Improved Virtual Queueing and Dynamic EPD Techniques for TCP over ATM

Yuan Wu Kai-Yeung Siu Wenge Ren
Massachusetts Institute of Technology

Abstract—
It is known [14] that TCP throughput can degrade significantly over UBR service in a congested ATM network, and the early packet discard (EPD) technique has been proposed to improve the performance. However, recent studies [11], [16] show that EPD cannot ensure fairness among competing VCs in a congested network, but the degree of fairness can be improved using various forms of fair buffer allocation techniques. Based upon the work in [16], we propose an improved scheme that utilizes only a single shared FIFO queue for all VCs and admits simple implementation for high speed ATM networks. Our scheme achieves nearly perfect fairness of throughput among multiple TCP connections, comparable to the expensive per-VC queuing technique. Analytical and simulation results are presented to show the validity of this new scheme and significant improvement in performance as compared with existing fair buffer allocation techniques for TCP over ATM.

I. Introduction

While ATM was originally conceived as a carrier of integrated traffic, the recent momentum on the rapid standardization of the technology has come from data networking applications. Since most data applications cannot predict their own bandwidth requirements, they usually require a service that allows all competing active virtual connections (VCs) to dynamically share the available bandwidth. Unspecified bit rate (UBR) service is the simplest service in ATM standardized for the support of data applications. Since transmission control protocol (TCP) is perhaps the most widely used transport layer protocol in existing data networks, the performance of TCP over UBR service in ATM is of major interest to ATM equipment vendors and service providers.

UBR service is designed for those data applications that want to use any available bandwidth and are not sensitive to cell loss or delay. Such connections are not rejected on the basis of bandwidth shortage and not policed for their usage behavior. During congestion, the cells may be lost but the sources are not expected to reduce their cell rates. Instead, these applications may have their own higher-level loss recovery and retransmission mechanisms, such as the window flow control employed by TCP.

Although it is relatively simple to support UBR service in ATM switches, recent results suggest that UBR without any ATM layer congestion control mechanism does not yield adequate performance [14]. The reason for such performance degradation is that when the loss mechanism simply discards arriving cells when buffer overflow occurs, each discarded cell is likely to belong to a different packet. A significant portion of the available bandwidth is then wasted since many packets transmitted are already corrupted by cell loss and thus need to be retransmitted.

To address this problem, more sophisticated frame discarding mechanisms such as the *Early Packet Discard* (EPD) algorithm [14] for UBR service have been proposed. The basic idea of EPD is to discard an entire packet prior to buffer overflow, so that the bandwidth is only utilized for the transmission of non-corrupted packets. EPD in fact represents a general class of packet discarding algorithms, which can be applied to any packet-based protocol running over ATM, such as TCP, UDP, and IPX etc.

One drawback of the EPD scheme proposed in [14] is that it cannot achieve fair bandwidth allocation among competing VCs [11]. For example, EPD tends to allocate less bandwidth for VCs with longer round-trip time and for VCs traversing multiple congested nodes. This is often referred to as the "VC starvation" problem. In addition, connections with bulky data tend to get more bandwidth than highly bursty traffic connections under EPD.

To improve the fairness performance, EPD can be extended by incorporating a fair buffer allocation mechanism to alleviate the fairness problems. In [12], two fair buffer allocation techniques have been studied: per-VC accounting and per-VC queuing. Per-VC accounting uses a single shared FIFO queue and state information on a per-VC basis. The per-VC state information can be, for example, the number of cells of each VC in the FIFO queue. On the other hand, per-VC queuing implements a separate queue for each VC and permits output cell scheduling on a per-VC basis such as round-robin cell scheduling. Simulation results in [12] show significant improvement of fairness performance

Please send all correspondences to K.Y. Siu at Rm. 3-449C, 77 Massachusetts Ave., MIT, Cambridge, MA 02139. Ph. & Fax: (617) 253-5589. Email: siu@list.mit.edu. This work was supported in part by the NSF Young Investigator Awards Program (Grant # MIP-9696144) and by the Department of Energy (Grant #DE-FG03-95ER25273.)

Wenge Ren is currently with Network System in Lucent Technologies, Inc.

0-8186-8061-X/97 $10.00

using these fair buffer allocation techniques. The simulation results also show that per-VC queuing offers far better fairness performance than the specific per-VC accounting technique examined in [12]. However, per-VC queue is much more expensive to implement than a shared FIFO queue.

A technique called virtual queuing is proposed in [2] to regulate the traffic between ABR and VBR interactions. This technique is applied in [16] to emulate the round robin service of per-VC queuing on a FIFO queue for improving the performance of TCP over ATM. This application of the virtual queuing technique can be viewed as a sophisticated form of per-VC accounting. Simulation results in [16] illustrate that when combined with EPD techniques, virtual queuing can provide significant improvement of fairness performance.

The original virtual queuing technique utilizes the queue occupancy on a per-VC basis in a shared FIFO queue. While this choice of per-VC state variable is quite intuitive and yields reasonably good performance, no analysis is given in [16] to provide a theoretical understanding of the technique. Our results in this paper differ from those in [16] in several ways. First, we propose a new choice of per-VC state variable, which has a precise analytical relation to the number of cells transmitted for each VC. Second, we propose a new efficient EPD technique based on dynamic discarding queue thresholds, which can better adapt to highly bursty traffic. The theoretical insights allow us to design an improved scheme that achieves nearly perfect fairness of average throughput among multiple TCP connections.

II. Further Discussion on Related Works

A. Packet Discard Algorithms

As mentioned before, the EPD technique was first proposed in [14] to improve the throughput of TCP in congested ATM networks. Prior to the study in [14], a simpler strategy called *Partial Packet Discard* (PPD) or *selective cell discarding* was studied in [1]. In PPD, an ATM switch will drop all subsequent cells from a packet if one cell of that packet has been dropped due to buffer overflow. As shown in [14], the performance of TCP over ATM using PPD is lower than EPD. This is because in PPD, cells are forced to be discarded when buffer overflows even if the cells belong to packets which already had all of their previous cells transmitted. The bandwidth used to transmit the previous cells of the packets are wasted since the packets are corrupted.

Another packet discard algorithm called *Random Early Detection* (RED) is proposed in [6]. A randomized algorithm, RED uses a mechanism that discards packets with probability proportional to the *average* queue occupancy. RED is designed to maintain a low average queue size in cooperation with end-to-end congestion control mechanisms. It is suggested in [14] that EPD and RED can be used together to improve the performance.

B. TCP over ATM

Following the work in [14], another simulation study of TCP performance over ATM is presented in [4] with various congestion control schemes, including selective cell-drop strategies, and a credit-based flow control scheme that back-pressures individual VCs. Similar to the work in [14], they have shown that comparing with selective cell-drop schemes, TCP performs poorly under plain UBR service. The performance degradation of TCP over UBR under cell loss is similarly observed in [9], where the buffer requirement for TCP over UBR is also studied.

In [11], simulation results are given to compare the performance of TCP over ABR service with an explicit rate congestion control algorithm and TCP over UBR service with EPD (UBR+EPD). In terms of hardware complexity, existing explicit rate congestion control schemes for ABR are more expensive to implement in ATM switches and end systems than UBR+EPD schemes. The purpose of the work in [11] is to see how much performance improvement ABR has over UBR with its added complexity. Simulation results in [11] show the following: (1) UBR+EPD has low fairness performance and requires a relatively large buffer to achieve throughput comparable to ABR even with a small number of active VCs in a LAN configuration. (2) For the same network configurations and with ABR, TCP achieves good performance in terms of fairness and link utilization, and requires a relatively small buffer.

The work in [11] is extended in [12] which studies the performance of TCP over UBR service when EPD schemes are combined with some fair queuing techniques. In particular, simulation results show that in a LAN environment, per-VC based EPD schemes can significantly improve the performance of TCP over UBR service in terms of fairness. Similar results are also reported in [5].

C. Fair Queuing Techniques

The design of fair queuing techniques for conventional networks has been a subject of intensive research in recent years [3], [7], [8], [10], [13], [15], [18]. These works focus on the fair allocation of network resources for different packet streams. The difficulty arises from the fact that the packets are of different sizes. Even if each packet stream is buffered at a separate queue (per-connection queuing), a round robin scheduling service

on the packet streams will still result in unfair bandwidth allocation.

In ATM networks, since cells are of fixed size, given that per-VC queuing is used, round robin scheduling can already provide nearly exact fair bandwidth allocation. The fairness problems addressed in this paper are due to the fact that all cell streams are buffered at a single FIFO queue, where the order of cell arrival completely determines the order of cell departure from the queue.

A survey on various queuing and scheduling mechanisms can be found in [17].

III. Improved Virtual Queuing Technique

In this section, we first discuss a technique called virtual queuing [16], which emulates on a single shared FIFO queue the round-robin buffer allocation provided by per-VC queuing. In particular, a separate "virtual" queue is implemented for each VC by maintaining a state variable $\tilde{Q}_j$ for each VC_j. The variable $\tilde{Q}_j$ is increased by one whenever a cell of VC_j is admitted to the FIFO queue. When the cells are transmitted out of the FIFO queue, however, the variables $\tilde{Q}_j$'s are decremented in a round-robin fashion, regardless of which VC the cell transmitted actually belongs to. Thus, $\tilde{Q}_j$ does not correspond to the actual queue occupancy of VC_j. Instead, it represents the queue occupancy of VC_j as if per-VC queuing and round-robin output scheduling were implemented. The EPD mechanism is then applied to the incoming packets as with per-VC accounting [11], [12], except that the virtual queue $\tilde{Q}_j$ is used instead of the actual queue occupancy Q_j for each VC.

The following is a pseudocode of the EPD mechanism combined with the virtual queuing technique. We denote "Q" and "Qmax" as the current queue occupancy and the maximum capacity of the ATM switch buffer, respectively. "Th" is the EPD threshold. $\tilde{Q}_j$ is the virtual queue size of VC_j. L is a list containing identifiers of the active VCs (i.e., connections with at least one cell in the buffer).

```
When a cell in VC_i comes to an ATM switch:
    if the cell is the first cell of a packet
        Th := Th/N
        if Q ≥ Th and Q̃_i ≥ Th
            discard this cell
        else
            accept this cell into the FIFO queue
            Q̃_i := Q̃_i + 1;
            if Q̃_i = 1
                append i to the tail of L
    else
        if any cell of the packet has been discarded
            discard this cell
        else
            if Q ≥ Q_max
                discard this cell
            else
                accept this cell into the FIFO queue
                Q̃_i := Q̃_i + 1
                if Q̃_i = 1
                    append i to the tail of L

When a cell of the FIFO queue is transmitted
    if L is not empty
        let i denote the first connection identifier in L
        remove i from L
        Q̃_i := Q̃_i − 1
        if Q̃_i > 0
            append i to the tail of L                    □
```

Note that the round robin buffer allocation mechanism for virtual queuing has the limitation that buffer allocation does not apply to connections with empty virtual queue. This is because $\tilde{Q}_j$ is considered to be the *physical* queue size of VC_j as if per-VC queuing were used, and physical queue size cannot have a negative value. This limitation leads to the loss of buffer allocation to active connections with temporarily empty virtual queue.

This limitation, however, can be overcome simply by *allowing $\tilde{Q}_j$ to be negative.* With this change, round-robin buffer allocation is achieved at all time, which means fair bandwidth utilization is guaranteed. Since physical queue size cannot have a negative value, it would be confusing to continue to interpret $\tilde{Q}_j$ as a queue size. Instead, it may be generalized and viewed as the *excess bandwidth usage* for VC_j (see Theorem 2). Our improved virtual queuing technique is based on this idea.

Conceptually, virtual queuing uses the virtual queue occupancy $\tilde{Q}_j$ as the state variable to indicate the number of cells transmitted for VC_j. While this choice of state variable is quite intuitive, the correspondence between $\tilde{Q}_j$ and the number of cells transmitted by VC_j is only approximate. Our improved virtual queuing scheme uses a new state variable M_j that, as we shall show, corresponds *precisely* to the number of cells transmitted by VC_j, which means exact fairness of bandwidth utilization can be achieved if M_j is made fair among the VCs.

The implementation of our improved scheme is very similar to that of virtual queuing, with $\tilde{Q}_j$ replaced by M_j. The difference between M_j and $\tilde{Q}_j$ is that M_j is allowed to be negative while $\tilde{Q}_j$ is not. However, a lower bound $-w$ (a negative value) needs to be imposed on M_j. This is because packets of VC_j are accepted into the buffer as long as M_j is less than the discard threshold ($\overline{Th}$). If M_j is unbounded and becomes very negative, a large burst of incoming packets of VC_j will be accepted into the shared FIFO queue, which could result in buffer overflow.

The pseudocode for our improved virtual queuing scheme is as follows:

```
Assume initially L contains connection ID (i) for all active VCs.
When a cell in VC_i reaches an ATM switch:
    if the cell is the first cell of a packet
        Th' := Th/N
        if Q ≥ Th and M_i ≥ Th'
            discard this cell
        else
            accept this cell into the FIFO queue
            M_i := M_i + 1;
            if M_i = -w + 1
                append i to the tail of L
    else
        if any cell of the packet has been discarded
            discard this cell
        else
            if Q ≥ Q_max
                discard this cell
            else
                accept this cell into the FIFO queue
                M_i := M_i + 1
                if M_i = -w + 1
                    append i to the tail of L

When a cell of the FIFO queue is transmitted
    if L is not empty
        let i denote the first connection identifier in L
        remove i from L
        M_i := M_i - 1
        if M_i > -w
            append i to the tail of L                    □
```

The discard threshold $\overline{Th}$ used in our pseudocode above is fixed for a given number of VCs. This simple choice of discard threshold does not adapt to the changing traffic conditions. Here we propose a dynamic threshold discarding technique, which can improve the fairness performance. Since the ideas of this technique are based on the analytical results to be presented in the next section, we shall postpone its discussion until Section IV-B.

Compared with our improved scheme with the original virtual queuing technique, we see that the increase in implementation complexity is negligible. However, the performance improves dramatically. In fact, our scheme achieves exact fairness in the long run, as we will show analytically in the next section.

IV. Analytical Results

Our improved virtual queuing, like other fair queuing techniques, has two functions: provide fairness of bandwidth utilization, and prevent buffer overflow. First, we will find the relationship between M_j and the total queue occupancy, from which we will see that our scheme maintains the queue size around the EPD threshold. Then, we shall describe a new EPD technique that adapts the per-VC threshold dynamically to the changing traffic.

A. Relationship between Actual and Virtual Queues

Theorem 1: Let Q be the total queue occupancy and M_j be the virtual queue variable as defined previously. Then $Q = \sum_{j=1}^{n} M_j$. In other words, the sum of all M_j's equals to the total queue occupancy.

Proof: Initially, $Q = \sum_{j=1}^{n} M_j = 0$. From the pseudocode, we see that $\sum_{j=1}^{n} M_j$ increases by one when a cell is accepted into the FIFO queue, i.e., when the total queue occupancy increases by one. Similarly, $\sum_{j=1}^{n} M_j$ decreases when a cell is transmitted out of the FIFO queue, i.e., when the total queue occupancy decreases by one. □

From Theorem 1, we see that if $\sum_{j=1}^{n} M_j$ is bounded, then the total queue occupancy will be bounded and no buffer overflow will occur. In our scheme, the discard threshold for each VC is Th/N. Thus, on average, $M_j \approx Th/N$ and $\sum_{j=1}^{n} M_j \approx Th$. Therefore, with our implementation, the total queue occupancy will be maintained around the EPD threshold Th.

Theorem 2: Assume that every session has a weight of one, and that each M_j never reaches $-w$. Then $M_j = Cell_j - K$, where $Cell_j$ is the number of cells of VC_j remaining in buffer or transmitted out of the FIFO queue, and K is the number of rounds of service.

Proof: Before service starts, $M_j = Cell_j = K = 0$. M_j is incremented by one if and only if $Cell_j$ is incremented by one when a new cell of VC_j is accepted. M_j's are decremented in a round robin fashion, thus, M_j is decremented by one in each round of service, and this is the only way to decrement M_j. In K rounds, M_j is decremented by K. Hence, $M_j = Cell_j - K$. □

Theorem 2 indicates that if we keep M_j fair among VCs, then the throughput $Cell_j$ is also fair. This is because if $M_i = M_j$, then $Cell_i = Cell_j$. Therefore, by using packet discarding techniques to balance M_j's, one can maintain fair bandwidth utilization among all VCs.

Our improved virtual queuing ensures equal *average* bandwidth utilization among VCs. However, the *instantaneous* transmission rate of each VC may be a little different. This is mainly because our scheme still uses a FIFO queuing discipline. For data transfer or other non real-time applications, however, the average bandwidth utilization is more important and the small variation of instantaneous bandwidth can be safely neglected. For highly bursty traffic (where the per-VC queue or virtual queue may be empty frequently), other schemes, including the expensive per-VC queuing, cannot ensure fair average bandwidth utilization since no buffer allocation is provided to VCs with empty queue. Our scheme, however, still ensures fair bandwidth al-

location since M_j is allowed to be negative, and can continue to be decremented (until the lower bound is reached) even when the actual queue is empty.

B. Dynamic Discard Threshold

Among the established UBR connections, some may be idle, and some may be active but use less bandwidth than being allocated. In both cases, the M_j values for these sessions will be less than their "fair shares." From Theorem 1, M_j's for the other sessions will exceed the fair share since the sum of all M_j's equals to the total queue occupancy which is maintained near a constant value with the EPD mechanism. The discard threshold ($\hat{Th}$) should be set to the desired value for sessions with *more* data to send than their fair shares. This is because only these sessions are affected by the discard threshold. The equation for the discard threshold can be obtained as follows. As usual, let

$$\overline{Th} = \frac{EPD\ Threshold}{N}$$

where N is the number of established connections (idle or active). $\overline{Th}$ can be considered as the "fair share" value of M_j. From Theorem 1,

$$Q = \sum_{all\ j} M_j \ .$$

Grouping the sessions according to the value of M_j, we get

$$\sum_{M_j < \overline{Th}} M_j + \sum_{M_j > \overline{Th}} M_j = Q \ ,$$

where sessions with $M_j < \overline{Th}$ are sessions which are idle or using less bandwidth than requested, sessions with $M_j > \overline{Th}$ are active sessions using more bandwidth than requested. Rearrange the above equation, yields

$$\sum_{M_j > \overline{Th}} M_j = Q - \sum_{M_j < \overline{Th}} M_j \ . \qquad (1)$$

Equation (1) gives the sum of M_j's for all sessions with more data to send than the fair share. It is desirable for these sessions to have equal value of M_j so that the throughput would be fair among them according to Theorem 2. Let $N_{M_j > \overline{Th}}$ denote the number of such sessions. To divide the sum of M_j's equally among them, the discard threshold must be

$$\hat{Th} = \frac{\sum_{M_j > \overline{Th}} M_j}{N_{M_j > \overline{Th}}} \ . \qquad (2)$$

Substituting (1) into (2) and using the fact that the total queue occupancy is maintained near the EPD threshold, we obtain

$$\hat{Th} = \frac{EPD\ Threshold\ -\ \sum_{M_j < \overline{Th}} M_j}{N_{M_j > \overline{Th}}} \ . \qquad (3)$$

Equation (3) gives the dynamic discard threshold. The frequency at which $\hat{Th}$ is updated involves a trade off between the amount of processing required and the accuracy. Updating once after every round of service is a reasonable compromise. In Section VI, we shall show by simulation that by using a dynamic discard threshold, the fairness performance can be improved quite significantly in some cases.

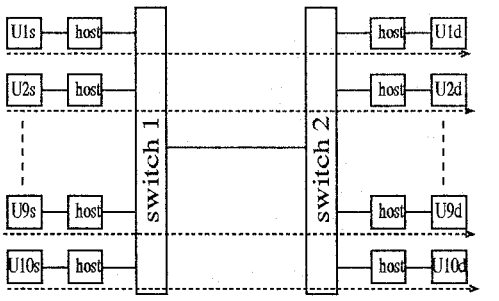

Fig. 1. A 10-VC peer-to-peer configuration

V. Simulation Model

This section presents simulation results comparing the performance of the various queuing mechanisms. Our simulation tool is based on the MIT Network Simulator (NetSim). NetSim is an event-driven simulator composed of various components that send messages to one another. We have built upon the NetSim package by adding various ATM and TCP related components. These components include an ATM switch component, a SONET OC-3c link component, an ATM host component, a greedy source component, a multiplexing and demultiplexing component, and a TCP Reno component.

The ATM host component performs ATM Adaptation Layer for data services (AAL5) including segmentation and reassembly (SAR) of TCP packets. The ATM switch component used for this paper models a UBR switch combined with early packet discarding capability. It is a 64 × 64 output buffered switch, whose internal speedup is 64, i.e., there is no internal blocking caused by switching fabric. Each output port has a single FIFO queue, which is referred to as an output buffer.

Figure 1 illustrates the simulation model of a network with ten peer-to-peer connections based on our simulation components. On the sending side, the user components (U1s – U10s) generate data for TCP components,

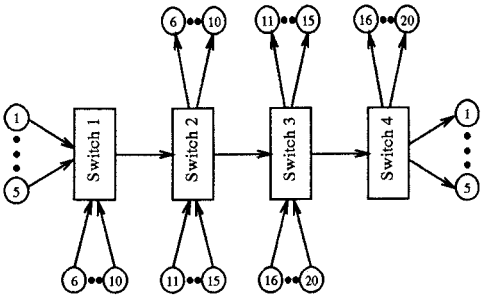

Fig. 2. A chain configuration.

which form TCP packets, and which in turn are passed onto host adapters for AAL5 processing. The two ATM switches perform cell switching between their input and output ports. On the receiving side, cells are reassembled and passed to the TCP components and then user components (U1d – U10d). By running ten concurrent TCP connections, we create a congested link between the two ATM switches. This set of simulation employs *persistent* traffic sources (so that there is always data to transmit when bandwidth is available). Bursty sources, which generate data intermittently, will be used later.

In addition to the peer-to-peer configuration, we shall also simulate a more complicated "chain configuration" with 20 persistent data connections (Figure 2). For simplicity, instead of illustrating the details of the components, we group user, TCP, and ATM host components into a single node in this Figure. Note that VC 1-5 traverse three congested links but each of the other VCs (6-20) traverses only one congested link. Moreover, each link is traversed by 10 VCs and is assumed to have the same capacity (150Mbps).

A third set of simulation is run using the chain configuration with some of the persistent data sources replaced by bursty sources. The bursty sources send out a burst of 520KB of data to TCP every 400ms. In each of the group of five sources (Figure 2), three are persistent data sources, and two are bursty sources. For example, sources 6-8 are persistent, and sources 9-10 are bursty.

The following parameters are employed in our simulations:

TCP:
- Mean Packet Processing Delay = 300 μsec,
- Packet Processing Delay Variation = 10 μsec,
- Packet Size = 2KB,
- Maximum Receiver Window Size = 64KB,
- Default Timeout = 500 $m$sec.

Link:
- Speed = 150 Mbps,
- Delay = 500 $n$sec (Distance = 100m).

UBR-End System:
- Packet Processing Delay = 500 $n$sec,
- Buffer Size = infinity,
- Cell Transmission Rate = 150Mbps.

UBR-Switch:
- Packet Processing Delay = 4 μsec,
- Buffer Size (Q_{max}) = 3000 cells,
- EPD Threshold (Th) = 1000 cells

The TCP packet processing delay is the time that it takes the TCP source to handle the transmission of a data packet or the receipt of an acknowledgement packet. It also represents the time that it takes the TCP destination to handle the receipt of a data packet or the transmission of an acknowledgement packet. Since the TCP processing time may vary from packet to packet, we simulate it by using the mean packet processing delay plus or minus a random time, which is in the range of packet processing delay variation.

Note that the TCP default timeout will be used only when the first packet of a connection is lost, since no round trip time can be applied to calculate the retransmission timeout. However, in all our simulations, since no first packet is lost due to sufficiently large buffer, the default timeout is never triggered.

In our simulations, we consider a link delay of 500 $n$sec (about the distance of 100 m) to model a LAN environment. We expect TCP performance to be similar as long as the link distance lies within LAN range, e.g., up to 1000 m. This is because even with a 1000 m link distance, the link delay is only 5 μsec, which is negligible compared with the TCP packet processing delay of 300 μsec in our simulation.

VI. Simulation Results

The 10-VC peer-to-peer configuration (Figure 1) represents a symmetric configuration with each session traversing the same number of switches through the network (queuing delay is roughly fair). We observe from Figure 3 that both schemes achieve nearly perfect fairness for this simple configuration, with our improved virtual queuing scheme yielding slightly better results in this case.

The fairness problems for the various mechanisms are much more pronounced in the chain configuration (Figure 2). To achieve max-min fair bandwidth allocation in the chain configuration, the throughput of each VC should be equal. However, simulation results in [12] have shown that with FIFO queue and simple EPD, the throughputs of VC1 to VC5 are significantly less than those of the other VCs. Virtual queuing improves the fairness by incorporating round-robin buffer allocation. Our improved virtual queuing technique achieves the best fairness in the long run, even though the switches use FIFO queuing. This is the result of the complete round-robin buffer allocation. To see how our scheme improves the fairness intuitively, refer to Figure 4. During slow start, the bandwidth for connections traversing more nodes (slope B) is lower than the bandwidth

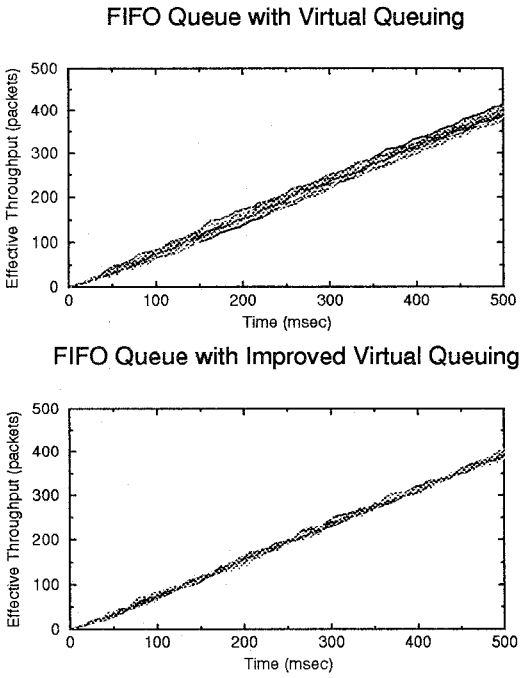

Fig. 3. TCP effective throughputs in the peer-to-peer configuration.

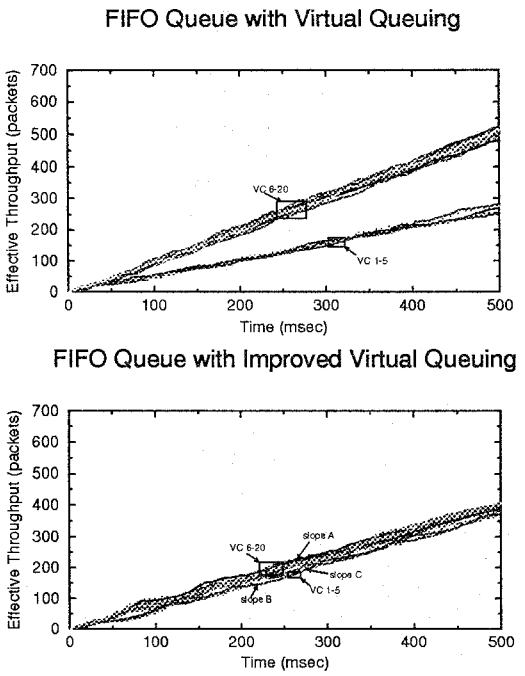

Fig. 4. TCP effective throughputs in the chain configuration.

for connections traversing fewer nodes (slope A). However, since buffer allocation is fair, connections which use less buffer space during slow start will have a lower M_j value, and will be able to regain the bandwidth lost by staying longer in full bandwidth (slope C).

Figure 5 shows the simulation results when both persistent and bursty data sources are used. Our improved scheme provides fair bandwidth utilization in the long run because it is able to allocate buffer to the bursty sources when their queue occupancy is zero during the idle periods.

In many data applications, sources may be idle from time to time. As discussed in Section IV-B, idle connections may cause fairness problems if the fixed threshold discarding scheme is employed. We described in Section IV-B a dynamic threshold discarding technique to alleviate this potential problem. Simulation results in Figure 6 illustrate the effectiveness of our technique using the chain configuration (Figure 2). In this simulation, VC1, VC6, VC11, and VC16 have limited data (200 Kbytes each) to send and become idle after sending all data around 100 *m*sec. All other VCs are persistent. Simulation results clearly show that VC2 to VC5 are treated unfairly when a fixed per-VC threshold is used, whereas nearly perfect fairness can be achieved for all VCs when our dynamic threshold discarding technique is used.

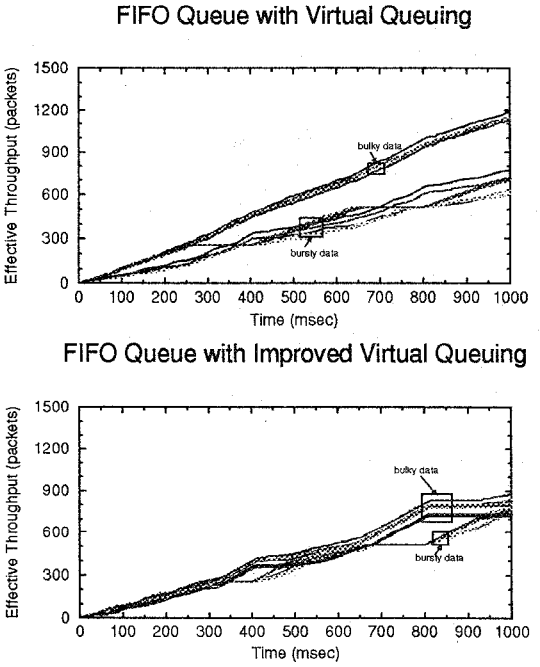

Fig. 5. TCP effective throughputs in the chain configuration with persistent and bursty data sources.

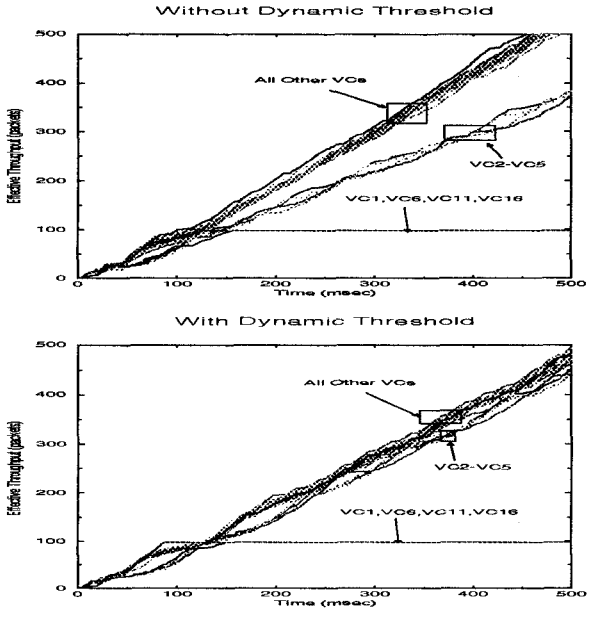

Fig. 6. TCP effective throughputs with/without dynamic threshold in the chain configuration.

VII. Concluding Remarks

We have shown analytically and through simulation that our improved virtual queuing technique combined with dynamic EPD scheme can significantly improve the performance of TCP over UBR service. While per-VC queuing is expensive to implement, our scheme requires low implementation cost and provides performance of TCP over UBR comparable to the performance of the expensive per-VC queuing technique.

References

[1] G. J. Armitage and K. M. Adams. Packet reassembly during cell loss. *IEEE Network*, Sept. 1993, vol.7, (no.5):26-34.

[2] F. Chiussi, Y. Xia, and V. P. Kumar. Virtual Queuing Techniques for ABR Service: Improving ABR/VBR Interaction. Proceedings of Infocom'97.

[3] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queuing algorithm. *Internetworking: Research and Experience*, Sept. 1990, vol.1, (no.1):3-26.

[4] C. Fang, H. Chen, and J. Hutchins. A simulation study of TCP performance in ATM networks. *Proceedings of IEEE GLOBECOM*, pp. 1217-1223, San Francisco, Nov 28 - Dec 2, 1994.

[5] C. Fang and A. Lin. On TCP Performance of UBR with EPD and UBR-EPD with a Fair Buffer Allocation Scheme. *ATM Forum Contribution 95-1645*, December 1995.

[6] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, Aug. 1993, vol.1, (no.4):397-413.

[7] S. J. Golestani. Network delay analysis of a class of fair queuing algorithms. *IEEE Journal on Selected Areas in Communications*, Aug. 1995, vol.13, (no.6):1057-70.

[8] A. G. Greenberg and N. Madras. How fair is fair queuing?. *Journal of the Association for Computing Machinery*, July 1992, vol.39, (no.3):568-98.

[9] R. Jain, R. Goyal, S. Kalyanaraman, and S. Fahmy. Performance of TCP over UBR and buffer requirements. *ATM Forum Contribution 96-0518*, April 1996.

[10] S. Keshav. On the efficient implementation of fair queuing. *Internetworking: Research and Experience*, Sept. 1991, vol.2, (no.3):157-73.

[11] H. Li, K.-Y. Siu, H.-Y. Tzeng, C. Ikeda and H. Suzuki. A simulation study of TCP performance over ABR and UBR services in ATM LANs. *IEICE Transactions on Communications*, Special Issue on High Speed Local Area Network, May 1996, vol.E79-B, (no.5):658-67.

[12] H. Li, K.-Y. Siu, H.-Y. Tzeng, C. Ikeda and H. Suzuki. On TCP Performance in ATM Networks with Per-VC Early Packet Discard Mechanisms. *Computer Communications*, Special Issue on Enabling ATM Networks, Nov. 1996, vol.19, (no.13):1065-76.

[13] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Transactions on Networking*, June 1993, vol.1, (no.3):344-57.

[14] A. Romanow and S. Floyd. Dynamics of TCP Traffic over ATM Networks. *IEEE Journal on Selected Areas in Communications*. pp. 633–41, vol. 13, No. 4, May 1995.

[15] M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round robin. *Computer Communication Review*, Oct. 1995, vol.25, (no.4):231-42.

[16] H.-Y. Tzeng and K.-Y. Siu. Performance of TCP over UBR IN ATM with EPD and Virtual Queuing Techniques. Proceedings of Workshop on Transport Layer Protocols over High Speed Networks, IEEE Globecom, Nov. 1996.

[17] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, Oct. 1995, vol.83, (no.10):1374-96.

[18] L. Zhang. Virtual Clock: A New Traffic Control Algorithm for Packet Switched Networks. *ACM Transactions on Computer Systems*, 9(2):101-125, May 1991.