate* (UBR) service, which lacks any such congestion control mechanisms.

It is clear that overload periods will occur in ATM networks from time-to-time, making it important to improve performance of networks during these overload periods. One approach to improving performance is to add sufficient buffering at each ATM network link, that overload periods lasting up to a few network round-trip times can be tolerated. While this can potentially eliminate packet loss, it can require large and fairly expensive buffers as well as complicated buffer controllers, significantly increasing the cost of ATM switching systems, for the sake of a mechanism that is used only during occasional overload events. In this paper, we focus instead, on mechanisms that accept that cell loss will occur from time-to-time, but attempt to ensure that the available network capacity is effectively utilized, by preserving the integrity of transport level packets.

In this note, we analyze two well-known mechanisms for preserving packet integrity in ATM networks, and introduce two variants on one of them which can achieve very high throughput during overloads, even with very modest buffer sizes. For this analysis, we focus on a single queue operating under overload conditions. While such an analysis addresses only part of the question of overall network performance, the detailed behavior of the individual queues is an important determinant of the overall network performance and understanding this behavior can yield insights into how these mechanisms can be improved.

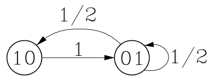
## 2 Packet Tail Discarding

If a cell must be discarded from a transport level packet, because there is no room for it in the queue, there is clearly no point in adding any additional cells from that packet to the queue, since they will simply consume buffer space and transmission bandwidth without delivering useful data to the destination. Thus, if we keep track of which traffic streams have experienced cell discarding, we can discard the remainder of the packet (up to the last cell), freeing up buffer space and transmission bandwidth for other packets. The mechanism needed to implement this scheme is quite simple and it can be effective under certain conditions. Floyd and Romanov [2] have performed a simulation study to assess the performance of packet tail discarding in a simple network configuration, where the link in question is heavily overloaded and carries TCP/IP traffic. In this section, we present an approximate analysis that can be used to explore the performance of packet tail discarding under a wider range of conditions, although without considering the interaction with an end-to-end protocol.

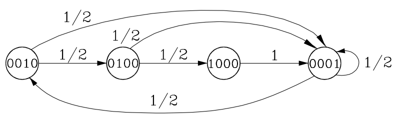
Consider a homogeneous situation in which $r$ virtual circuits transmit data continually at a normalized rate of $\lambda$ (that is, $\lambda$ is the fraction of the link bandwidth required by a single virtual circuit), where the *overbooking ratio* $r\lambda > 1$. Assume that all packets contain $\ell$ cells and let $k = \lfloor 1/\lambda \rfloor$ be the maximum number of virtual circuits that the link can handle without loss.

Define a given virtual circuit to be *active* if cells from that virtual circuit are being placed in the queue on arrival. Similarly, define a virtual circuit to be *inactive* if its cells are being discarded. We make two simplifying assumptions, one optimistic, and one pessimistic. The optimistic assumption is that so long as the number of virtual circuits is $\leq k$, the number of cells in the buffer does not increase, so no further cells can be lost until some new virtual circuit becomes active. This is optimistic, because it neglects the impact of jitter on the arrival time of cells to the queue. However, we also assume that whenever we have $k+1$ virtual circuits active, the buffer immediately overflows, causing one of the active virtual circuits to be made inactive. This is pessimistic, since it ignores the possibility that the buffer level will drop sufficiently during periods when there are $\leq k$ active virtual circuits to allow packets to be completed during a subsequent period when the number of active virtual circuits exceeds $k$.

We are interested in computing the *goodput* of the link, which we define as the fraction of the link's capacity that is used to carry complete packets. This is equal to the probability a packet is successfully propagated (that is, none of its cells are lost) times $r\lambda$. We can construct a Markov chain that models the packet loss process (given the assumptions mentioned above). At any given point in time, the state of the system consists of the active/idle status of the $r$ virtual circuits, listed in the order of the time of their next packet boundary, relative to the current time. So for example, the bit vector 100110010 could represent a system with $k = 4$ and $r = 9$ in which the next upcoming packet boundary belongs to an active virtual circuit, while the subsequent two packet boundaries belong to virtual circuits that are inactive and so forth. State transitions occur at packet boundaries. For $\lambda = k = 1$ and $r = 2$ we get the Markov chain




The transition from 10 to 01 with probability 1 models the fact that a packet boundary for an active virtual circuit causes that virtual circuit to become the last one in the list, because its next packet boundary is furthest in the future, but does not change the active/idle status of either virtual circuit. The two transitions with probability 1/2 from state 01 represent the fact that when a packet boundary occurs on the inactive virtual circuit, it becomes active, causing the buffer to overflow and causing a cell to be discarded from one of the two virtual circuits, hence making it inactive. By inspection, the steady state probabilities for the chain are 1/3 and 2/3 and the goodput (which is equal to the offered load times the sum of the steady-state probabilities of all states in which the first bit is 1) is 2/3. For $\lambda = k = 1$ and $r = 4$, we have a similar Markov chain




with steady state probabilities 1/15, 2/15, 4/15 and 8/15 (from left to right) and a resulting goodput of 4/15. In general, for $\lambda = k = 1$ we obtain a goodput of $r/(2^r - 1)$. Another special case that has a particularly simple solution is the one in which $k = 1/\lambda$ and $r = k + 1$. For $k = 3$ and $r = 4$ we get the Markov chain

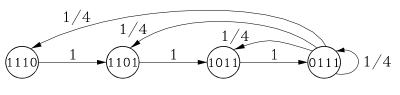



Figure 1: Goodput for Tail Discarding (analysis shown as boxes, simulation as lines)

which has steady state probabilities of 1/10, 2/10, 3/10 and 4/10 and a resulting goodput of 4/5. In general, when $k = 1/\lambda$ and $r = k$, the goodput is $1 - 1/(r + 1)$.

For general $r$ and $k$, the models become too large to work by hand. For computational purposes, it is more convenient to use an alternative state representation. In particular, we replace the bit vector with a list of integers specifying where the 1's would appear in the sequence (or alternatively, the zeros). So for example, the bit vector 10010010 could be represented by the list [0,3,6]. Using this notation, we can specify the transitions from any state $[s_1, \ldots, s_k]$ and their probabilities. There are two cases. If $s_1 = 0$ then there is a single transition from state $[s_1, \ldots, s_k]$ with probability 1 to state $[s_2 - 1, s_3 - 1, \ldots, s_k - 1, r - 1]$. If $s_1 > 0$ there are $k+1$ transitions, each with probability $1/(k + 1)$ to the states

$$\begin{aligned}
&[s_1 - 1, s_2 - 1, \ldots, s_k - 1],\\
&[s_2 - 1, s_3 - 1, \ldots, s_k - 1, r - 1],\\
&[s_1 - 1, s_3 - 1, \ldots, s_k - 1, r - 1], \ldots,\\
&[s_1 - 1, s_2 - 1, \ldots, s_{k-1} - 1, r - 1]
\end{aligned}$$

The probability that a packet is successfully transmitted is the sum of the steady state probabilities for those states $[s_1, \ldots, s_k]$ with $s_1 = 0$ and the goodput is this probability times $r\lambda$. The Markov chain has $\binom{r}{k}$ states and $\binom{r-1}{k-1} + \binom{r-1}{k}(k + 1)$ state transitions. When $r$ and $k$ are not too large it can be solved using direct iterative techniques, but for large $n$ and $k$ this becomes impractical. While various approximations are possible, direct simulation also yields fast accurate results.

Figure 1 shows results obtained using analysis and simulation. For those points where both analytical and simulation results are available, the relative error is typically about 1%. The goodput drops as the offered load increases and as $\lambda$ decreases. This latter effect is caused by the fact that as $\lambda$ decreases, $r$ must increase to achieve a given overbooking ratio, and when $r$ is large, each packet is exposed to discarding with greater frequency. Of course, bursty traffic comprising virtual circuits with smaller peak rates is better in the sense that it is less likely to lead to overload in the first place, but once we're in overload, small virtual circuits lead to poorer performance. It's also important to note that when $\lambda$ does not evenly divide the link rate, performance deteriorates due to fragmentation effects. The figure shows that while packet tail discarding can improve performance during overload, it cannot stave off congestion collapse by itself. While reasonable goodput can be achieved when the entering traffic is less than twice the link rate, beyond that, the performance deteriorates significantly.

## 3 Early Packet Discard (EPD)

Early packet discard is another technique that attempts to preserve the integrity of transport level packets. In early packet discard, whenever a virtual circuit begins transmission of a new packet, a decision is made to attempt to propagate the packet or not. In particular, if the number of cells in the queue exceeds some specified threshold, then the packet is not propagated into the queue. If the number of cells is below the threshold, the packet is propagated. In addition, if any cell of the packet must be discarded due to queue overflow, the remainder of the packet is discarded.

Unlike packet tail discarding, early packet discard can avoid packet loss entirely if the queue is sufficiently large. We start therefore, with an analysis of how large a queue is needed to avoid loss entirely, under worst-case conditions. Later, we will consider how early packet discard performs with a smaller queue.

Consider a general situation in which $r$ virtual circuits are sending packets continually at rates $\lambda_1, \ldots, \lambda_r$. Assume also that the corresponding packet lengths are $\ell_1, \ldots, \ell_r$ and that for $1 \leq i < r$, $(\ell_i/\lambda_i) \leq (\ell_{i+1}/\lambda_{i+1})$.

Now, if we observe the number of cells in the queue as a function of time, we will observe a cyclic behavior in which the number of cells in the queue rises above the threshold, then as various virtual circuits complete packets, the number of cells stops increasing and drops back down. When it falls below threshold, and new packets start arriving at the queue, the buffer level stops falling and eventually begins to rise again.

Now, so long as the queue occupancy curve always levels off before it either fills with cells or becomes empty, we propagate only complete packets into the queue and the link never becomes idle. That is, all of the link's capacity is used to carry complete packets, meaning we achieve 100% goodput. So the question is, how far can the buffer level rise above (or drop below) threshold before leveling off.

In the worst-case, all $r$ virtual circuits begin sending packets just before the queue occupancy curve rises above the threshold. In this case, the slope of the queue occupancy curve, at the time the threshold is crossed is $(\sum_{i=1}^{r} \lambda_i) - 1$. If we define $s$ to be the largest integer for which $\sum_{i=s}^{r} \lambda_i > 1$, the buffer level will continue to increase until the packet that just started on virtual circuit $s$ completes. That means that the time that passes before the queue occupancy curve levels off is $\ell_s/\lambda_s$. We can determine how far the queue level rises in this time by analyzing the shape of the queue occupancy curve and notice that assuming an ideally smooth flow of cells in and out of the queue, the queue occupancy curve is piecewise linear with a slope that depends on which virtual circuits continue to be active (that is, continue to place cells in the queue). This is illustrated in Figure 2. By inspection of the figure, we see that the queue level exceeds the threshold by at most

$$\begin{aligned}
&\frac{\ell_1}{\lambda_1}\left[\left(\sum_{i=1}^{r}\lambda_i\right)-1\right]+\left(\frac{\ell_2}{\lambda_2}-\frac{\ell_1}{\lambda_1}\right)\left[\left(\sum_{i=2}^{r}\lambda_i\right)-1\right]\\
&+\left(\frac{\ell_3}{\lambda_3}-\frac{\ell_2}{\lambda_2}\right)\left[\left(\sum_{i=3}^{r}\lambda_i\right)-1\right]\\
&\qquad+\cdots+\left(\frac{\ell_s}{\lambda_s}-\frac{\ell_{s-1}}{\lambda_{s-1}}\right)\left[\left(\sum_{i=s}^{r}\lambda_i\right)-1\right]\\
&=\sum_{i=1}^{s}\ell_i-\frac{\ell_s}{\lambda_s}\left(1-\sum_{i=s+1}^{r}\lambda_i\right)\leq\sum_{i=1}^{s}\ell_i
\end{aligned}$$

By a similar argument, the maximum amount the queue level drops below threshold is

$$\sum_{i=1}^{t-1}\ell_i+\left(\frac{\ell_t}{\lambda_t}\right)\left(1-\sum_{i=1}^{t-1}\lambda_i\right)\leq\sum_{i=1}^{t}\ell_i$$

where $t$ is the smallest integer such that $\sum_{i=1}^{t}\lambda_i \geq 1$. For the uniform case, $\lambda_i = \lambda$, $\ell_i = \ell$ for all $i$ and $s = r-k$ ($k = \lfloor 1/\lambda \rfloor$). The maximum excursion above the threshold is

$$(r-k)\ell - (\ell/\lambda)(1-k\lambda) = (r-1/\lambda)\ell$$

and the maximum excursion below threshold is

$$\lceil 1/\lambda \rceil \ell + (\ell/\lambda)(1-\lambda\lceil 1/\lambda \rceil) = \ell/\lambda$$

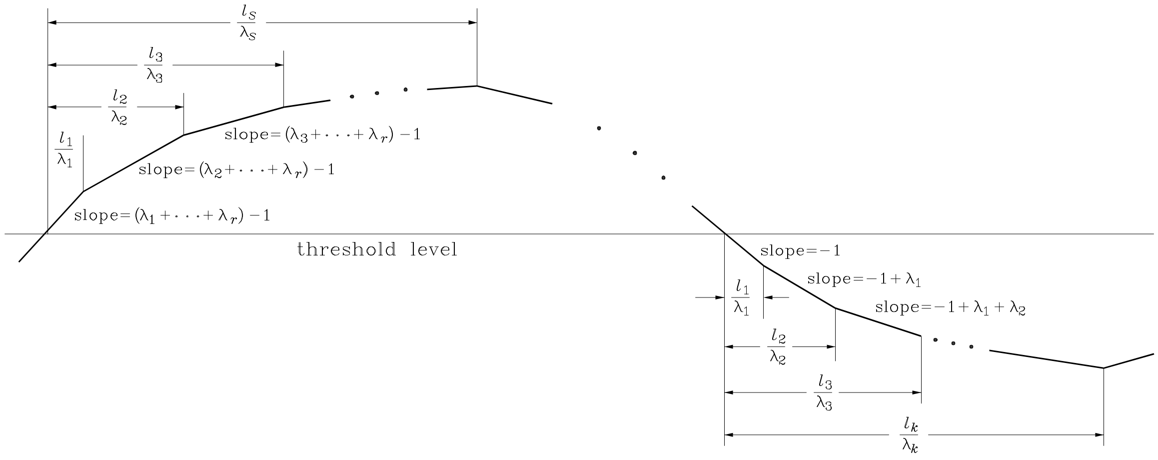



Figure 2: Queue Occupancy with Early Packet Discard and Large Queues

The total required buffer size is $r\ell$ or one packet for every virtual circuit.

Consider now what happens if the amount of buffering available is more limited, so that the queue does sometimes overflow. Let $B$ be the number of cells the queue can accommodate and let $b < B$ be the threshold used by the early packet discard algorithm. Consider a homogeneous situation in which all virtual circuits are sending packets with $\ell$ cells each at a rate of $\lambda$. We'll assume that $b$ is large enough to prevent underflow of the queue and focus on the packet loss due to overflow.

As before, the queue occupancy curve passes through cycles where it rises above threshold until enough packets end for it to level off or until the queue becomes full, causing cells to be discarded from some packets. It then drops below threshold, and as new packets start, the queue occupancy begins to rise again, starting a new cycle. If the queue is too small to allow packets to complete between the time they start turning on at the bottom of the cycle and the time the queue becomes full, then in each cycle $k$ packets are successfully transmitted. If the duration of the cycle is $T$, then the goodput is just $k\ell/T$, since the best we could hope for is to propagate $T/\ell$ complete packets during a period of length $T$ (the time unit is the time it takes to transmit a cell). The worst throughput is then obtained when $T$ is as long as possible. The worst case occurs if all virtual circuits begin a new packet just before a downward threshold crossing. This causes the queue level to drop to a level of $\ell/\lambda$ below the threshold, before starting to rise with a slope of $r\lambda - 1$. Each of these packets requires time $\ell/\lambda$ to complete. Since we're interested in the small buffer case, we assume that before this happens, the buffer fills. Consequently, all but $k$ of the virtual circuits will lose cells causing the remainder of their packets to be discarded. The buffer occupancy curve then starts falling with a slope of $k\lambda - 1$ until the virtual circuits whose cells are still being placed in the queue end. At this point, these still active virtual circuits become inactive, and the buffer level drops with a slope of $-1$ until it crosses below threshold, completing the cycle.

Figure 3 shows the shape of the buffer occupancy curve under these assumptions. Initially it drops with a slope of $-1$, then it rises with a slope of $r\lambda - 1$ until it becomes full. The length of the time period during which the buffer level rises is $((B-b)+(\ell/\lambda))/(\lambda r - 1)$. It then starts dropping with a slope of $k\lambda - 1$ until the packets that did not lose cells finish. This happens at time $\ell/\lambda$ after the buffer level first began to rise, meaning that the queue level drops to a level of

$$(B-b) - (1-k\lambda)\left(\frac{\ell}{\lambda} - \frac{(B-b)+\ell/\lambda}{\lambda r - 1}\right)$$

before it starts falling with a slope of $-1$. Putting this together, we find that

$$T \leq 2\frac{\ell}{\lambda} + (B-b) - (1-k\lambda)\left(\frac{\ell}{\lambda} - \frac{(B-b)+\ell/\lambda}{\lambda r - 1}\right)$$

The resulting worst-case estimate for the goodput is then

$$k\lambda\left[2 + \frac{B-b}{\ell}\lambda + (1-k\lambda)\left(1 - \frac{1+\lambda(B-b)/\ell}{\lambda r - 1}\right)\right]^{-1}$$

Note, that this analysis requires that $\ell/\lambda > ((B-b)+\ell/\lambda)/(\lambda r - 1)$, or equivalently

$$(B-b) + \ell/\lambda < (r - 1/\lambda)\ell$$

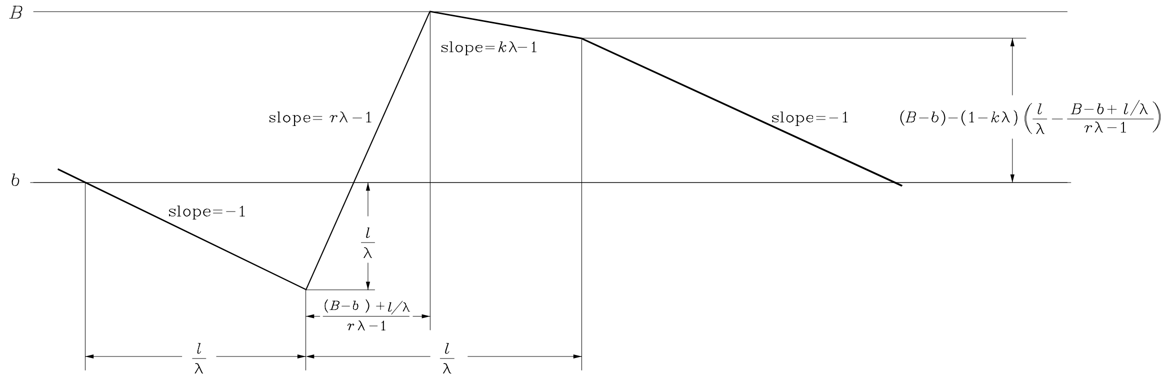



Figure 3: Queue Occupancy with Early Packet Discard and Small Queues

Note that the left side of this expression is the range over which the buffer occupancy varies during an overload period. When $1/\lambda$ is an integer, the expression for goodput simplifies to

$$\frac{1}{2+\lambda(B-b)/\ell}$$

The expression $(B-b)/\ell$ is just the number of packets that can be accommodated in the queue between the threshold and the 'top.' There are some interesting observations one can make from this expression. First, notice that in the region where this analysis applies, the goodput is independent of $r$. That is, as you increase the number of virtual circuits, the goodput remains constant. Of course, for values of $r$ that are small enough, we saw from the previous analysis that it's possible to achieve zero loss. So, as $r$ increases from smaller values to larger, there are two distinct regions of operation with a fairly abrupt transition between the two. Also notice that since the denominator is $\geq 2$, the goodput is never greater than 1/2 and if $B-b=\ell$, the goodput is never less than 1/3. Finally, observe that as $B$ increases, the goodput actually drops, since so long as we stay within the region where the queue overflows, increasing $B$ does not increase the number of packets successfully transmitted per cycle. It merely increases the duration of the cycle.

It should be noted that since these analyses are based on a worst-case assumption about the relative timing of the packet boundaries on different virtual circuits, they can be unrealistically pessimistic. Ideally, one would determine average throughputs based on random 'phase' relationships among the different virtual circuits. A simple first step in this direction is to determine how the queue would behave if the virtual circuits are all uniformly offset from one another (that is, the time between successive packet boundaries is $\ell/r\lambda$). An analysis similar to those given above leads to the conclusion that for $r\lambda \geq 2$, the total range over which the buffer varies is $k\ell(1-1/r\lambda)$ or roughly $k\ell$ when the offered load is very large. This is a substantial improvement over the $r\ell$ required by the worst-case analysis for heavy loads, although the dependence on $k$ still implies a potentially substantial buffer size when the traffic consists of many low rate connections.

Another feature of these analyses that can be legitimately questioned is that they assume that the sources pace their transmissions so that cells are evenly spaced. In some cases, hosts on an ATM network may perform rate control at the packet level, rather than the cell level, meaning that cells in a given packet will be sent in a 'clump' rather than distributed over time. It turns out that this behavior does not affect the worst-case analyses at all. However, it does affect the 'uniform offset' analysis described above. It turns out that the results of that analysis can be reinterpreted for 'clumped' transmissions, by substituting the peak rate during the clump for the peak rate for paced transmissions. This does lead to improved performance for clumped transmissions under the uniform offset assumption. However, it should be understood that while clumped transmissions give better performance during overload for a given buffer size, they also make it more likely that the system will become overloaded in the first place, making it desirable to avoid clumped transmissions wherever possible.

While goodput is perhaps the most important characteristic of the performance of mechanisms for maintaining packet integrity, the issue of fairness can also be important in certain situations. We can get some insight into this by considering a typical cycle in the operation of early packet discard. From the discussion above, in a typical cycle, all active virtual circuits become inactive during the period the queue level is above the threshold. However, it's not always the case that all inactive virtual circuits become active during the period when the queue level is below threshold. In particular, the virtual circuits with small packet durations are most likely to turn on before the queue level rises above threshold. In cases where the packet lengths are equal or nearly equal, this implies that the highest rate virtual circuits get more opportunities to turn on than the lowest rate virtual circuits. Since, the high rate virtual circuits also transmit at a higher rate when they are active, they obtain a disproportionate fraction of the link's real capacity.

## 4 EPD With Hysteresis

As we have seen, the achievable goodput for early packet discard under heavy loads can be limited unless large buffers are used. To improve the performance of early packet discard when buffer space is limited, we need to increase the number of packets successfully transmitted during each cycle. To accomplish this, we introduce some hysteresis into the process of selecting packets for discarding. As in early packet discard, each virtual circuit transmitting packets, is in one of two states, active or inactive. Cells belonging to active virtual circuits are added to the queue, while those belonging to inactive virtual circuits are discarded. The rules for determining the state of a virtual circuit are described below. Also, as with early packet discard, there is a threshold used to make decisions and an additional queue level called the *floor* level, which is typically be set to some small number of cells (say 10). At the end of a packet belonging to virtual circuit $V$, we execute the following algorithm:

> If the current queue level exceeds the threshold and exceeds the maximum level since the most recent threshold crossing, make $V$ inactive.
>
> If the current queue level is below threshold and is either below the floor level or below the minimum level since the last threshold crossing, make $V$ active.
>
> In all other cases, do not change the state of $V$

In addition to these rules, if a cell is discarded from a virtual circuit because there is no room for it in the queue, then the virtual circuit's state is changed to inactive.

For uniform virtual circuits with rate $\lambda$, this algorithm leads to a cyclic behavior that is similar to that for early packet discard. However, unlike early packet discard, the slope of the buffer occupancy curve is smaller, leading to longer cycles and more packets transmitted successfully per cycle. In particular, during the rising part of a cycle, the slope of the buffer occupancy curve will typically be $(k+1)\lambda-1$ and during the falling part of a cycle, the slope of the buffer occupancy curve will typically be $k\lambda-1$. This is illustrated in Figure 4. As with early packet discard, the queue level can increase for a time period of at most $\ell/\lambda$ following an upward threshold crossing. This means that the maximum excursion above the threshold is

$$((k+1)\lambda-1)(\ell/\lambda) = (k+1-1/\lambda)\ell \leq \ell$$

and similarly, the maximum excursion below threshold is

$$(1-k\lambda)(\ell/\lambda) = ((1/\lambda)-k)\ell \leq \ell$$

Adding the two together, we see that the total over which the queue level ranges is bounded by $\ell$ and if we have sufficient buffering for two complete packets and fix the threshold at the half-full level, that we can avoid loss completely. This remains true, even for non-uniform rates and packet lengths. To see this, let $\lambda_i$ be the maximum rate among the set of active virtual circuits at the time of an upward threshold crossing. Notice that the slope of the buffer occupancy curve will generally be less than $\lambda_i$, since at the bottom of the cycle we only turn on enough virtual circuits to stop the queue level from dropping. If $\ell_i$ is the packet length for virtual circuit $i$, then the packet from virtual circuit $i$ will end $\ell_i/\lambda_i$ time units after the threshold crossing, so if the queue level is still rising at that point, virtual circuit $i$ will become inactive, causing the queue level to fall. A similar argument applies when the queue level is dropping, so if the queue size is large enough for two maximum size packets, we can avoid loss completely.

What makes the basic algorithm effective with small buffers is that it bounds the slope of the buffer occupancy curve. However, if the buffer occupancy curve is close to zero, the time between status changes for individual virtual circuits can become very long, meaning that virtual circuits are locked in either the active or inactive states for extended periods of time. We can increase the rate of turn-over among the virtual circuits by simply augmenting the basic algorithm with a counter that forces additional state changes after a certain amount of time has expired. For example, if the queue level is above threshold and falling, and the time since the last state change is greater than some specified bound, we can turn off additional active virtual circuits to increase the magnitude of the slope of the buffer occupancy curve. Similarly, if the queue level is below threshold and rising, we can turn on additional virtual circuits if too much time has passed since the last state change.

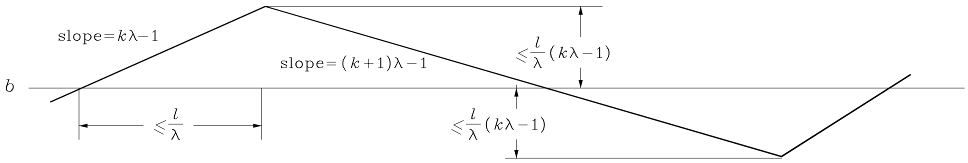



Figure 4: Early Packet Discard with Hysteresis

Note that the long time between state changes can be a positive characteristic. In particular, if end-to-end transport protocols use a go-back-$N$ retransmission strategy, then once we start discarding packets from a virtual circuit, it's better to continue discarding for a time equal to a network round-trip time. Of course, slow turn-over rates also raise the issue of fairness in allocating the link bandwidth. To get more insight into this issue, we need to look more closely at how inactive virtual circuits become active and how active virtual circuits become inactive.

During each cycle in the buffer occupancy curve, inactive virtual circuits have an opportunity to become active during the portion of the cycle where the buffer occupancy curve is below threshold and falling. Virtual circuits that have packets ending early in this time interval are most likely to become active. If we assume there is no systematic synchronization between packet boundaries and the timing of the buffer occupancy cycle, then the virtual circuits that are most likely to become active are those for which the packet duration $(\ell_i/\lambda_i)$ is small. For virtual circuits sending long bursts of information, the packet lengths are typically the maximum allowed by the end-to-end transport protocols, so that most of the variation in the packet duration arises from rate differences, rather than packet length differences. Consequently, virtual circuits with a higher rate are more likely to get picked to become active than those with a lower rate. However, by the same reasoning, once active, a high rate virtual circuit is more likely to be made inactive than a lower rate virtual circuit. So the higher rate virtual circuits tend to cycle more rapidly between active and inactive, than the lower rate virtual circuits, but one would expect that each would get a similar fraction of their packets into the queue, over an extended period of time.

Thus, early packet discard with hysteresis can achieve a greater degree of fairness than the original early packet discard mechanism. If our primary concern is simply maintaining high throughput until end-to-end mechanisms (like rate-based flow control or adaptive windowing) begin to take effect, then this level of fairness may be sufficient. However, if we expect long lasting congestion periods, it may be important to achieve a higher degree of fairness, in which we attempt to allocate the link bandwidth equally among the competing virtual circuits. The next section describes a mechanism that seeks to achieve this objective.

## 5 Fair EPD With Hysteresis

In this section, we introduce another variant of early packet discard which achieves high throughput with modest queue sizes, but also attempts to achieve a level of fairness among the competing virtual circuits, when their rates differ significantly.

In this variation, we replace the single threshold of early packet discard with two thresholds; the lower one is denoted by $b_\ell$ and the upper one by $b_h$. The buffer controller attempts to keep the buffer occupancy between these two thresholds by activating or deactivating different virtual circuits at packet boundaries. The selection of virtual circuits to activate or deactivate depends on the number of cells sent during a measurement window, relative to the number sent by other virtual circuits.

For each virtual circuit we keep track of the number of cells that have been placed in the queue for that virtual circuit during the recent past. To make it easy to maintain this information, we divide time into a series of consecutive intervals, and for each virtual circuit, we maintain counts of the number of cells that have been queued in the current interval and the previous interval. The combination of the current and previous intervals is called the *current window* and by summing the counts for its component intervals (or half-windows), we can track the number of cells in the current window, with minimal overhead and logic complexity. The count of the number of cells sent in the current window on virtual circuit $i$ is called $n_i$. Define $A_m$ to be the smallest $n_i$ from the set of currently active virtual circuits. Similarly, define $I_M$ to be the largest $n_i$ from among the currently inactive virtual circuits. If there are no currently active virtual circuits, let $A_m = I_M$ and if there are no currently inactive virtual circuits, let $I_M = A_m$. Given these definitions, when a packet belonging to virtual circuit $V$ ends:

> If the current queue level exceeds $b_h$ and exceeds the maximum level since rising above $b_h$, make $V$ inactive.
>
> If the current queue level is below $b_\ell$ and is either below the floor level or below the minimum level since dropping below $b_\ell$, make $V$ active.
>
> If the current queue level is between $b_\ell$ and $b_h$, the queue level is falling, $V$ is inactive and $n_V < A_m$ then make $V$ active.
>
> If the current queue level is between $b_\ell$ and $b_h$, the queue level is rising, $V$ is active and $n_V > I_M$ then make $V$ inactive.
>
> In all other cases do not change the state of $V$.

The first two rules are similar to the rules in early packet discard with hysteresis, and ensure that so long as the queue capacity above the upper threshold and below the lower threshold is at least equal to the size of a packet, the queue will neither overflow nor underflow. The third and fourth rules attempt to achieve fairness by turning on virtual circuits that have sent fewer cells in the current window than at least one of the currently active virtual circuits and turning off virtual circuits that have sent more cells than any of the currently inactive virtual circuits. Note that since these rules only turn off virtual circuits when the queue level is rising and turn on virtual circuits when the queue level is falling, the slope of the queue occupancy curve is again controlled so that whenever the queue level exits from the middle region, we can be sure that it will turn around again before overflowing or underflowing the queue, without requiring large queue capacities. However, it should be noted that this prevents the fairness heuristic from providing any real guarantee of fair treatment, since when packet boundaries fall in the 'wrong' part of the buffer occupancy curve, the associated virtual circuits may miss the chance to change state. However, in the absence of synchronization between packet boundaries and the direction of the buffer occupancy curve, one would expect to achieve a reasonable degree of fairness over time, using this approach. The queue capacity between the thresholds affects the degree of fairness that can be achieved. With a larger middle region, each time the queue enters the middle region, it can be expected to remain there for a longer period of time. The longer it stays, the larger the fraction of state changes that are driven by the fairness objective, rather than the objective of simply avoiding overflow and underflow.

As with the original early packet discard with hysteresis algorithm, we can augment this algorithm to ensure that state changes occur with enough frequency that there is some turn-over among the set of active virtual circuits. This will primarily come into play when the buffer is above the upper threshold and falling (slowly) or below the lower threshold and rising, since in the middle region, the fairness heuristic ensures sufficient turn-over.

## 6 Closing Remarks

In most practical situations, packet discard mechanisms are required to cope with overload periods lasting for time periods of tens to hundreds of milliseconds, making long-term fairness issues of limited importance. In these situations, the simple early packet discard with hysteresis algorithm, appears the most appropriate. For situations where long-term fairness is an issue, the additional mechanisms of the fairness heuristic described in the previous section may be justified. At this point, we have no adequate analysis of the degree of fairness achieved by that heuristic and simulation studies will likely be required to resolve this question.

## References


[1] Boyer, P. "A Congestion Control for the ATM," *International Teletraffic Congress Seminar on Broadband Technologies: Architectures, Applications and Performance*, 10/90.

[2] Floyd, Sally and Allyn Romanov. "Dynamics of TCP Traffic over ATM Networks," *Computer Communication Review*, vol. 24, no. 4, 10/94.

[3] Jain, Raj. "Congestion Control and Traffic Management in ATM Networks: Recent Advances and a Survey," draft technical report, Ohio State University, Department of Computer and Information Science, 1/26/95.

[4] Mahdavian, Seyyed. "Resource Management and Bandwidth Allocation in ATM Networks," Doctoral dissertation, Washington University, Department of Electrical Engineering, 12/94.

[5] Turner, Jonathan S. "Managing Bandwidth in ATM Networks with Bursty Traffic," *IEEE Network*, vol. 6, no. 5, September 1992, 50–58.