# On the Performance of ATM-UBR with Early Selective Packet Discard [*]

Kangsik Cheon[†] and Shivendra S. Panwar
Center for Advanced Technology in Telecommunications
Polytechnic University
5 Metrotech Center, Brooklyn, New York 11201
Email: {kcheon, panwar}@kanchi.poly.edu

## Abstract

*We investigate the performance of the Early Packet Discard(EPD) and the Early Selective Packet Discard(ESPD) scheme that we have previously proposed*[12]. *In an effort to reduce the complexity while maintaining good performance, we pick two special cases of ESPD and compare their performance with that of EPD. For performance evaluation purposes, the effective throughput and fairness index of these schemes was determined through a simulation study. We observed that the ESPD scheme improves effective throughput over EPD by up to 16% with our network model. We also found that ESPD is more effective in alleviating TCP's unfairness among connections which have different roundtrip times. A major factor causing throughput degradation of EPD was determined to be the synchronization of TCP windows.*

## 1 Introduction

For applications like non-real-time data traffic, the ATM Forum has defined two different services, available bit rate(ABR) service and unspecified bit rate(UBR) service [1]. ABR service has attracted much attention as a research topic in recent years and is expected to deliver better quality of service than UBR. But, considering the fact that the implementation complexity and cost of ABR is significantly higher than UBR, and that other low-cost technologies for high speed networks like gigabit Ethernet are developing rapidly, there is interest in investigating the possibility of using UBR services as an interim low-cost alternative to ABR service [2-6].

One of the most popular schemes for UBR's throughput improvement is the early packet discard(EPD) scheme[6]. EPD discards BOM(Begin of Message) cells and the following cells of the same packet after the buffer occupancy exceeds a threshold, thus emulating packet discard. A detailed description of the operation of EPD is given in Section 3.

EPD has been shown to lead to significant performance improvement for TCP connections[2, 6]. The improvement is mainly due to the fact that EPD discards cells selectively. But when it comes to packet discard, EPD discards packets *randomly* so that it spreads packet losses over many sessions. This randomness triggers timeouts across most TCP sessions and causes TCP synchronization, resulting in underloaded network resources.

Based on this observation, we have proposed a new packet discarding scheme, the *Early Selective Packet Discard(ESPD)*[12], which has an added packet selection mechanism compared to EPD. This allows ESPD to force selected sources to shrink their window size via the implicit feedback control of TCP.

In this paper, we test the performance of both schemes by varying a parameter at a time and evaluated both schemes by comparing effective throughput and a fairness index.

Section 2 describes the simulation network model and the parameter values that we used. Section 3 discusses how each packet discarding scheme works. In Section 4, we present the simulation results and a performance comparison. Our concluding remarks are in Section 5.

## 2 Network Model

In this section, we describe the simulation environment used in our simulation of TCP over ATM-UBR. Our simulation tool is based on OPNET(OPtimized

[*]This work is supported in part by Sumitomo Electric Industries, Inc., and the New York State Center for Advanced Technology in Telecommunications Polytechnic University, Brooklyn, New York.

[†]Kangsik Cheon is also a member of technical staff at Korea Telecom, Seoul, Korea.

Network Engineering Tools) which is provided by MIL 3, Inc. OPNET is a network simulator capable of simulating large communications networks with detailed protocol modeling and performance analysis.

All simulations presented in this paper are performed on the parking lot network configuration shown in Fig. 1 The network consists of 4 ATM switches, 15 source nodes and 15 destination nodes. In the figure, *src* refers to the source node whose application layer behaves as a TCP packet source only and *dest* refers to the destination node whose application layer behaves as a receiver only. The source node *i* identified by *src_i* communicates with the destination node *i* identified as *dest_i*. All application layers use TCP to communicate with their peer layers at the destination node. Each source node sends out as many packets as permitted by the sliding window control of TCP. Our TCP implementation does not use coarse-grain timers, but we restricted the minimum TCP RTO(Retransmission Timeout) to a certain value as described at the end of this section.

The IP service rate is set to 10,500 packets/sec for a packet size of 500 bytes and to 4,000 packets/sec for a packet size of 1,500 bytes. The 500 byte and 1,500 byte packets are broken into 12 ATM cells and 33 ATM cells, respectively. The IP buffer size is set to infinity to ensure that there is no packet loss at the IP layer.

The function of the ATM adaptation layer of our simulation is very simple. It adds a 14 byte overhead to each IP datagram, and adjusts the size to multiples of 48 bytes. It then does segmentation, adds a 5 byte overhead to each cell, sets an EOM(End of Message) bit at the last cell corresponding to each packet and forwards the cells down to the ATM layer. When an end station receives cells from a peer node and needs to forward them to the upper layer, it performs the inverse functions. Each of the links is full duplex with a bandwidth capacity of 155.52 Mbps, corresponding to the capacity of a SONET STS-3c link. The ATM switch of our model network is a simple output buffered switch that just reads VPI/VCI information of arriving cells and forwards them to the proper output port.

Since the sample TCP model of OPNET version 2.4 was based on RFC 793, we added the key components of the congestion control algorithm of TCP, such as the slow-start algorithm, congestion-avoidance mechanism, fast retransmit and fast recovery algorithm and a delayed acknowledgment scheme [9, 10]. In the TCP implementation of OPNET version 2.4, each packet being transmitted from the source node has its own RTO timer so that once a timeout occurs, many subse-

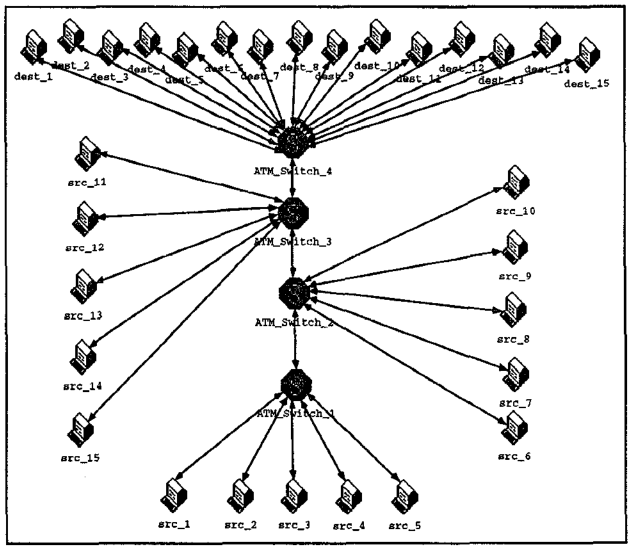

Figure 1: Simulation Network

quent timeouts generally follow. We change this timer scheme so that the resulting TCP sessions have one RTO timer each, making it consistent with common implementations. Because we intended to simulate the *ftp* application, we model a greedy source which generates and injects packets down to the TCP layer as soon as TCP moves the sliding window forward. Each TCP connection has its own corresponding ATM virtual circuit. Other parameters which are not mentioned above are as follows:

- Minimum TCP RTO(Retransmission Timeout): 50 *msec* (one fourth of Solaris default value)
- Maximum waiting time for a delayed ACK: 12.5 *msec*(one fourth of Solaris default value)
- Propagation delay of all the links: 1 *msec*
- Simulation time: 3 *sec*
- Maximum window size of TCP: 64 *Kbytes*
- Zero TCP processing time
- Buffer size of ATM switch: 2,000 cells/port

## 3 Packet Discard Schemes

### 3.1 Early Packet Discard(EPD)

EPD was proposed in [6] for ATM in order to discard an entire packet prior to buffer overflow, so that corrupted packets will not be transmitted by the switches.

The EPD mechanism sets a certain buffer threshold, and as soon as the queue length exceeds the threshold, the ATM switch is *ready to discard* incoming cells. The meaning of being *'ready to discard'* is to wait for incoming BOM(Begin of Message) cell instead of immediately discarding any incoming cell. After that, whenever the switch sees a BOM cell, and as long as the queue length is above the threshold, it drops the cell. All the subsequent cells of the packet followed by the dropped BOM cell are discarded. By doing so, and by setting the proper threshold, we can emulate packet discarding almost perfectly. For our simulation, we vary the threshold for packet discarding from 60 % to 95 % of buffer size.

## 3.2 Early Selective Packet Discard(ESPD)

The packet discards in EPD are random. This produces the sustained congestion period problem and the synchronized window expansion-shrinkage problem[12]. In an earlier work, we identified some situations where EPD synchronizes TCP sessions, and analyzed the mechanism of TCP synchronization for EPD[13]. These two problems reflect the fact that all the sessions share low and high activity periods together. *'Low activity'* means the aggregated traffic coming from all the sessions are lower than the output link capacity.

In order to avoid the TCP window synchronization problem of EPD, and based on a *first-come-first-catch* policy, ESPD makes the status of session red when congestion occurs, and it keeps discarding the packets from the red status sessions. The first-come-first-catch policy implies that the VPI/VCI of a BOM cell which comes earlier turns it into a red status session. Dropped packets make sessions shrink their TCP window size. On the other hand, the remaining intact sessions which did not shrink their TCP window can contribute to keep the output link busy resulting in better throughput. Some time later, the intact sessions which keep increasing their window sizes may become the major contributors to the next congestion epoch. This means those sessions are more likely to be turned into the red status by the first-come-first-catch policy. Therefore, sessions multiplexed onto one link tend to use the shared link capacity alternately under the ESPD policy.

The original ESPD [12] has a drop timer and three thresholds which are: the high threshold, the low threshold and threshold on the number of red status sessions. The high threshold is designed to make the status of VPI/VCIs red and the low threshold is for releasing VPI/VCI's from the red status. The purpose of the threshold on the number of red status sessions is to place a limit on the number of VPI/VCIs in red status. The drop timer is included for maintaining fairness across sessions.

After more comprehensive simulations, we have made two major changes on the original ESPD. One is to set the drop timer value to infinity, which in effect eliminates the timer. This followed our conclusion that the drop timer does not improve fairness notably, hence such an improvement does not compensate for increased implementation complexity. The other major change is not to use the threshold on the number of red status sessions. Even without this threshold, a high threshold and a low threshold together perform well to select a fraction of the sessions for packet discarding. After these changes, the new ESPD scheme works as follows.

If the buffer occupancy exceeds the high threshold, we change the status of VPI/VCI of BOM cell arrivals to red. All the subsequent cells, except the EOM(End of Message) cell, are discarded even if we have available buffers. We make the status of VPI/VCIs red on this *first-come-first-catch* policy because it is likely to find the BOM cells of high activity sessions first during a congestion epoch. In case the buffer becomes full, we discard all incoming cells, make the status of corresponding VPI/VCIs red, and keep discarding all subsequent cells as long as the status of VPI/VCI is red. We release the captured VPI/VCI from red status if the queue length become less than the low threshold and the corresponding EOM cell arrives. This VPI/VCI releasing policy makes us select as many sessions as we need to return an initially increasing queue length below the low threshold. Since EPD releases a VPI/VCI from the red status upon EOM cell arrival, it cannot concentrate packet discards on selected sessions. So, EPD spreads packet losses across many active sessions. For TCP sessions, this causes a low activity period as these sessions shrink their windows simultaneously, with corresponding loss of throughput. The following is the pseudocode for ESPD.

```
We have a cell arrival at an ATM switch:
if the cell's VPI/VCI status is red
   if the cell is an EOM cell
      if queue length < buffer_size
         insert the cell into buffer
      else
         discard the cell
      if queue length < low threshold
         release the VPI/VCI from red status
```

```
        else
            discard the cell
    else
        if queue_length < high threshold
            insert the cell into buffer
        else if (BOM cell or (the buffer is full))
            discard the cell
            capture the VPI/VCI into red status
        else
            insert the cell into buffer
```

## 4 Simulation Results

Tables 1 to 8 are the simulation results for EPD and ESPD. For the performance evaluation of EPD and ESPD, we considered three factors: TCP synchronization, effective throughput and fairness. TCP synchronization was evaluated qualitatively by observing window size synchronization. The effective throughput is defined as the total number of packets which arrive at their destination in a given interval of time. We counted duplicate packets generated as a result of packet retransmission as one packet. The fairness index is calculated as follows [4].

$$Fairness = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2} \qquad (1)$$

In the above equation, $x_i$ is the effective throughput of *i-th* session and $n$, the number of sessions, is 15 in our simulations. We denote the roundtrip propagation delay as $\tau$. According to the value of $\tau$, we divide the 15 TCP sessions into 3 groups: (1) sessions 1 to 5; (2) sessions 6 to 10; (3) sessions 11 to 15. The effective throughput of a session group is the total aggregated number of packets received by 5 destination nodes of each group.

Varying the seed, we ran five simulations for each parameter set in order to enhance the accuracy of simulation results. The effective throughput and fairness index in Table 1 to 8 are the averages over these 5 simulations. The simulation time was 3 seconds for every simulation but we collected results after 0.5 *sec*, in order to disregard the initial transient period. The maximum effective throughput indicated in those tables is the maximum number of packets that can be carried by an OC-3 link for 2.5 seconds.

### 4.1 TCP Synchronization

We plot the window-size of the 15 sessions as a function of time for each discarding scheme in Fig. 2 and 3. As we can see from Fig. 2, EPD apparently synchronizes the TCP sessions. That is, many TCP sessions shrink window-sizes at the same time when network congestion occurs, go through a low activity period together, and expand window-sizes resulting in another congestion. On the other hand, Fig. 3 shows that ESPD makes the window expansion and shrinkage asynchronous. This ability of ESPD results in significant effective throughput improvement over EPD.

The TCP desynchronization of ESPD mainly comes from the fact that ESPD does not release the captured sessions in the drop-list as long as the queue length remains above the lower threshold. ESPD keeps discarding packets from those sessions, freeing buffer space for the packets from other sessions. As a results, some sessions suffer packet discard and shrink their TCP window sizes, while other sessions remain intact due to the freed buffer space and keep increasing their TCP window size.

### 4.2 Effective Throughput

A summary of the EPD simulation is listed in Tables 1 and 2 for 500 byte and 1,500 bytes packets, repectively. From these tables, we notice that the throughput performance of EPD is not sensitive to buffer threshold. Within the threshold range of 70 to 95% of the total buffer size, the best effective throughput achieved at 90% gets only an 1% improvement over the worst throughput at 70% for 500 byte packets. For 1,500 byte packet, the best throughput is 3% above the worst throughput within the same range.

Tables 3 and 4 represent the corresponding results for ESPD, in which the high threshold is fixed to 95% of buffer size. With the fixed high threshold, we varied the low threshold only. We can see that the best throughput was obtained at a low threshold of 85%. This best throughput in Table 3 is a 16% improvement over the best throughput of EPD in Table 1. With a 1,500 byte packet, the ESPD's best is a 3% improvement over the EPD's best. In order to show that our improvement is statistically significant, we calculated the standard deviation for the 5 simulation runs.

We have additional ESPD simulation results in Tables 5 and 6. These ESPD simulations were performed with a fixed high threshold set to 100% of buffer size. In other words, we don't discard any incoming cell before the buffer becomes full. This scheme reduces the implementation complexity because it actually has only one threshold which needs to be set. With this high threshold, the ESPD in Table 5 still get a 14% improvement over EPD's best in Table 1. For the case of 1,500 byte packets, it improved the throughput by

| | Threshold | 95 % | 90 % | 80 % | 70 % | 60 % |
|---|---|---|---|---|---|---|
| Effective Throughput | Sessions 1- 5 : $\tau = 10$ msec | 12,429 | 11,873 | 11,601 | 11,448 | 11,945 |
| | Session 6-10 : $\tau = 8$ msec | 16,363 | 16,311 | 17,692 | 15,701 | 15,354 |
| | Sessions 11-15 : $\tau = 6$ msec | 30,262 | 31,348 | 29,781 | 31,574 | 29,261 |
| Sum* (Percentile Value) | | 59,054 (77%) | 59,532 (78%) | 59,074 (77%) | 58,723 (77%) | 56,560 (74%) |
| [ Standard Deviation ] | | [837] | [848] | [1,015] | [414] | [1,366] |
| Fairness Index | | 0.837 | 0.802 | 0.830 | 0.791 | 0.834 |

* Maximum Value is 76,415.

Table 1: Performance of EPD (500 Bytes Packet ).

| | Threshold | 95 % | 90 % | 80 % | 70 % | 60 % |
|---|---|---|---|---|---|---|
| Effective Throughput | Sessions 1- 5 : $\tau = 10$ msec | 5,618 | 6,016 | 6,014 | 5,962 | 5,768 |
| | Session 6-10 : $\tau = 8$ msec | 7,477 | 6,842 | 7,107 | 6,937 | 6,570 |
| | Sessions 11-15 : $\tau = 6$ msec | 12,693 | 12,634 | 12,132 | 12,204 | 12,116 |
| Sum* (Percentile Value) | | 25,788 (93%) | 25,492 (92%) | 25,253 (91%) | 25,103 (90%) | 24,454 (88%) |
| [ Standard Deviation ] | | [174] | [177] | [147] | [162] | [247] |
| Fairness Index | | 0.867 | 0.860 | 0.884 | 0.875 | 0.864 |

* Maximum Value is 27,787.

Table 2: Performance of EPD (1,500 Bytes Packet ).

| | Low Threshold | 90 % | 85 % | 80 % | 70 % | 60 % |
|---|---|---|---|---|---|---|
| Effective Throughput | Sessions 1- 5 : $\tau = 10$ msec | 17,332 | 14,929 | 17,243 | 16,830 | 13,559 |
| | Session 6-10 : $\tau = 8$ msec | 19,544 | 20,479 | 19,501 | 17,769 | 18,107 |
| | Sessions 11-15 : $\tau = 6$ msec | 33,791 | 36,151 | 33,797 | 33,733 | 33,289 |
| Sum * (Percentile Value) | | 70,673 (92%) | 71,559 (94%) | 70,541 (92%) | 68,332 (89%) | 64,955 (85%) |
| [ Standard Deviation ] | | [ 1,182 ] | [ 555 ] | [ 516 ] | [ 1,605 ] | [ 608 ] |
| Fairness Index | | 0.859 | 0.798 | 0.858 | 0.802 | 0.774 |

*Maximum Value is 76,415.

Table 3: Performance of ESPD with a fixed High Threshold = 95 % (500 Bytes Packet ).

| | Low Threshold | 90 % | 85 % | 80 % | 70 % | 60 % |
|---|---|---|---|---|---|---|
| Effective Throughput | Sessions 1- 5 : $\tau = 10$ msec | 6,579 | 6,983 | 6,423 | 6,496 | 7,024 |
| | Session 6-10 : $\tau = 8$ msec | 7,279 | 7,451 | 7,343 | 7,223 | 6,937 |
| | Sessions 11-15 : $\tau = 6$ msec | 12,724 | 12,278 | 12,839 | 12,634 | 11,842 |
| Sum * (Percentile Value) | | 26,582 (96%) | 26,712 (96%) | 26,615 (96%) | 26,353 (95%) | 25,803 (93%) |
| [ Standard Deviation ] | | [124] | [410] | [188] | [169] | [315] |
| Fairness Index | | 0.881 | 0.866 | 0.864 | 0.839 | 0.864 |

*Maximum Value is 27,787.

Table 4: Performance of ESPD with a fixed High Threshold = 95 % (1,500 Bytes Packet ).

| | Low Threshold | 95 % | 90 % | 80 % | 70 % | 60 % |
|---|---|---|---|---|---|---|
| Effective Throughput | Sessions 1- 5 : $\tau = 10$ msec | 18,710 | 19,522 | 17,506 | 21,058 | 18,919 |
| | Session 6-10 : $\tau = 8$ msec | 23,426 | 23,927 | 25,388 | 21,661 | 22,991 |
| | Sessions 11-15 : $\tau = 6$ msec | 26,052 | 27,141 | 22,794 | 24,298 | 23,186 |
| Sum * (Percentile Value) | | 68,188 (89%) | 70,590 (92%) | 65,688 (86%) | 67,017 (88%) | 65,096 (85%) |
| [ Standard Deviation ] | | [ 866 ] | [ 1,620 ] | [ 793 ] | [ 1,966 ] | [ 1,252 ] |
| Fairness Index | | 0.912 | 0.902 | 0.917 | 0.910 | 0.887 |

* Maximum Value is 76,415.

Table 5: Performance of ESPD with a fixed High Threshold =100 % (500 Bytes Packet ).

| | Low Threshold | 95 % | 90 % | 80 % | 70 % | 60 % |
|---|---|---|---|---|---|---|
| Effective Throughput | Sessions 1- 5 : $\tau = 10$ msec | 8,369 | 7,902 | 9,176 | 9,187 | 8,920 |
| | Session 6-10 : $\tau = 8$ msec | 8,810 | 8,950 | 8,878 | 9,628 | 9,361 |
| | Sessions 11-15 : $\tau = 6$ msec | 8,072 | 9,614 | 8,480 | 7,174 | 6,870 |
| Sum * (Percentile Value) | | 25,251 (91%) | 26,466 (95%) | 26,534 (95%) | 25,989 (94%) | 25,251 (91%) |
| [ Standard Deviation ] | | [296] | [233] | [330] | [201] | [251] |
| Fairness Index | | 0.941 | 0.924 | 0.938 | 0.915 | 0.909 |

* Maximum Value is 27,787.

Table 6: Performance of ESPD with a fixed High Threshold =100 % (1,500 Bytes Packet ).

| | High Threshold | 55 % | 60 % | 70 % | 80 % | 90 % |
|---|---|---|---|---|---|---|
| Effective Throughput | Sessions 1- 5 : $\tau = 10$ msec | 14,505 | 14,795 | 16,998 | 15,323 | 13,518 |
| | Session 6-10 : $\tau = 8$ msec | 16,774 | 19,265 | 18,615 | 17,424 | 17,913 |
| | Sessions 11-15 : $\tau = 6$ msec | 35,365 | 34,474 | 32,611 | 31,470 | 31,302 |
| Sum * (Percentile Value) | | 66,644 (87%) | 68,534 (90%) | 68,224 (89%) | 64,217 (84%) | 62,733 (82%) |
| [ Standard Deviation ] | | [ 1,277 ] | [ 1,785 ] | [ 1,346] | [ 1,629 ] | [ 2,538 ] |
| Fairness Index | | 0.805 | 0.822 | 0.845 | 0.783 | 0.791 |

* Maximum Value is 76,415.

Table 7: Performance of ESPD with a fixed Low Threshold = 50 % (500 Bytes Packet ).

| | High Threshold | 55 % | 60 % | 70 % | 80 % | 90 % |
|---|---|---|---|---|---|---|
| Effective Throughput | Sessions 1- 5 : $\tau = 10$ msec | 6,089 | 6,440 | 6,980 | 6,593 | 6,852 |
| | Session 6-10 : $\tau = 8$ msec | 7,312 | 6,781 | 6,828 | 6,555 | 7,236 |
| | Sessions 11-15 : $\tau = 6$ msec | 12,059 | 12,680 | 12,142 | 11,955 | 11,227 |
| Sum * (Percentile Value) | | 25,400 (91%) | 25,901 (93%) | 25,950 (93%) | 25,593 (92%) | 25,315 (91%) |
| [ Standard Deviation ] | | [456] | [272] | [231] | [193] | [268] |
| Fairness Index | | 0.858 | 0.850 | 0.851 | 0.831 | 0.872 |

* Maximum Value is 27,787.

Table 8: Performance of ESPD with a fixed Low Threshold = 50 % (1,500 Bytes Packet ).

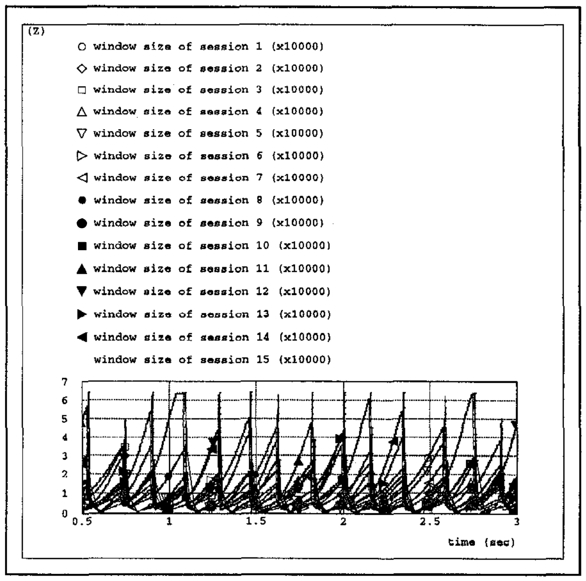

Figure 2: Session window sizes for EPD as a function of time. (Threshold = 95%, Packet Size = 500 bytes)

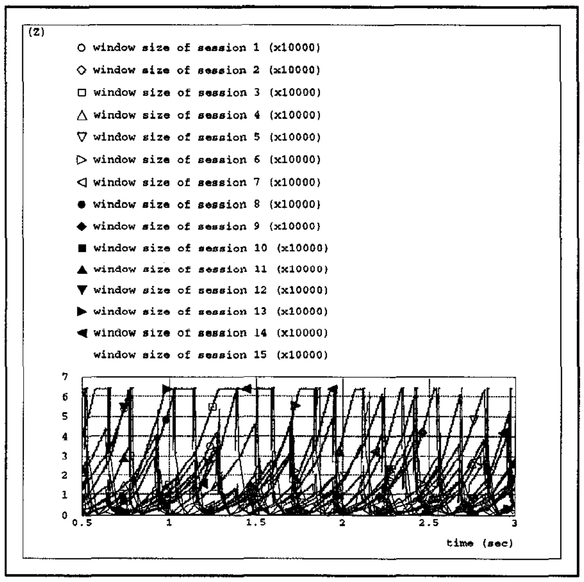

Figure 3: Session window sizes for ESPD as a function of time. (High Threshold = 95%, Low Threshold = 90%. Packet Size = 500 bytes)

2 to 3% only. One issue of potential concern with this version of ESPD is that it produces useless cells from the head part of these corrupted packets. But the amount of useless cells seems not to be significant, because these cells come from only the first lost packet among many lost packets in a session.

The last simulation set we ran is summarized in Tables 7 and 8. While the preceding two versions of ESPD have a fixed high threshold, this ESPD has a fixed low threshold which is 50 % of buffer size. We varied the high threshold from 55% to 90%. The best throughput was found at the high threshold of about 60% buffer size in Tables 7 and 8. The effective throughput degrades as the high threshold is increased further. We might expect that we get more throughput as the high threshold goes up, because the increased high threshold means a bigger effective buffer size. However, we can conclude, for this example network, that the best throughput of ESPD is achieved when there is a proper gap(10% in our case) between the high threshold and the low threshold. Another conclusion is that the ESPD delivers up to a 16% throughput improvement over EPD.

## 4.3 Fairness

Comparing the fairness index in tables 1 to 8 which is calculated over 15 TCP sessions by using equation 1, we can see that ESPD does not sacrifice fairness for increased throughput. In fact, the ESPD in Tables 5 and 6 show significantly improved fairness index over EPD.

Other than using the fairness index for the fairness performance comparison, we also explore another aspect of fairness between session groups. From Table 1, we can find that there is considerable throughput unfairness between the different session groups with EPD. Although this unfairness is also apparent in Table 3, ESPD with a low threshold of 90% buffer size improved the throughput of sessions(1-5) by *39.4%* over that of EPD, while it improved the throughput of sessions(11-15) by only *11.7%* over that of EPD. This means ESPD provides more throughput enhancement to a long roundtrip time session rather than to a short roundtrip time session. Considering the fact that sessions(1-5) have the longest roundtrip time and sessions(11-15) experience the shortest roundtrip delay, we can say that ESPD helps to alleviate TCP's generic unfairness[10] resulting from the window evolution mechanism. This kind of fairness enhancement is more pronounced in Table 5. In Table 5, sessions(11-15) actually have reduced their throughput as compared to EPD even though the total throughput has increased. This kind of fairness improvement is even more marked for the corresponding case(Table 6) with 1,500 byte packets.

This better fairness of ESPD results from the fact that ESPD has session selection capability owing to a "first-come-first-catch" policy and dual thresholds. The "first-come-first-catch" policy makes ESPD more likely to catch high activity sessions first and a low threshold makes ESPD keep discarding packets from those sessions, freeing buffer space for the packets from other low activity sessions. As a result, the low activity sessions, with long roundtrip times, are more likely to survive the congestion periods than the high activity sessions with short roundtrip times.

## 5 Conclusions

In this paper, we presented the performance of two packet discarding schemes, EPD and ESPD, varying parameter values. Our simulation results show that ESPD improved the effective throughput over EPD by 3% to 16% for our network configuration. We attribute the throughput degradation of EPD to the nature of random packet discard which causes TCP synchronization. This simulation work also demonstrates that ESPD improve the fairness between session groups which have different roundtrip times, especially with a high threshold of 100 % of buffer size and a low threshold of 90 % of buffer size. After closer investigation of ESPD performance, it appears that the best throughput is achieved when there is a gap of 10% between the high threshold and low threshold.

## References

[1] ATM Forum, "Traffic Management Specification Version 4.0," April, 1996. available as *ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0056.000.ps*

[2] Chien Fang, Helen Chen and Jim Hutchins, "A simulation study of TCP performance in ATM networks," *Proc. IEEE GLOBECOM '94*, pp. 1217-1223, Nov. 1994.

[3] H. Li, K.-Y. Siu, H.-Y. Tzeng, C. Ikeda and H. Suzuki, "Performance of TCP over UBR service in ATM networks with per-VC early packet discarding schemes," *Proc. the International Phoenix Conference on Computers and Communications*, pp. 350-357, Mar. 1996.

[4] R. Goyal, G. Jain, S. Fahmy, S. Fahmy and S Kim, "Performance of TCP over UBR+," AF-TM 96-1269, Oct. 1996.

[5] H. Li, K. Siu, H. Tzeng, C. Ikeda and H. Suzuki, "A simulation Study of TCP Performance in ATM Networks with ABR and UBR Services," *Proc. IEEE INFOCOM'96*, pp. 1269-1276, Mar. 1996.

[6] Allyn Romanow and Sally Floyd, "Dynamics of TCP Traffic over ATM Networks," *IEEE Journal of Selected Areas in Communications*, vol. 13, no. 4, May 1995.

[7] V. Jacobson. "Congestion Avoidance and Control," *Proc. ACM SIGCOM'88*, pp. 314-329, August 1988.

[8] W. R. Stevens, *TCP/IP Illustrated*, volume 1. Addison-Wesley, 1994.

[9] G. Armitage and K. Adams, "Packet Reassembly during Cell Loss," *IEEE Network Mag.*, vol. 7, no. 5, pp. 26-34, Sept. 1993.

[10] T. V. Lakshman and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss," *IEEE/ACM Transactions on Networking*, vol. 5, no. 3, pp. 336-350, June 1997.

[11] S. Shenker, L Zhang and D. D. Clark, "Some Observations on the Dynamics of a Congestion Control Algorithm," *Computer Communication Review*, pp. 30-39, October 1990.

[12] K. Cheon and S. S. Panwar, "Early Selective Packet Discard for Alternating Resource Access of TCP over ATM-UBR," *Proc. of the 22nd IEEE Annual Conference on Computer Networks(LCN'97)*, pp. 306-316, Minneapolis, Nov. 1997.

[13] K. Cheon and S. S. Panwar, "On the Performance of ATM-UBR with Early Selective Packet Discard: Extended Version" *CATT Tech. Report*, Center for Advanced Technology in Telecommunications. Polytechnic University, January 1998.