IMPROVED ANALYSIS OF EARLY PACKET DISCARD

Maurizio Casoni[a] and Jonathan Turner[b*]

[a] University of Bologna, mcasoni@deis.unibo.it

[b] Washington University, jst@cs.wustl.edu.

Appears in *Proceedings of the International Teletraffic Congress*, June 1997, Washington D.C., Copyright held by Elsevier Publishing.

In a previous paper, one of the authors, gave a worst-case analysis for the Early Packet Discard technique for maintaining packet integrity during overload in ATM switches. This analysis showed that to ensure 100% goodput during overload under worst-case conditions, requires a buffer with enough storage for one maximum length packet from every active virtual circuit. This paper refines that analysis, using assumptions that are closer to what we expect to see in practice. Our principal result is that 100% goodput can be achieved with substantially smaller buffers, although the required buffer space can be significant when the link speed is substantially higher than the rate of the individual virtual circuits. These results are validated by comparison with simulation. We also give a simple analysis to determine the amount of buffering needed to bound the probability of buffer overflow and underflow.

1. Introduction

Early packet discard (EPD) [2] is an ATM buffer management technique to ensure high end-to-end throughputs for bursty data applications during overload periods. EPD addresses the problem that arises when transport level packets are discarded at a receiving terminal, because one or more of their component cells were discarded at an overloaded link, within the network. When ATM switches discard cells without regard for the packet-level structure, the effective throughput experienced by users can drop during overload periods, leading to congestion collapse. EPD attempts to avoid this by observing the packet-level structure and discarding entire packets, when overload makes it necessary to do so.

EPD is only one of several mechanisms that have been proposed for handling congestion in ATM networks. In particular, rate-based flow control has now been standardized by the ATM Forum to manage congestion for Available Bit Rate (ABR) traffic streams [3]. While the use of ABR flow control will reduce the need for techniques like EPD, it will not eliminate them, since buffers using ABR flow control can still experience overloads during transient periods, before rate adjustment mechanisms can react to traffic changes,

*This work was supported by the ARPA Computing Systems Technology Office, NEC, NTT, Samsung, Southwestern Bell, Sun Microsystems and Tektronix.

and since these mechanisms will not be applied to Unspecified Bit Rate (UBR) streams. Indeed, it is not even clear that ABR mechanisms will be universally applied, since they are complex enough that users may choose to deploy them only at selected bottleneck links where there is a clear payoff.

A previous paper [4] analyzes packet level discard mechanisms including Early Packet Discard. That analysis is based on worst-case assumptions that over-state the amount of buffering required for high effective throughputs. In this paper, we refine that analysis, using more realistic assumptions and use this to determine the amount of buffering required to achieve 100% goodput (here, we define goodput to be the fraction of the link's capacity that is used to carry complete transport level packets). The analysis naturally divides into two distinct cases. The first applies when the offered load is between one and two times the link rate, and the second when the offered load is more than twice the link rate. Section 2 gives the necessary definitions and modeling assumptions. The analyses for the two cases are given in sections 3 and 4, respectively. In section 5, we give numerical results and also give results that account for the dynamic behavior of on-off bursty sources to estimate the long-term goodput (rather than just the goodput during an overload period).

2. Definitions and Assumptions

We are interested in determining the amount of buffering required to obtain 100% goodput on an overloaded link in which the buffer controller implements early packet discard. In EPD, 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 that packet is discarded.

We consider a homogeneous situation in which r virtual circuits transmit data continually at a normalized rate of λ (that is, λ is the fraction of the link bandwidth required by a single virtual circuit), and define the *overbooking ratio* as $r\lambda$. For an overloaded link $r\lambda > 1$. We also assume that all packets contain ℓ cells and let $k = \lfloor 1/\lambda \rfloor$ be the maximum number of virtual circuits that the link can handle without loss. For simplicity, we will also assume that $1/\lambda$ is an integer; although extension to non-integral values is straightforward, it adds little new information and obscures the key issues. We let B denote the number of cells the buffer can contain, and let b denote the threshold level.

We define a given virtual circuit to be *active* if cells from that virtual circuit are being placed in the queue on arrival. Similarly, we define a virtual circuit to be *inactive* if its cells are being discarded. We assume that the flow of cells on a given virtual circuit is smooth, ignoring the effects of jitter caused by variable delays upstream of the buffer under consideration. Under this assumption, the buffer level rises or falls in a predictable way, depending only on the number of active virtual circuits.

If we observe the number of cells in a queue managed using EPD 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. In this paper, we focus on the amount of buffering required to ensure that the buffer never overflows and never underflows during an overload period. Under these conditions, all of the link's capacity is used to carry complete packets.

Our prior analysis of early packet discard was based on the worst-case assumption that just before every threshold crossing, the first cell of a packet arrived on every virtual circuit, meaning that the next packet boundary was delayed as long as possible after the threshold crossing, leading to wide excursions around the threshold level. While such worst-case synchronization of packet boundaries is possible, it is hardly likely, and would certainly not be expected to persist over an extended period of time. Hence, in this paper we analyze the queue behavior under another simple assumption that more closely reflects what can be expected to happen in practice. Our new modeling assumption is that packet boundaries on different virtual circuits are offset from one another by an equal amount, leading to the occurrence of a packet boundary every $\ell/r\lambda$ cell times. We also assume that threshold crossings fall half-way in between a pair of packet boundaries. Simulation results, reported in section 5, show that these simple assumptions capture the essential performance characteristics of EPD and yield results that are accurate enough to use for estimating buffer requirements.

3. Small Overbooking Ratios

In this section we analyze the performance of EPD when $r\lambda \leq 2$. In particular, we want to determine for what values of b and $B-b$ the buffer will never overflow or underflow. To do this, we simply assume that b and $B-b$ are both very large and determine the maximum excursion above and below the threshold. When $r\lambda \leq 2$, the maximum excursion around the threshold occurs when, at each upward threshold crossing, all r virtual circuits are active. Packet boundaries occur at regular intervals following the threshold crossing. For simplicity, we assume that threshold crossings fall half-way between successive packet boundaries, meaning that packet boundaries will follow threshold crossings by $\ell/2r\lambda$, $3\ell/2r\lambda$, $5\ell/2r\lambda$ and so forth. Figure 1 illustrates how the buffer oscillates around the threshold when $k = 4$ and $r = 6$. The lines at the top indicate the packets flowing on the six virtual circuits. Heavy lines indicate packets that are accepted for transmission, while blank spaces indicate packets discarded by the buffer controller.

At each packet boundary following an upward threshold crossing, the rate at which the buffer level rises gets smaller. As shown in Figure 1, the slope of the curve is $r\lambda - 1$ at the time the threshold is crossed, and each successive packet boundary decreases the slope by λ. When there are exactly k active virtual circuits, the slope is zero and there is no further rise in the buffer level. From this discussion, we can see that the maximum excursion above the threshold is

$$\Delta U = \frac{1}{2}\frac{\ell}{r\lambda}(r\lambda - 1) + \frac{\ell}{r\lambda}\sum_{i=1}^{r-k-1}(r-k-i)\lambda = (1/2)(r-k)^2(\ell/r)$$

Note that this implies $\Delta U \leq (k/4)\ell$. By the worst-case analysis given in [4], the maximum excursion above the threshold can be $(r-k)\ell$, which is always at least four times larger than the "even offset" analysis predicts.

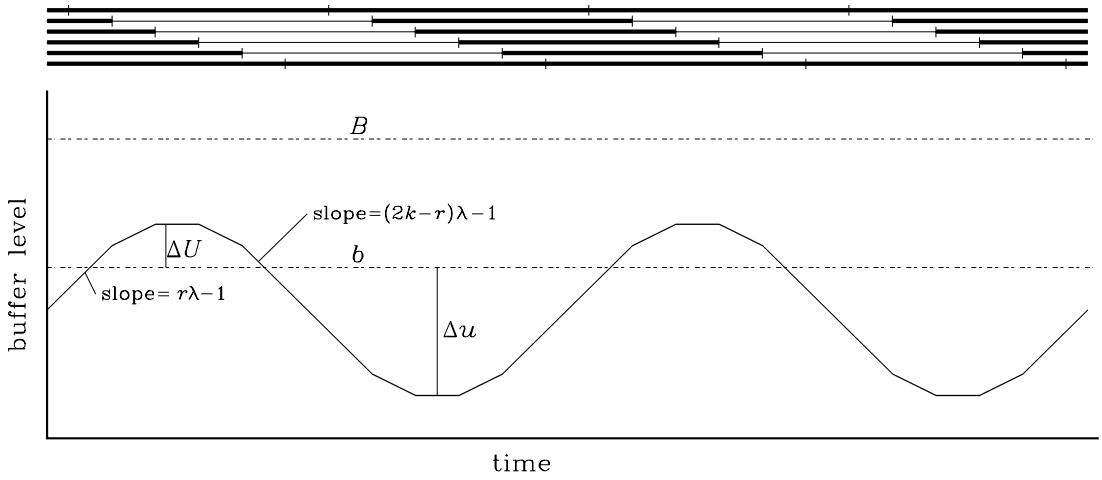


Figure 1. Buffer Range In Absence of Overflow and Underflow

To determine the excursion below the threshold, we need to know the number of active virtual circuits at the downward threshold crossing. This can be determined by inspection from Figure 1 and noting that the even offset assumption on the packet boundaries implies that at the downward threshold crossing, the slope of the buffer occupancy curve has the same absolute magnitude as at the upward threshold crossing. That is, the slope is $1-r\lambda$ and the number of active virtual circuits is $2k-r$. After crossing the threshold, the buffer level continues to drop until all $2k-r$ active virtual circuits complete their packets and new packets start on the inactive virtual circuits. At this point, packet boundaries occur on the inactive virtual circuits at intervals of $\ell/r\lambda$, causing the slope of the buffer occupancy curve to increase, reaching zero when exactly k virtual circuits are active. From the threshold crossing to the first packet boundary at which the slope changes, the buffer level drops by

$$(2k-r+1/2)(r\lambda-1)(\ell/r\lambda) = (2k-r+1/2)(r-k)(\ell/r)$$

From the point where the slope first changes until the buffer reaches its lowest point, the drop in the buffer level is

$$\frac{\ell}{r\lambda}\sum_{i=1}^{r-k-1} i\lambda = (1/2)(r-k-1)(r-k)(\ell/r)$$

So,

$$\Delta u = (1/2)(3k-r)(r-k)(\ell/r)$$

Note that as r increases, Δu rises initially and then falls, reaching a maximum when $r=\sqrt{3}k$. The maximum value of Δu is thus $(2-\sqrt{3})k\ell \approx .27k\ell$. The worst-case analysis in [4] gives a maximum excursion below threshold of $k\ell$. The total buffer range is thus,

$$\Delta U + \Delta u = k(r-k)(\ell/r)$$

and since $k < r < 2k$, the buffer range is $\leq \frac{k\ell}{2}$. It's interesting to note that the time duration of a cycle is exactly $2\ell/\lambda$ which is the time it takes for exactly two packets to arrive. This means that under our assumptions, the same virtual circuits have packets discarded on every cycle. So the lucky virtual circuits (the top and bottom ones in Figure 1) are able to send every packet, while the unlucky ones are able to transmit only half their packets. While in practice, we would expect differences in packet lengths and virtual circuit rates to prevent this pattern from persisting over long periods of time, this does indicate the inherent lack of fairness of the simple EPD algorithm.

4. Large Overbooking Ratios

We now consider the case of overbooking ratios greater than two. As in the last section, we are interested in the maximum excursion around the threshold when buffers are large enough to avoid overflow and underflow. When the overbooking ratio is larger than two, the queue behavior changes qualitatively. In particular, the largest buffer excursion occurs in this case when no virtual circuits remain active at a downward threshold crossing. Thus, following such a threshold crossing, the slope of the buffer occupancy curve is initially -1, and at successive packet boundaries, the slope increases until, when k virtual circuits have become active, the slope is zero and the buffer level drops no further. We can determine the excursion below the threshold using an analysis much like that in the last section. We find

$$\Delta u = \frac{1}{2}\frac{\ell}{r\lambda} + \frac{\ell}{r\lambda}\sum_{i=1}^{k-1}(1 - i\lambda) = (1/2)k^2(\ell/r)$$

Note that when $r \to \infty$, $\Delta u \to 0$ so that Δu is in the range $\left[0, \frac{k\ell}{4}\right]$.

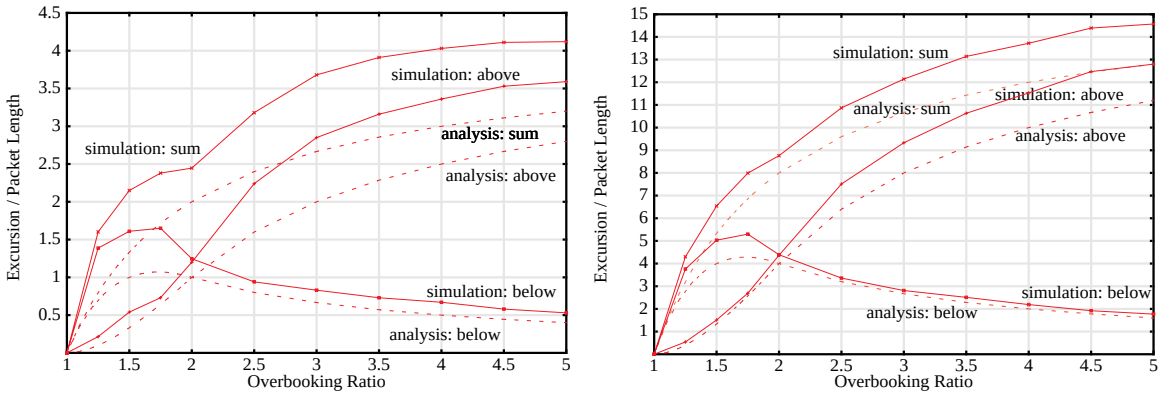
After the buffer level reaches its lowest level, it starts to rise as packet boundaries occur on additional virtual circuits. The time interval during which the buffer occupancy is below the threshold is $2k(\ell/r\lambda)$. Packet boundaries occurring after the buffer rises above the threshold do not immediately cause new virtual circuits to become active, so the slope of the buffer occupancy curve remains constant until the first virtual circuit that became active following the previous downward threshold crossing comes to the end of its packet. This occurs ℓ/λ cell times after it began. At this point the slope changes to $(2k-1)\lambda - 1$ and as additional packet boundaries occur it drops to zero. Thus, the total excursion above the threshold is:

$$\Delta U = \frac{\ell}{\lambda} - (2k - 1/2)\frac{\ell}{r\lambda} + \frac{\ell}{r\lambda}\sum_{i=1}^{k-1}[(2k - i)\lambda - 1] = (1/2)(2r - 3k)k(\ell/r)$$

When $r \to \infty$, $\Delta U \to k\ell$ so ΔU lies within the range $\left[\frac{k\ell}{4}, k\ell\right]$. Adding Δu and ΔU gives a total buffer range of $(r-k)k(\ell/r)$. The worst-case analysis of [4] gives a maximum buffer range of $r\ell$.

5. Numerical Results

Figure 2 shows results from our analysis. Two cases are shown, one with $k = 4$ and one with $k = 16$. In the simulations, packet lengths are fixed, as in the analysis, but the relative phases of the packets in different virtual circuits were randomized at the



(a) k = 4 (b) k = 16

Figure 2. Normalized excursion above and below the threshold

start of each simulation run and the maximum excursion above and below threshold over the entire run was recorded. The values plotted are the average values from multiple simulation runs (more than 100 runs per data point).

Notice that when the overbooking ratio is ≤ 2, the maximum excursion above the threshold is always smaller than the excursion below the threshold and that when $r\lambda \geq 2$ this is reversed. Also notice that the analysis tracks the simulation results most closely when k is large. This is to be expected, since for small k, there is a greater likelihood that randomly selected phase offsets will differ substantially from the idealized even offset assumption of the analysis.

The analysis reported here gives much better estimates of the required buffer size than the worst-case analysis, allowing the buffers and the required threshold to be more closely tailored to the real needs. For the two cases shown, the worst-case analysis requires a total buffer size of 20 and 80 when the overbooking ratio is five. Notice however that when k is large (meaning that the peak virtual circuit rate is much smaller than the link rate), the amount of buffering required can still be substantial. This can be problematical for high speed links, for which cell buffers can be an expensive resource. As one example, a buffer controller for a 2.4 Gb/s link will require a buffer of 4.8 MB to ensure 100% goodput in the presence of an overload caused by virtual circuits carrying 4 KB packets with a transmission rate of 2 Mb/s per virtual circuit. Of course, it's unlikely that a gigabit link carrying traffic from bursty virtual circuits with peak rates of 2 Mb/s will become overloaded in the first place. In general, overload is most likely to arise in those cases when k is small, allowing one to get by with much smaller buffers.

Suppose we have a total of n virtual circuits that are each actively sending packets with probability p, and that we're willing to tolerate a probability of ϵ that the link experiences either overflow or underflow. Let K be the smallest integer for which

$$\sum_{r=K+1}^{n} \binom{n}{r} p^r (1-p)^{n-r} \leq \epsilon$$

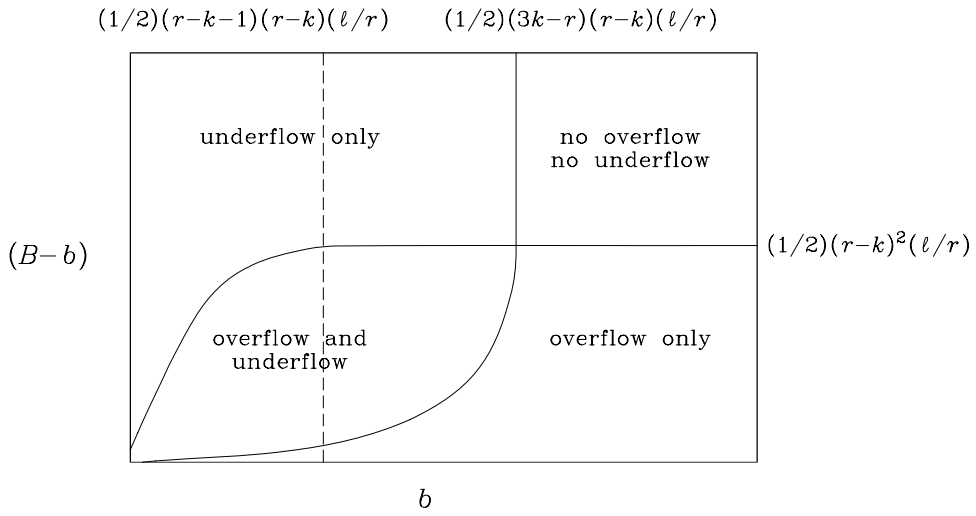


Figure 3. Idealized Goodput Map for $r\lambda \leq 2$

Then we require that the buffer be large enough so that for all $r \leq K$, the excursion around the threshold does not lead to underflow or overflow. For example, if we have 30 virtual circuits, each with a peak rate of 1/4 and a peak-to-average ratio of 10:1 (that is, $p = 0.1$) and we let $\epsilon = 0.05$, we find that $K = 6$, meaning we only need a buffer large enough to avoid overflow and underflow when $r \leq 6$. It is sufficient in this case to have $b = \ell$ and $B = 1.33\ell$. As another example, suppose we have 120 virtual circuits, each with a peak rate of 1/16, $p = 0.1$, $\epsilon = 0.05$. In this case, $K = 18$, so we need a buffer large enough to avoid overflow when $r \leq 18$. In this case, $b = 1.67\ell$ and $B = 1.78\ell$ are enough. In both of these examples, the average offered load is 75% of the link capacity, so the average carried load is at least 71%. Thus with even fairly modest buffer sizes, EPD can achieve reasonably high throughput for bursty traffic.

6. Closing Remarks

In a companion paper [1], we have studied the performance of early packet discard when the buffer is not large enough to ensure 100% goodput. Depending on the values of b and B, the buffer will experience underflow, overflow or some combination of the two. This is shown schematically in Figure 3 for the case of small overbooking ratios. Figure 4 shows a computed version of this "goodput map" using the analysis in [1]. Notice that while decreasing either b or $B - b$ from the values required for 100% goodput causes a deterioration in performance, the deterioration is not monotonic. Indeed, we get better performance for very small buffers than for intermediate values. This apparently anomalous behavior is characteristic of early packet discard. The explanation is that when the buffer is small enough to allow overflow (or underflow), the slope of the buffer occupancy curve at subsequent downward (resp. upward) threshold crossings is reduced, causing a smaller swing below (above) threshold. Because of this, reducing the amount of buffering can actually lead to an improvement in goodput. Similar maps can be produced

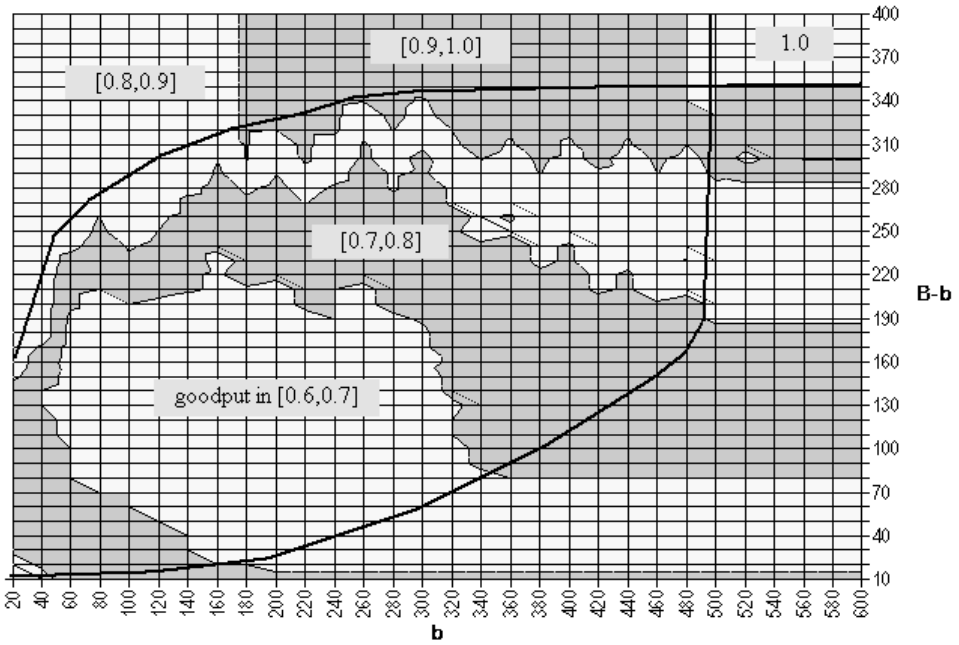


Figure 4. Computed Goodput Map for $k = 20$, $r = 35$, $\ell = 100$

for large overbooking ratios. This behavior has been confirmed via simulation studies which typically show goodputs within a few percent of the values predicted by analysis.

Our earlier work also considers variants of early packet discard that provide 100% goodput with much smaller buffers than are needed for early packet discard (one or two packets total, under worst-case assumptions) and have better characteristics with respect to fairness. See [4] for details.

REFERENCES

1. Casoni, Maurizio and Jonathan Turner. "On the Performance of Early Packet Discard," to appear in *IEEE Journal on Selected Areas in Communications*. Also available from Washington University Computer Science Department as WUCS-96-13, 4/96.
2. Floyd, Sally and Allyn Romanow. "Dynamics of TCP Traffic over ATM Networks," *IEEE Journal on Selected Areas in Communications*, 5/95.
3. Jain, Raj. "Congestion Control and Traffic Management in ATM Networks: Recent Advances and a Survey," *Computer Networks and ISDN Systems*, 10/96.
4. Turner, Jonathan S. "Maintaining High Throughput During Overload in ATM Switches", *Proceedings of Infocom*, 3/96.