NASA/CR—1999-209158

Design Issues for Traffic Management for the ATM UBR+ Service for TCP Over Satellite Networks

Raj Jain
Ohio State University, Columbus, Ohio

Prepared under Contract NAS3–97198

National Aeronautics and
Space Administration

Glenn Research Center

July 1999

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076
Price Code: A09

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100
Price Code: A09

Overview of the Report

This project was a comprehensive research program for developing techniques for improving the performance of internet protocols over Asynchronous Transfer Mode (ATM) based satellite networks. Among the service categories provided by ATM networks, the most commonly used category for data traffic is the unspecified bit rate (UBR) service. UBR allows sources to send data into the network without any feedback control.

Several issues arise in optimizing the performance of Transmission Control Protocol (TCP) when ATM-UBR service is used over satellite links. In this project, we studied several TCP mechanisms as well as ATM-UBR mechanisms to improve TCP performance over long-delay ATM networks. The UBR mechanisms that we studied in this project are:

UBR with frame level discard policies,

UBR with intelligent buffer management,

UBR with guaranteed rate,

Guaranteed Frame Rate (GFR),

The following TCP mechanisms were studied:

Vanilla TCP with slow start and congestion avoidance,

TCP Reno with fast retransmit and recovery,

TCP New Reno

TCP with selective acknowledgements (SACK)

We studied almost all possible combinations of these mechanisms using an extensive set of simulations and quantified the effect of each of these mechanisms.

We found that vanilla TCP over the UBR service category achieves low throughput and low fairness over satellite networks. This is because during packet loss, TCP loses significant amount of time waiting for retransmission timeout.

In the presence of bursty packet losses, fast retransmit and recovery (FRR) (without SACK) further hurts TCP performance over UBR for long delay-bandwidth product networks. This is because after two fast retransmissions, the congestion window is too small to send out new packets that trigger duplicate acks. In the absence of duplicate acks, the third lost packet is not retransmitted, and a timeout occurs at a small window. This results in congestion avoidance with a small window, which is very slow for long delay networks.

Frame level discard policies such as early packet discard (EPD) improve the throughput significantly over cell-level discard policies. However, the fairness is not guaranteed unless intellient buffer management with per virtual circuit (VC) accounting is used.

Throughput increases further with more aggressive New Reno and SACK. SACK gives the best performance in terms of throughput. We found that for long delay paths, the throughput improvement due to SACK is more than that from discard policies and buffer management. Also, a buffer size equal to half the round-trip delay was found to be sufficient.

Another method for improving the UBR performance is the so called "guaranteed rate" (GR) in which a small fraction of the bandwidth is reserved in the switches for the UBR service category. This bandwidth is shared by all UBR VCs. Using guaranteed rates helps in the presence of a high load of higher priority traffic such as Constant Bit Rate (CBR) or Variable Bit Rate (VBR) traffic. We found that reserving just a small fraction, say 10%, of the bandwidth for UBR significantly improves TCP performance. This is because the reserved bandwidth ensures that the flow of TCP packets and acknowledgements is continuous and prevents TCP timeouts due to temporary bandwidth starvation of UBR. Note that this mechanism is different from the GFR service category where each VC (rather than the entire UBR class) has a minimum rate guarantee. The results described above hold for both persistent TCP as well as world-wide web TCP traffic.

We also studied GFR service category, which is an enhancement of UBR and provides per-VC minimum cell rate (MCR) guarantee. We showed that per-VC queueing and scheduling are sufficient to provide the guarantees. If a switch does not have per-VC queueing, as might be the case in an on-board switch, special buffer allocation is required to achieve efficient and fair allocation of resources. We designed a "Differential Fair Buffer Allocation (DFBA)" scheme that allows MCR guarantees with a single queue using only per-VC accounting.

The project resulted in the numerous ATM Forum contributions and papers. These are listed below against each of the seven deliverables of the project.

I. Switch and end-system policies for satellite networks.

I.A **"Traffic Management for TCP/IP over Satellite-ATM Networks,"** Rohit Goyal, Raj Jain, Sastri Kota, Mukul Goyal, Sonia Fahmy, Bobby Vandalore, To appear in IEEE CommunicationsMagazine, March 1999, 18 pp., http://www.cis.ohio-state.edu/~jain/papers/comm399.htm

I.B **"Improving the performance of TCP/IP over Satellite-ATM Networks,"** Rohit Goyal, Raj Jain, Sastri Kota, Mukul Goyal, Sonia Fahmy, Bobby Vandalore, Under preparation. To be submitted to International Journal of Satellite Communications, Special Issue on Internet Protocols over Satellite.

II. UBR switch drop policies and the minimum rate guarentee interaction with TCP congestion control algorithms.

II.A **"TCP Selective Acknowledgments and UBR Drop Policies to Improve ATM-UBR Performance over Terrestrial and Satellite Networks",** Reference : Rohit Goyal, Raj Jain, Shivkumar Kalyanaraman, Sonia Fahmy, Bobby Vandalore, Sastri Kota, Proc. ICCCN '97, Las Vegas, September 1997, pp. 17-27, http://www.cis.ohio-state.edu/~jain/papers/ic3n97.htm

II.B **"Buffer Management for the GFR Service,"** Rohit Goyal, Raj Jain, Sonia Fahmy, Bobby Vandalore, ATM_Forum/98-0405, July 1998, http://www.cis.ohio-state.edu/~jain/atmf/a98-0405.htm

II.C **"Buffer Management for the GFR Service,"** Rohit Goyal, Raj Jain, Sonia Fahmy, Bobby Vandalore, Submitted to Journal of Computer Communications, January 1999, 33 pp., http://www.cis.ohio-state.edu/~jain/papers/dfba.htm

II.D **"GFR Implementation Options,"** Rohit Goyal, Raj Jain, Sonia Fahmy, Bobby Vandalore ATM_Forum/98-0406, July 1998, http://www.cis.ohio-state.edu/~jain/atmf/a98-0406.htm

III. **Buffer requirements as a function of delay-bandwidth products.**

III.A **"Analysis and Simulation of Delay and Buffer Requirements of Satellite-ATM Networks for TCP/IP Traffic,"** Rohit Goyal, Sastri Kota, Raj Jain, Sonia Fahmy, Bobby Vandalore, Jerry Kallaus Under preparation, http://www.cis.ohio-state.edu/~jain/papers/jsac98.htm

III.B **"UBR Buffer Requirements for TCP/IP over Satellite Networks,"** Reference: Rohit Goyal, Raj Jain, Sonia Fahmy, Bobby Vandalore, Shiv Kalyanaraman, Sastri Kota, Pradeep Samudra, ATM Forum/97-0616, July 1997, http://www.cis.ohio-state.edu/~jain/atmf/a97-0616.htm

IV. **UBR bandwidth starvation by higher priority VBR traffic.**

IV.A **"Guaranteed Rate for Improving TCP Performance on UBR+ over Terrestrial and Satellite Networks,"** Rohit Goyal, Raj Jain, Shiv Kalyanaraman, Sonia Fahmy, Bobby Vandalore, Xiangrong Cai, Seong-Cheol Kim, Sastri Kota, ATM Forum/97-0424, April 1997, http://www.cis.ohio-state.edu/~jain/atmf/a97-0424.htm

V. **Bursty Sources**

V.A **"Performance Analysis of TCP Enhancements for WWW Traffic using UBR+ with Limited Buffers over Satellite Links"** Mukul Goyal, Rohit Goyal, Raj Jain, Bobby Vandalore, Sonia Fahmy, Tom VonDeak, Kul Bhasin, Norm Butts, and Sastri Kota, , ATM_Forum/98-0876R1, December 1998, http://www.cis.ohio-state.edu/~jain/atmf/a98-0876.htm

VI. **Large congestion window and the congestion avoidance phase.**

VI.A See ICCCN'97 paper under deliverable 2 above.

VII. **Optimizing the performance of SACK TCP**

VII.A We analyzed the performance of SACK TCP using delayed retransmit. It was found to not have any significant effect on the performance. No papers were published on this topic.

I. Switch and end-system policies for satellite networks

I.A

"Traffic Management for TCP/IP over Satellite-ATM Networks"

I.B

"Improving the Performance of TCP/IP over Satellite-ATM Networks"

Traffic Management for TCP/IP over Satellite-ATM Networks[1]

Rohit Goyal[a], Raj Jain[a], Sastri Kota[b], Mukul Goyal[a], Sonia Fahmy[a], Bobby Vandalore[a]

a). The Ohio State University

b). Lockheed Martin Telecommunications

Abstract

Several Ka-band satellite systems have been proposed that will use ATM technology to seamlessly transport Internet traffic. The ATM UBR, GFR and ABR service categories have been designed for data. However, several studies have reported poor TCP performance over satellite-ATM networks. We describe techniques to improve TCP performance over satellite-ATM networks. We first discuss the various design options available for TCP end-systems, IP-ATM edge devices as well as ATM switches for long latency connections. We discuss buffer management policies, guaranteed rate services, and the virtual source/virtual destination option in ATM. We present a comparison of ATM service categories for TCP transport over satellite links. The main goal of this paper is to discuss design and performance issues for the transport of TCP over UBR, GFR and ABR services for satellite-ATM networks.

---

[1] This work was partially supported by the NASA Glenn Research Center.

1. Introduction

ATM technology is expected to provide quality of service based networks that support voice, video and data applications. ATM was originally designed for fiber based terrestrial networks that exhibit low latencies and low error rates. With an increasing demand for electronic connectivity across the world, satellite networks play an indispensable role in the deployment of global networks. Ka-band satellites using the gigahertz frequency spectrum can reach user terminals across most of the populated world. ATM based satellite networks can effectively provide real time as well as non-real time communications services to remote areas.

However, satellite systems have several inherent constraints. The resources of the satellite communication network, especially the satellite and the earth station are expensive and typically have low redundancy; these must be robust and be used efficiently. The large delays in GEO systems, and delay variations in LEO systems, affect both real time and non-real time applications. In an acknowledgment and timeout based congestion control mechanism (like TCP), performance is inherently related to the delay-bandwidth product of the connection. Moreover, TCP Round Trip Time (RTT) measurements are sensitive to delay variations that may cause false timeouts and retransmissions. As a result, the congestion control issues for broadband satellite networks are somewhat different from those of low latency terrestrial networks. Both interoperability issues, as well as performance issues need to be addressed before a transport layer protocol like TCP can satisfactorily work over long latency satellite-ATM networks.

In this paper, we describe the various design options for improving he performance of TCP/IP over satellite-ATM networks. The next section describes the ATM service categories and options available to TCP/IP traffic. We then describe each ATM design option as well as TCP mechanism, and evaluate their performance over satellite networks. We conclude with a comparison of ATM service categories for TCP transport over satellite links.

2. Design Issues for TCP/IP over Satellite-ATM

Satellite-ATM networks can be used to provide broadband *access* to remote locations, as well as to serve as an alternative to fiber based *backbone* networks. In either case, a single satellite is designed to support thousands of earth terminals. The earth terminals set up VCs through the on-board satellite switches to transfer ATM cells among one another. Because of the limited capacity of a satellite switch, each earth terminal has a limited number of VCs it can use for TCP/IP data transport. In backbone networks, these earth terminals are IP-ATM edge devices that terminate ATM connections, and route IP traffic in and out of the ATM network. These high capacity backbone routers must handle thousands of simultaneous IP flows. As a result, the routers must be able to aggregate multiple IP flows onto individual VCs. Flow classification may be done by means of a *QoS manager* that can use IP source-destination address pairs, as well as transport layer port numbers[2]. The QoS manager can further classify IP packets into flows based on the differentiated services priority levels in the TOS byte of the IP header.

In addition to flow and VC management, the earth terminals must also provide means for congestion control between the IP network and the ATM network. The on-board ATM switches must perform traffic management at the cell and the VC levels. In addition, TCP hosts can implement various TCP flow and congestion control mechanisms for effective network bandwidth utilization. Figure 1 illustrates a framework for the various design options available to networks and TCP hosts for congestion control. The techniques in the figure can be used to implement various ATM services in the network. Enhancements that perform intelligent buffer management policies at the switches can be developed for UBR to improve transport layer throughput and fairness. A policy for selective cell drop based on per-VC accounting can be used to improve fairness.

Providing a minimum Guaranteed Rate (GR) to the UBR traffic has been discussed as a possible candidate to improve TCP performance over UBR. The goal of providing guaranteed rate is to protect the UBR service category from total bandwidth starvation, and provide a continuous

[2] TCP/UDP port numbers are accessible only if end-to-end security protocols are not used.

minimum bandwidth guarantee. It has been shown that in the presence of high load of higher priority Constant Bit Rate (CBR), Variable Bit Rate (VBR) and Available Bit Rate (ABR) traffic, TCP congestion control mechanisms benefit from a guaranteed minimum rate.

Guaranteed Frame Rate (GFR) has been recently proposed in the ATM Forum as an enhancement to the UBR service category. Guaranteed Frame Rate will provide a minimum rate guarantee to VCs at the frame level. The GFR service also allows for the fair usage of any extra network bandwidth. GFR is likely to be used by applications that can neither specify the traffic parameters needed for a VBR VC, nor have capability for ABR (for rate based feedback control). Current internetworking applications fall into this category, and are not designed to run over QoS based networks. Routers separated by satellite-ATM networks can use the GFR service to establish guaranteed rate VCs between one another. GFR and GR can be implemented using per-VC queuing or buffer management.

The Available Bit Rate (ABR) service category is another option to implement TCP/IP over ATM. The *Available Bit Rate (ABR)* service category is specified by a PCR and Minimum Cell Rate (MCR) which is guaranteed by the network. ABR connections use a rate-based closed-loop end-to-end feedback-control mechanism for congestion control. The network tries to maintain a low Cell Loss Ratio by changing the allowed cell rates (ACR) at which a source can send. Switches can also use the virtual source/virtual destination (VS/VD) feature to segment the ABR control loop into smaller loops. Studies have indicated that ABR with VS/VD can effectively reduce the buffer requirement for TCP over ATM especially for long delay paths. ABR can be implemented using the feedback control mechanisms in figure 1.

In addition to network based drop policies, end-to-end flow control and congestion control policies can be effective in improving TCP performance over UBR. The fast retransmit and recovery mechanism, can be used in addition to slow start and congestion avoidance to quickly recover from isolated segment losses. The selective acknowledgments (SACK) option has been proposed to recover quickly from multiple segment losses. A change to TCP's fast retransmit and recovery has been suggested in [HOE96]. The use of performance enhancing TCP gateways to improve performance over satellite links has also been proposed in recent studies. The following

sections discuss the design and performance issues for TCP over UBR, GFR and ABR services for satellite networks.

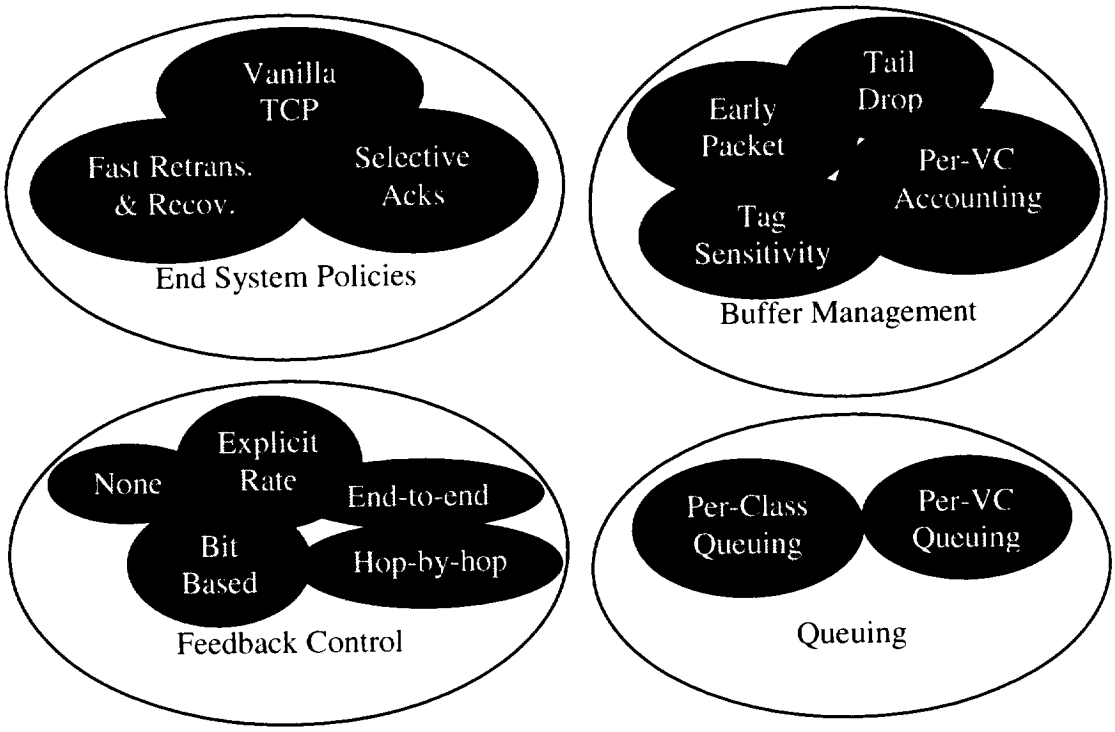


Figure 1 Design Issues for TCP over ATM

3. TCP over UBR

In its simplest form, an ATM switch implements a tail drop policy for the UBR service category. If cells are dropped, the TCP source loses time, waiting for the retransmission timeout. Even though TCP congestion mechanisms effectively recover from loss, the link efficiency can be very low, especially for large delay-bandwidth networks. In general, link efficiency typically increases with increasing buffer size. Performance of TCP over UBR can be improved using buffer management policies. In addition, TCP performance is also effected by TCP congestion control mechanisms and TCP parameters such as segment size, timer granularity, receiver window size, slow start threshold, and initial window size.

TCP Reno implements the fast retransmit and recovery algorithms that enable the connection to quickly recover from isolated segment losses. However fast retransmit and recovery cannot

efficiently recover from multiple packet losses within the same window. A modification to Reno is proposed in [HOE96] so that the sender can recover from multiple packet losses without having to time out.

TCP with Selective Acknowledgments (SACK TCP) is designed to efficiently recover from multiple segment losses. With SACK, the sender can recover from multiple dropped segments in about one round trip. Comparisons of TCP drop policies for persistent traffic over satellite-ATM are presented in [GOYAL97]. The studies show that in low delay networks, the effect of network based buffer management policies is very important and can dominate the effect of SACK. The throughput improvement provided by SACK is very significant for long latency connections. When the propagation delay is large, timeout results in the loss of a significant amount of time during slow start from a window of one segment. Reno TCP (with fast retransmit and recovery), results in worst performance (for multiple packet losses) because timeout occurs at a much lower window than vanilla TCP. With SACK TCP, a timeout is avoided most of the time, and recovery is complete within a small number of roundtrips. For lower delay satellite networks (LEOs), both NewReno and SACK TCPs provide high throughput, but as the latency increases, SACK significantly outperforms NewReno, Reno and Vanilla.

3.1. UBR+: Enhancements to UBR

Recent research has focussed on fair buffer management for best effort network traffic. In these proposals, packets are dropped when the buffer occupancy exceeds a certain threshold. Most buffer management schemes improve the efficiency of TCP over UBR. However, only some of the schemes affect the fairness properties of TCP over UBR. The proposals for buffer management can be classified into four groups based on whether they maintain multiple buffer occupancies (Multiple Accounting -- MA) or a single global buffer occupancy (Single Accounting -- SA), and whether they use multiple discard thresholds (Multiple Thresholds -- MT) or a single global discard threshold (Single Threshold -- ST). Table 1 lists the four classes of buffer management schemes and examples of schemes for these classes. The schemes are briefly discussed below.

The SA schemes maintain a single count of the number of cells currently in the buffer. The MA schemes classify the traffic into several classes and maintain a separate count for the number of cells in the buffer for each class. Typically, each class corresponds to a single connection, and these schemes maintain per-connection occupancies. In cases where the number of connections far exceeds the buffer size, the added over-head of per-connection accounting may be very expensive. In this case, a set of active connections can be defined as those connections with at least one packet in the buffer, and only the buffer occupancies of active connections need to be maintained.

Table 1 Classification of Buffer Management Schemes

Buffer Management Class	Examples	Threshold Type (Static/Dynamic)	Drop Type (Deterministic/ Probabilistic)	Tag Sensitive (Yes/No)	Fairness
SA--ST	EPD, PPD	Static	Deterministic	No	None
	RED	Static	Probabilistic	No	Equal allocation in limited cases
MA--ST	FRED	Dynamic	Probabilistic	No	Equal allocation
	SD, FBA	Dynamic	Deterministic	No	Equal allocation
	VQ+Dynamic EPD	Dynamic	Deterministic	No	Equal Allocation
MA--MT	PME+ERED	Static	Probabilistic	Yes	MCR guarantee
	DFBA	Dynamic	Probabilistic	Yes	MCR guarantee
	VQ+MCR scheduling	Dynamic	Deterministic	No	MCR guarantee

SA--MT	Priority Drop	Static	Deterministic	Yes	--

Single threshold (ST) schemes compare the buffer occupancy(s) with a single threshold and drop packets when the buffer occupancy exceeds the threshold. Multiple thresholds (MT) can be maintained corresponding to classes, connections, or to provide differentiated services. Several modifications to this drop behavior can be implemented, including averaging buffer occupancies, static versus dynamic thresholds, deterministic versus probabilistic discards, and discard levels based on packet tags. Examples of packet tags are the CLP bit in ATM cells or the TOS octet in the IP header of the IETF's differentiated services architecture.

The SA-ST schemes include Early Packet Discard (EPD), Partial Packet Discard (PPD) and Random Early Detection (RED). EPD and PPD improve network efficiency because they minimize the transmission of partial packets by the network. Since they do not discriminate between connections in dropping packets, these schemes are unfair in allocating bandwidth to competing connections [LI96]. Random Early Detection (RED) maintains a global threshold for the average queue. When the average queue exceeds this threshold, RED drops packets probabilistically using a uniform random variable as the drop probability.

However, it has been shown in [LIN97] that RED cannot guarantee equal bandwidth sharing. The paper also contains a proposal for Fair Random Early Drop (FRED). FRED maintains per-connection buffer occupancies and drops packets probabilistically if the per-connection occupancy exceeds the average queue length. In addition, FRED ensures that each connection has at least a minimum number of packets in the queue. FRED can be classified as one that maintains per-connection queue lengths, but has a global threshold (MA-ST).

The Selective Drop (SD) [GOYAL97] and Fair Buffer Allocation (FBA) schemes are MA-ST schemes proposed for the ATM UBR service category. These schemes use per-connection accounting to maintain the current buffer utilization of each UBR Virtual Channel (VC). A fair allocation is calculated for each VC, and during congestion (indicated when the total buffer occupancy exceeds a threshold), if the VC's buffer occupancy exceeds its fair allocation, its

subsequent incoming packet is dropped. *Both Selective Drop and FBA improve both fairness and efficiency of TCP over UBR.* This is because cells from overloading connections are dropped in preference to underloading ones.

The Virtual Queuing (VQ) [WU97] scheme achieves equal buffer allocation by emulating on a single FIFO queue, a per-VC queued round-robin server. At each cell transmit time, a per-VC variable (γ_i) is decremented in a round-robin manner, and is incremented whenever a cell of that VC is admitted in the buffer. When γ_i exceeds a fixed threshold, incoming packets of the ith VC are dropped. An enhancement called Dynamic EPD changes the above drop threshold to include only those sessions that are sending less than their fair shares.

Since the above MA-ST schemes compare the per-connection queue lengths (or virtual variables with equal weights) with a global threshold, they can only guarantee equal buffer occupancy (and thus throughput) to the competing connections. These schemes do not allow for specifying a guaranteed rate for connections or groups of connections. Moreover, in their present forms, they cannot support packet discard levels based on tagging.

Another enhancement to VQ, called MCR scheduling [SIU97], proposes the emulation of a weighted scheduler to provide Minimum Cell Rate (MCR) guarantees to ATM connections. In this scheme, a per-VC, weighted variable (W_i) is updated in proportion to the VCs MCR, and compared with a global threshold. [FENG] proposes a combination of a Packet Marking Engine (PME) and an Enhanced RED scheme based on per-connection accounting and multiple thresholds (MA-MT). PME+ERED is designed for the IETF's differentiated services architecture, and can provide loose rate guarantees to connections. The PME measures per-connection bandwidths and probabilistically marks packets if the measured bandwidths are lower than the target bandwidths (multiple thresholds). High priority packets are marked, and low priority packets are unmarked. The ERED mechanism is similar to RED except that the probability of discarding marked packets is lower that that of discarding unmarked packets.

The DFBA scheme [GOYAL98b] proposed for the ATM GFR service provides MCR guarantees for VCs carrying multiple TCP connections. DFBA maintains high and low target buffer

occupancy levels for each VC, and performs probabilistic drop based on a VCs buffer occupancy and its target thresholds. The scheme gives priority to CLP=0 packets over CLP=1 packets.

A simple SA-MT scheme can be designed that implements multiple thresholds based on the packet discard levels. When the global queue length (single accounting) exceeds the first threshold, packets with the lowest discard level are dropped. When the queue length exceeds the next threshold, packets from the lowest and the next discard level are dropped. This process continues until EPD/PPD is performed on all packets.

As discussed in the previous section, for satellite-ATM networks, TCP congestion control mechanisms have more effect on TCP throughput than ATM buffer management policies. However, these drop policies are necessary to provide fair allocation of link capacity, to provide differentiated services based on discard levels, and to provide minimum cell rate guarantees to low priority VCs. The Guaranteed Frame Rate service describes in the next section makes extensive use of the intelligent buffer management policies described here.

4. Guaranteed Frame Rate

The GFR service guarantee requires the specification of a minimum cell rate (MCR) and a maximum frame size (MFS) for each VC. If the user sends packets (or frames) of size at most MFS, at a rate less than the MCR, then all the packets are expected to be delivered by the network with low loss. If the user sends packets at a rate higher than the MCR, it should still receive at least the minimum rate. The minimum rate is guaranteed to the untagged (CLP=0) frames of the connection. In addition, a connection sending in excess of the minimum rate should receive a fair share of any unused network capacity. The exact specification of the fair share has been left unspecified by the ATM Forum.

There are three basic design options that can be used by the network to provide the per-VC minimum rate guarantees for GFR -- tagging, buffer management, and queueing:

- **Tagging**: Network based tagging (or policing) can be used as a means of marking non-conforming packets before they enter the network. This form of tagging is usually performed

when the connection enters the network. Figure 2 shows the role of network based tagging in providing a minimum rate service in a network. Network based tagging on a per-VC level requires some per-VC state information to be maintained by the network and increases the complexity of the network element. Tagging can isolate conforming and non-conforming traffic of each VC so that other rate enforcing mechanisms can use this information to schedule the conforming traffic in preference to non-conforming traffic.

- **Buffer management**: Buffer management is typically performed by a network element (like a switch or a router) to control the number of packets entering its buffers. In a shared buffer environment, where multiple VCs share common buffer space, per-VC buffer management can control the buffer occupancies of individual VCs. Figure 2 shows the role of buffer management in the connection path. The DFBA scheme can be used by the on-board ATM network to provide minimum rate guarantees to GFR VCs.

- **Scheduling**: Figure 2 illustrates the position of scheduling in providing rate guarantees. While tagging and buffer management, control the entry of packets into a network element, queuing strategies determine how packets are scheduled onto the next hop. FIFO queuing cannot isolate packets from various VCs (or groups of VCs) at the egress of the queue. Per-VC queuing, on the other hand, maintains a separate queue for each VC (or groups of VCs) in the buffer. A scheduling mechanism can select between the queues at each scheduling time. However, scheduling adds the cost of per-VC queuing and the service discipline. For a simple service like GFR, this additional cost may be undesirable for an on-board switch.

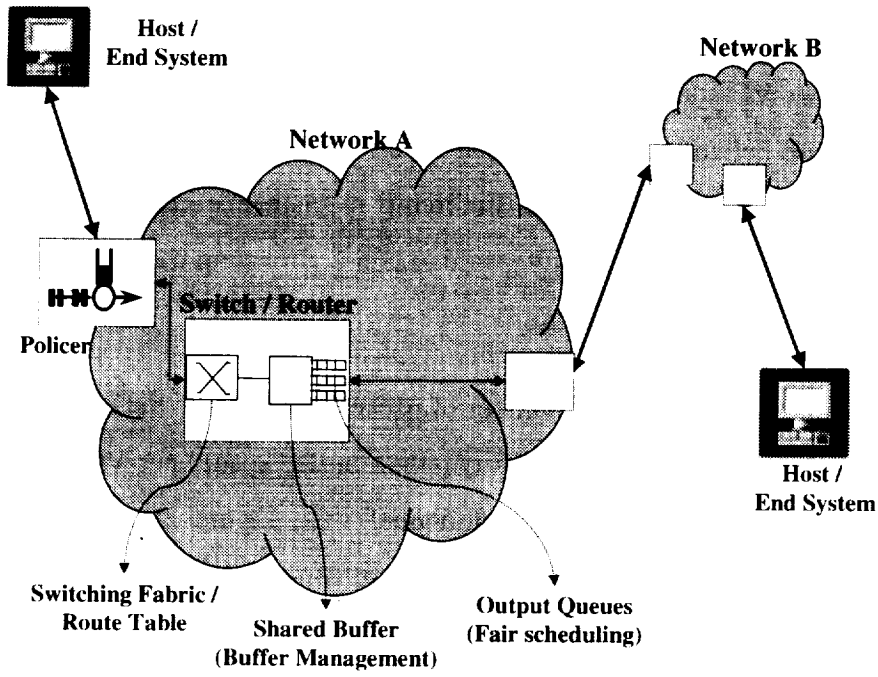


Figure 2 Buffering, Scheduling and Policing in the Network

5. ABR over Satellite

[KALYAN97] provides a comprehensive study of TCP performance over the ABR service category. We discuss a key feature ABR called virtual source/virtual destination, and highlight its relevance to long delay paths. Most of the discussion assumes that the switches implement a rate based switch algorithm like ERICA+. Credit based congestion control for satellite networks has also been suggested. However, in this paper, we focus on rate based control as specified in the ATM standards.

In long latency satellite configurations, the feedback delay is the dominant factor (over round trip time) in determining the maximum queue length. A feedback delay of 10 ms corresponds to about 3670 cells of queue for TCP over ERICA, while a feedback delay 550 ms corresponds to 201850 cells. This indicates that satellite switches need to provide at least one feedback delay worth of buffering to avoid loss on these high delay paths. A point to consider is that these large queues should not be seen in downstream workgroup or WAN switches, because they will not

provide so much buffering. Satellite switches can isolate downstream switches from such large queues by implementing the virtual source/virtual destination (VS/VD) option.

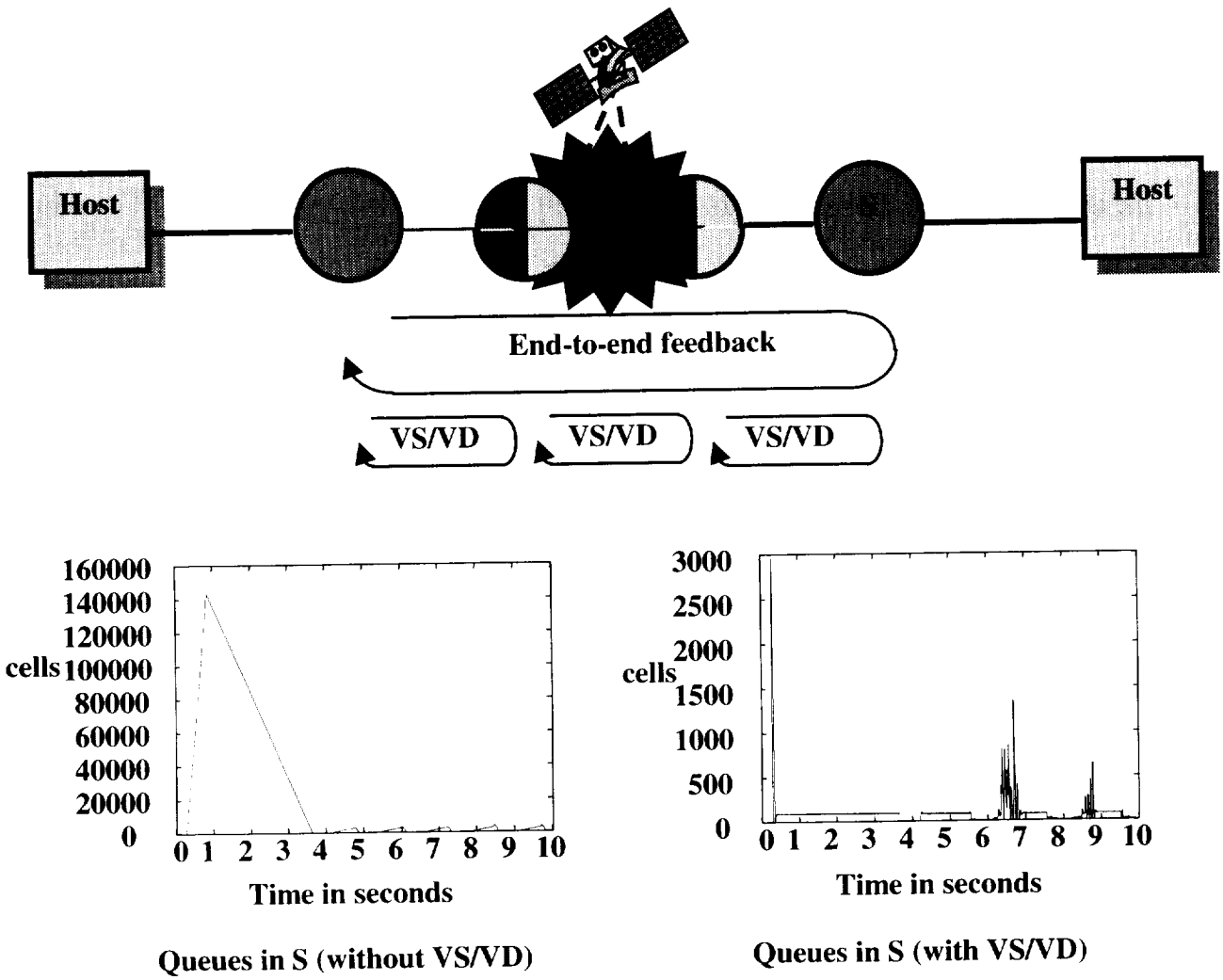


Figure 3 The VS/VD option in ATM-ABR

[GOYAL98a] has examined some basic issues in designing VS/VD feedback control mechanisms. VS/VD can effectively isolate nodes in different VS/VD loops. As a result, the buffer requirements of a node are bounded by the feedback delay-bandwidth product of the upstream VS/VD loop. VS/VD helps in reducing the buffer requirements of terrestrial switches that are connected to satellite gateways. Figure 3 illustrates a the results of a simulation experiment showing the effect of VS/VD on the buffer requirements of the terrestrial switch S. In the figure the link between S and end host is the bottleneck link. The feedback delay-bandwidth

product of the satellite hop is about 16000 cells, and dominates the feedback delay-bandwidth product of the terrestrial hop (about 3000 cells). Without VS/VD, the terrestrial switch S, that is a bottleneck, must buffer cells of upto the feedback delay-bandwidth product of the entire control loop (including the satellite hop). With a VS/VD loop between the satellite and the terrestrial switch, the queue accumulation due to the satellite feedback delay is confined to the satellite switch. The terrestrial switch only buffers cells that are accumulated due to the feedback delay of the terrestrial link to the satellite switch.

6. Comparison of ATM Service Categories

Existing and proposed ATM standards provide several options for TCP/IP data transport over a satellite-ATM network. The three service categories -- ABR, UBR and GFR -- and their various implementation options present a cost-performance tradeoff for TCP/IP over ATM. A comparison of the service categories can be based on the following factors

- Implementation Complexity
- Buffering requirements for switches and ATM end-systems
- Network bandwidth utilization
- Bandwidth allocation (fairness and MCR guarantees)

Higher complexity arises from resource allocation algorithms for Connection Admission Control (CAC) and Usage Parameter Control (UPC), as well as from sophisticated queuing and feedback control mechanisms. While UPC is performed at the entrance of the ATM network to control the rate of packets entering the network, CAC is performed during connection establishment by each network element. UBR is the least complex service category because it does not require any CAC or UPC. Typical UBR switches are expected to have a single queue for all UBR VCs. Buffer management in switches can vary from a simple tail drop to the more complex per-VC accounting based algorithms such as FBA. An MCR guarantee to the UBR service would require a scheduling algorithm that prevents the starvation of the UBR queue. The GFR service could be implemented by either a single queue using a DFBA like mechanism, or per-VC queues and scheduling. The ABR service can be implemented with a single ABR queue in the switch. The

VS/VD option requires the use of per-VC queuing and increases the implementation complexity of ABR. The CAC requirements for GFR and ABR are similar. However, the tagging option, CLP conformance and MFS conformance tests in GFR add complexity to the UPC function.

The additional complexity for ABR feedback control presents a tradeoff with ABR buffer requirements. Network buffering is lower for ABR than for UBR or GFR. In addition, ABR has controlled buffer requirements that depend on the bandwidth-delay product of the ABR feedback loop. At the edge of the ATM network, network feedback can provide information for buffer dimensioning. Large buffers in edge routers can be used when the ABR network is temporarily congested. In the case of UBR and GFR, edge devices do not have network congestion information, and simply send the data into the ATM network as fast as they can. As a result, extra buffers at the edge of the network do not help for UBR or GFR. This is an important consideration for large delay bandwidth satellite networks. With ABR, satellite gateways (routers at the edges of a satellite-ATM network) can buffer large amounts of data, while the buffer requirements of the on-board ATM switches can be minimized. The buffer requirements with UBR/GFR are reversed for the gateways and on-board switches.

The ABR service can make effective use of available network capacity by providing feedback to the sources. Edge devices with buffered data can fill up the bandwidth within one feedback cycle of the bandwidth becoming available. This feedback cycle is large for satellite networks. With UBR and GFR, available bandwidth can be immediately filled up by edge devices that buffer data. However, the edge devices have no control on the sending rate, and data is likely to be dropped during congestion. This data must be retransmitted by TCP, and can result in inefficient use of the satellite capacity.

In addition to efficient network utilization, a satellite-ATM network must also fairly allocate network bandwidth to the competing VCs. While vanilla UBR has no mechanism for fair bandwidth allocation, UBR or GFR with buffer management can provide per-VC fairness. ABR provides fairness by per-VC rate allocation. A typical satellite ATM network will carry multiple TCP connections over a single VC. In ABR, most losses are in the routers at the edges of the network, and there routers can perform fair buffer management to ensure IP level fairness. In UBR and GFR on the other hand, most losses due to congestion are in the satellite-ATM

network, where there is no knowledge of the individual IP flows. In this case, fairness can only be provided at the VC level.

7. Concluding Remarks

Several issues arise in optimizing the performance of TCP when ATM is deployed over satellite links. This paper emphasizes that both TCP mechanisms as well as ATM mechanisms should be used to improve TCP performance over long-delay ATM networks. ATM technology provides at least 3 service categories for data: UBR, ABR, and GFR. Each of these categories can be improved by a number of mechanisms including:

- UBR with intelligent buffer management,
- UBR with guaranteed rate,
- ABR with network feedback,
- ABR with virtual source/virtual destination (VS/VD),

In addition, TCP provides several congestion control mechanisms including:

- Vanilla TCP with slow start and congestion avoidance,
- TCP Reno with fast retransmit and recovery,
- TCP New Reno
- TCP with selective acknowledgements (SACK)

It has been shown that vanilla TCP over the UBR service category achieves low throughput and high unfairness over satellite networks. This is because during packet loss, TCP loses time waiting for its coarse granularity retransmission timeout. In the presence of bursty packet losses, fast retransmit and recovery (FRR) (without SACK) further hurts TCP performance over UBR for long delay-bandwidth product networks.

Frame level discard policies such as EPD improve the throughput significantly over cell-level discard policies. However, the fairness is not guaranteed unless intelligent buffer management using per-VC accounting is used. Throughput increases further with more aggressive New Reno and SACK. SACK gives the best performance in terms of throughput. It has been found that for long delay paths, the throughput improvement due to SACK is more than that from discard

policies and buffer management. Using guaranteed rates (GR or GFR) helps in the presence of a high load of higher priority traffic such as Constant Bit Rate (CBR) or Variable Bit Rate (VBR) traffic.

For TCP over ABR, virtual source/virtual destination (VS/VD) can be used to isolate long-delay segments from terrestrial segments. This helps in efficiently sizing buffers in routers and ATM switches. As a result, terrestrial switches only need to have buffers proportional to the bandwidth-delay products of the terrestrial segment of the TCP path. Switches connected to the satellite VS/VD loops must have buffers proportional to the satellite delay-bandwidth products.

8. References

[FENG] Wu-chang Feng, Dilip Kandlur, Debanjan Saha, Kang G. Shin, "Techniques for Eliminating Packet Loss in Congested TCP/IP Networks," University of Michigan, CSE-TR-349-97, November 1997.

[GOYAL97] Rohit Goyal, Raj Jain, Shivkumar Kalyanaraman, Sonia Fahmy, Bobby Vandalore, Sastri Kota, "TCP Selective Acknowledgments and UBR Drop Policies to Improve ATM-UBR Performance over Terrestrial and Satellite Networks", Proc. ICCCN '97, Las Vegas, September 1997, pp. 17-27, http://www.cis.ohio-state.edu/~jain/papers/ic3n97.htm

[GOYAL98a] Rohit Goyal, Xiangrong Cai, Raj Jain, Sonia Fahmy, Bobby Vandalore Per-VC rate allocation techniques for ABR feedback in VS/VD networks, Proceedings of Globecom'98, November 1998, http://www.cis.ohio-state.edu/~jain/papers/globecom98.htm

[GOYAL98b] Rohit Goyal, Raj Jain, Sonia Fahmy, Bobby Vandalore, "Buffer Management for the GFR Service," ATM_Forum/98-0405, July 1998, http://www.cis.ohio-state.edu/~jain/atmf/a98-0405.htm

[HOE96] Janey C. Hoe, "Improving the Start-up Behavior of a Congestion Control Scheme for TCP," Proceedings of SIGCOMM'96, August 1996.

[KALYAN97] Shivkumar Kalyanaraman, "Traffic Management for the Available Bit Rate (ABR) Service in Asynchronous Transfer Mode (ATM) Networks," PhD Dissertation, The Ohio State University, 1997.

[LI96] H. Li, K.Y. Siu, H.T. Tzeng, C. Ikeda and H. Suzuki, "TCP over ABR and UBR Services in ATM," Proc. IPCCC'96, March 1996.

[LIN97] Dong Lin, Robert Morris, "Dynamics of Random Early Detection," Proceedings of SIGCOMM97, 1997.

[SIU97] Kai-Yeung Siu, Yuan Wu, Wenge Ren, "Virtual Queuing Techniques for UBR+ Service in ATM with Fair Access and Minimum Bandwidth Guarantee," Proceedings of Globecom'97, 1997.

[WU97] Yuan Wu, Kai-Yeung Siu, Wenge Ren, "Improved Virtual Queuing and Dynamic EPD Techniques for TCP over ATM," Proceedings of ICNP97, 1997.

Improving the Performance of TCP/IP over Satellite-ATM Networks [*]

Rohit Goyal, Raj Jain, Sonia Fahmy, Bobby Vandalore, Mukul Goyal
Department of Computer and Information Science, The Ohio State University
395 Dreese Lab, 2015 Neil Avenue
Columbus, OH 43210-1277, U.S.A.
E-mail: goyal@cis.ohio-state.edu

Sastri Kota
Lockheed Martin Telecommunications
1272 Borregas Ave, Sunnyvale, CA 94089

Thomas vonDeak
NASA Glenn Research Center, MS: 54-6
21000 Brookpark Road
Cleveland, OH 44135

Abstract: *In this paper we present tehniques to to improve the performance TCP/IP over satellite-ATM networks, and to provide minimum rate guarantees to VCs carrying TCP/IP traffic. Many future systems are proposing to use ATM or ATM like technology to transport TCP/IP based data. These systems must be designed to support best effort services, as well as enhanced services that provide minimum rate guarantees to TCP/IP traffic. In this paper we study four main issues. First, we discuss tehniques for optimizing TCP performance over satellite networks using the Unspecified Bit Rate (UBR) service. We then demonstrate that high priority background traffic can degrade TCP performance over UBR, and we discuss the use of rate guarantees to improve performance. We design a full factorial experiment to assess the buffer requirements for TCP over satellite-ATM networks. Finally, we discuss the use of the Guaranteed Frame Rate (GFR) service category to provide minimum rate guarantees to VCs carrying TCP/IP traffic. We propose the Differential Fair Buffer Allocation (DFBA) scheme for buffer management that provides GFR guarantees to TCP/IP traffic over satellite latencies. We use full factorial experimental design with various latencies, buffer sizes, and TCP types for our simulations.*

Keywords: TCP, ATM, UBR, GFR, Buffer Management, Satellite

[*]This work was partially sponsored by the NASA Glenn Research Center under contract number NAS3-97198

[†]Some results in section 3 have appeared in [6]. Results in section 5 have appeared in [7]. This paper is a much enhanced and consolidated version, and no results have been published in a journal.

1 Introduction

The TCP over Satellite (TCPSAT) working group in the IETF has designed mechanisms for transporting TCP over satellite networks. The group has focussed its efforts on modifying the TCP protocol so that its performance over satellite links is improved. While TCP based enhancements do improve satellite network performance, network based techniques should also be used to optimize performance of TCP over satellite networks. The research on TCP is also geared towards a best effort service framework. Recent developments in broadband communications have promoted the design of multiservice network architectures. The implementation of such architectures requires network based mechanisms that support QoS guarantees. Moreover, network based traffic management is required to provide basic service guarantees in a multiservice network. The increasing capabilities of on-board switching and processing architectures have enabled the implementation of relatively complex traffic management mechanisms in the network.

More than 50% of the planned Ka-band satellite systems propose to use on-board ATM or ATM like fast packet switching [13]. ATM based on board switching and processing provide a new set of techniques for traffic management in satellite networks. For satellite systems that do not perform on-board processing, packet switching ground stations can use intelligent techniques for congestion avoidance and improving end-to-end performance. While some of these mechanisms improve performance, they also increase the complexity and hence the cost of designing the network elements. The satellite network architect is faced with the complex decision of designing and deploying earth terminals and on-board switches for optimizing the cost-performance tradeoff.

In this paper, we study the techniques for traffic management in satellite networks for TCP/IP transport. The main goals are to optimize TCP performance over satellite, and to enable the provision of basic service guarantees in the form of guaranteed throughput to TCP data. We discuss these techniques within the framework of ATM technology being deployed over satellite (satellite-ATM networks). However, the general mechanisms and performance results discussed in this paper are equally applicable to any packet switched satellite network. We discuss TCP based enhancements and propose network based ATM mechanisms. We present simulations for

the various TCP and ATM enhancements, and discuss performance results for these experiments. Based on the experimental results and analysis, we provide guidelines for designing satellite-ATM network architectures that can efficiently transport TCP/IP data.

The paper does not propose any new TCP enhancements, but analyzes the performance of existing and proposed TCP mechanisms including Vanilla TCP (with slow start and congestion avoidance) TCP Reno (with fast retransmit and recovery), and TCP SACK (with selective acknowledgements). A performance analysis of TCP New Reno is presented in [14]. In this paper, we present techniques for buffer management, and throughput guarantees using ATM that improve TCP throughput, and are used to provide rate guarantees over satellite-ATM networks. The simulation and analysis are performed for various satellite latencies covering multihop LEO, and GEO systems. The results show that the design considerations for satellite networks are different than those for terrestrial networks, not only with respect to TCP, but also for the network. Several recent papers have analyzed various TCP policies over satellite latencies. These have been listed in [18]. The emphasis on network design issues for traffic management and basic service guarantees for TCP/IP over satellite-ATM is the unique contribution of this research.

2 Problem Statement: TCP over Satellite-ATM

There are three ATM service categories that are primarily designed for best effort data traffic. These are:

Unspecified Bit Rate (UBR): UBR is a best effort service category that provides no guarantees to the user. Past results have shown that TCP performs poorly over UBR. Two main reasons for the poor performance are the coarse grained TCP transmission timeout and TCP synchronization [15]. The performance of TCP over UBR can be enhanced by intelligent drop policies in the network. These drop policies include Early Packet Discard (EPD), Random Early Discard (RED), Selective Drop (SD) and Fair Buffer Allocation (FBA). These are discussed in [16]. Providing minimum rate guarantees to the UBR service category has also been suggested as a means for improving TCP performance over UBR. In this paper, we will

analyze the UBR enhancements for TCP over satellite. The enhanced version of UBR has been informally termed UBR+.

Guaranteed Frame Rate (GFR): GFR is a frame based service that provides a Minimum Cell Rate (MCR) guarantee to VCs. In addition to MCR, GFR also provides a fair share of any unused network capacity. Several implementation options exist for GFR, including, network policing, per-VC scheduling, and intelligent buffer management. In this paper we show how to implement the GFR service using a buffer management algorithm called Differential Fair Buffer Allocation (DFBA). We discuss the performance of DFBA for TCP over satellite-ATM networks.

Available Bit Rate (ABR): The ABR service provides an MCR guarantee to the VCs, and a fair share of any unused capacity. ABR is different from GFR in several ways, but the most important is that ABR uses a rate based closed loop feedback control mechanism for congestion control. ABR also the feedback control to be end-to-end, or be broken into several hops using the virtual source/virtual destination option (VS/VD). A complete description and analysis of ABR and VS/VD is presented in [17]. In in this paper, we focus on TCP performance over UBR and GFR services.

The design of a multiservice satellite network present several architectural options for the ATM network component. These include the choice of the various ATM service categories and their implementations. We study the following design options for a satellite-ATM network supporting efficient services to transport TCP data:

UBR with tail drop or frame based discard like EPD. Among frame based discard policies, the Early Packet Discard [10] policy is widely used [24]. The Early Packet Discard maintains a threshold R, in the switch buffer. When the buffer occupancy exceeds R, then all new incoming packets are dropped. Partially received packets are accepted if possible. It has been shown [16] that for terrestrial networks, EPD improves the efficiency of TCP over UBR but does not improve fairness. We will examine the effect on EPD on satellite latencies.

UBR with intelligent buffer management. The Selective Drop scheme is an example of an intelligent buffer management scheme. This scheme uses per-VC accounting to maintain the current buffer utilization of each UBR VC. A fair allocation is calculated for each VC, and if the VC's buffer occupancy exceeds its fair allocation, its subsequent incoming packet is dropped. The scheme maintains a threshold R, as a fraction of the buffer capacity K. When the total buffer occupancy exceeds R×K, new packets are dropped depending on the VC_i's buffer occupancy (Y_i). In the Selective Drop scheme, a VC's *entire packet* is dropped if

$$(X > R) \text{ AND } (Y_i \times N_a / X > Z)$$

where N_a is the number of active VCs (VCs with at least one cell the buffer), and Z is another threshold parameter ($0 < Z \leq 1$) used to scale the effective drop threshold. In terrestrial networks, SD has been shown to improve the fairness of TCP connections running over UBR [16]. However, the effect of SD over satellite network has not been studied.

UBR with guaranteed rate allocation. A multiservice satellite network will transport higher priority variable bit rate traffic along with UBR traffic. The effect of higher priority traffic on TCP over UBR has not been studied before. Our simulations will show that higher priority traffic can degrade TCP performance in some cases. The results will also show, how rate guarantees to UBR can improve TCP performance in the presence of higher priority traffic.

GFR with buffer management, policing or scheduling. The GFR service enables the support of minimum rate guarantees to data traffic, and can be used to provide basic minimum rate services to data traffic. Currently very few suggested implementations of the GFR service exist. Sample implementations can use a combination of policing, buffer management and scheduling in the network. We will describe a buffer management scheme callled Differential Buffer Management (DFBA) scheme that can be used to implement the GFR service.

In addition to the network based options, there are four TCP congestion control techniques that are of interest in performance analysis over satellite links [18]:

Slow start and congestion avoidance (TCP Vanilla)

Fast retransmit and recovery (TCP Reno)

TCP New Reno

Selective acknowledgements (TCP SACK)

Vanilla and Reno TCP are standard mechanisms that are widely deployed in TCP stacks. TCP New Reno and SACK have recently been proposed as performance enhancements to TCP congestion control, and are being incorporated in TCP implementations. Several studies have reported performance results of the above TCP options over satellite latencies [18]. However, these studies have focussed only on TCP mechanisms, and have not considered intelligent network based traffic management and guaranteed rate policies. Also, the studies are all performed using a best effort service framework. Future broadband satellite networks must support the multiservice IP framework being adopted for terrestrial networks. Satellite networks usn an ATM based cell transport must use network based techniques to provide the service guarantees required for a multiservice network.

In this paper, we address the following components of optimizing TCP performance over Satellite-ATM networks:

Part 1 (Optimizing the performance of TCP over satellite-UBR) *We study the performance of TCP vanilla, TCP Reno, and TCP SACK, with buffer management policies within a best effort framework.*

Part 2 (Effect of higher priority traffic on TCP.) *We show how the performance of TCP degrades in the presence of higher priority traffic sharing the link. We also describe the use of guaranteed rate to improve TCP/UBR performance in the presence of higher priority traffic.*

Part 3 (Buffer requirements for TCP over satellite-UBR.) *We present simulation results to calculate the optimal buffer sizes for a large number of TCP sources over satellites.*

Part 4 (Performance of GFR over satellite.) *We describe the GFR service category, and propose the DFBA scheme that uses a FIFO buffer and provides per-VC minimum rate guarantees to*

TCP traffic.

2.1 Performance Metrics

When ATM networks carry TCP/IP data, the end-to-end performance is measured at the TCP layer in the form of TCP throughput. To measure network performance, the throughputs of all TCPs passing through the bottleneck link are added, and expressed as a fraction of the total capacity of the bottleneck link. This is called the *efficiency* of the network. We now define this formally.

Let N TCP source-destination pairs send data over a network with bottleneck link capacity R bits/sec. Let x_i be the observed throughput of the ith TCP source ($0 < i < N$). Let C be the maximum TCP throughput achievable on the link. Let E be the efficiency of the network.

Definition 1 (Efficiency, E) *The Efficiency of the network is the ratio of the sum of the actual TCP throughputs to the maximum possible throughput achievable at the TCP layer.*

$$E(x_1, \ldots, x_N, C) = \frac{\sum_{i=1}^{i=N} x_i}{C}$$

The TCP throughputs x_i's are measured at the destination TCP layers. Throughput is defined as the total number of bytes delivered to the destination application (excluding retransmission and losses) divided by the total connection time. This definition is consistent with the definition of *goodput* in [23]

The maximum possible TCP throughput C is the throughput attainable by the TCP layer running over an ATM network with link capacity R. For example consider TCP over UBR on a 155.52 Mbps link (149.7 Mbps after SONET overhead). with a 9180 byte byte TCP MSS. For 9180 bytes of data, the ATM layer receives 9180 bytes of data + 20 bytes of TCP header + 20 bytes of IP header + 8 bytes of LLC header + 8 bytes of AAL5 trailer. These are padded to produce 193 ATM cells. Thus, each TCP segment results in 10229 bytes at the ATM Layer. From this, the maximum possible throughput = 9180/10229 = 89.7% = 135 Mbps approximately. It should be noted that

ATM layer throughput does not necessarily correspond to TCP level throughput, because some bandwidth may be wasted during TCP retransmissions.

In addition to providing high overall throughput, the network must also allocate throughput fairly among competing connections. The definition of fairness is determined by the particular service guarantees. For example, although UBR makes not service guarantees, fairness for TCP over UBR can be defined as the ability for UBR to provide equal throughput to all greedy TCP connections. In ABR and GFR, fairness is determined ability to meet the MCR guarantee, and to share the excess capacity in some reasonable fashion. We measure fairness using the Fairness Index F.

Definition 2 (Fairness Index, F) *The Fairness Index is a function of the variability of the throughput across the TCP connections defined as*

$$F((x_1, e_1), \ldots, (x_n, e_N)) = \frac{(\sum_{i=1}^{1=N} x_i/e_i)^2}{N \times \sum_{i=1}^{i=N} (x_i/e_i)^2}$$

where x_i = observed throughput of the ith TCP connection ($0 < i \leq N$),
and e_i = expected throughput or fair share for connection i.

For a symmetrical configuration using TCP over UBR, e_i can be defined as an equal share of the bottleneck link capacity ($e_i = C/N$). Thus, the fairness index metric applies well to N-source symmetrical configurations. In this case, note that when $x_1 = x_2 = \ldots = x_n$ then fairness index = 1. Also, low values of the fairness index represent poor fairness among the connections. The desired values of the fairness index must be close to 1. We consider a fairness index of 0.99 to be near perfect. A fairness index of 0.9 may or may not be acceptable depending on the application and the number of sources involved. Also note that the fairness index may not be a good metric for a small number of connections. Details on the fairness metric can be found in [19]. This fairness index has been used in several studies including [23]. In general, for a more complex configuration, the value of e_i can be derived from a rigorous formulation of a fairness definition that provides max-min fairness to the connections.

Due to space constraints, in this paper, we do not present extensive fairness results, but provide brief discussions of fairness when appropriate. In [14], we provide more comprehensive fairness

results, and show that with sufficient buffers and a large number of TCP sources, good fairness values are achieved over UBR.

3 TCP over UBR+

Since TCP congestion control is inherently limited by the round trip time, long delay paths have significant effects on the performance of TCP over ATM. A large delay-bandwidth link must be utilized efficiently to be cost effective. In this section, we first present performance results results of TCP over UBR and its enhancements, with satellite delays.

3.1 Performance Results for Satellite Delays

All simulations use the N source configurations shown in Figure 1. All sources are identical and persistent TCP sources i.e., the sources always send a segment as long as it is permitted by the TCP window. Moreover, traffic is unidirectional so that only the sources send data. The destinations only send ACKs. The delayed acknowledgment timer is deactivated, i.e., the receiver sends an ACK as soon as it receives a segment. Each TCP is transported over a single VC. This enables the switch to perform per-TCP control using per-VC control (Selective Drop). When multiple TCPs are aggregrated over a single VC, per-TCP accounting cannot be performed, and the buffer management within a single VC becomes equivalent to EPD or RED. Aggregrated TCP VCs are further discussed in section 6.

We consider the following factors while performing our experiments

TCP mechanism. TCP Vanilla, Reno and SACK as described in section 2.

Round Trip Latency: GEO (550 ms) and LEO (30 ms). Our primary aim is to study the performance of large latency connections. The typical one-way latency from earth station to earth station for a single LEO (700 km altitude, 60 degree elevation angle) hop is about 5 ms [20]. The one-way latencies for multiple LEO hops can easily be up to 50 ms from earth station to earth station. GEO one-way latencies are typically 275 ms from earth station to earth station. For GEO's, the link between the two switches in Figure 1 is a satellite link

with a one-way propagation delay of 275 ms. The links between the TCP sources and the switches are 1 km long. This results in a round trip propagation delay of about 550 ms. The LEO configuration is modeled as a an access uplink to the on board satellite switch, one or more intersatellite hops, and a downlink to the earth terminal. For the set of simulations presented in this section, a single intersatellite link is used, and each link has a propagation delay of 5 ms, resulting in an end to end round trip time of 30 ms.

Switch Buffer Size. The buffer sizes used in the switch are 200,000 cells and 600,000 cells for GEO and 12,000 and 36,000 cells for LEO. These buffer sizes reflect aproximate bandwidth-delay equivalents of 1 RTT and 3 RTTs respectively. Similar buffer sizes have been used in [26] for studying TCP performance over ABR, and it is interesting to assess the performance of UBR in such situations. The relation between buffer sizes and round trip times is further explored in 5.

Switch discard policy. We use two discard policies, Early Packet Discard (EPD) and Selective Drop (SD) as described in section 2.

Higher priority cross traffic and guaranteed rates. We introduce the effects of cross traffic in section 4.

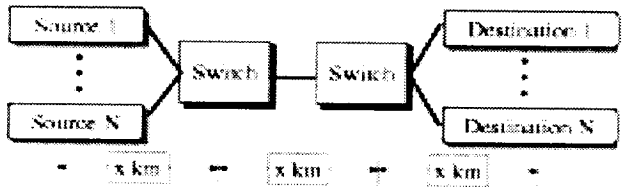


Figure 1: The N source TCP configuration

We first present the results for LEO and GEO systems with the following parameters:

- The number of sources (N) is set to 5. In general, the typical number of simultaneous sources might be active, but our simulations give a good representation of the ability of the TCPs to recover during congestion. In section 5 we further extend these results to a large number of

Table 1: TCP over Satellite (UBR): Efficiency

TCP Type	Buffer Size	Efficiency (LEO) EPD	SD	Efficiency (GEO) EPD	SD
SACK	1RTT	0.90	0.88	0.6	0.72
SACK	3RTT	0.97	0.99	0.99	0.99
Reno	1RTT	0.79	0.8	0.12	0.12
Reno	3RTT	0.75	0.77	0.19	0.22
Vanilla	1RTT	0.9	0.9	0.73	0.73
Vanilla	3RTT	0.81	0.81	0.81	0.82

sources.

- Cells from all TCP sources share a single FIFO queue in the each outgoing link.The FIFO is scheduled according to link availability based on the data rate. In these experiments, no other traffic is present in the network. Cross traffic is introduced in the next section.
- The maximum value of the TCP receiver window is 8,704,000 bytes for GEO and 600,000 bytes for LEO (the window scale factor is used). These window size are sufficient to fill the 155.52 Mbps pipe.
- The TCP maximum segment size is 9180 bytes. A large value is used because most TCP connections over ATM with satellite delays are expected to use large segment sizes.
- The duration of simulation is 40 seconds. This is enough time for the simulations to reach steady state.
- All link bandwidths are 155.52 Mbps.
- The effects of channel access such as DAMA are ignored in our simulations. This simplifies the analysis, and focuses on the properties of buffer management and end-system policies.

Table 1 shows the efficiency values for TCP over UBR with 5 TCP sources. The table lists the efficiency values for three TCP types, 2 buffer sizes, 2 drop policies and the 2 round trip times. Several conclusions can be made from the table:

Conclusion 1 (Performance of SACK) *For long delays, selective acknowledgments significantly*

improve the performance of TCP over UBR. For sufficient buffers, the efficiency values are typically higher for SACK than for Reno and vanilla TCP. This is because SACK often prevents the need for a timeout and can recover quickly from multiple packet losses. Under severe congestion, SACK can perform worse than Vanilla. This is because under severe congestion, retransmitted packets can be dropped, and the SACK sender experiences a timeout. As a result, all SACK information is reneged and the sender starts with a congestion window of 1. The lower efficiency is due to the bandwidth wasted in the aggressive fast retransmission due to SACK. Reduced SACK performance under severe congestion has also been reported in [14].

Conclusion 2 (Performance of fast retransmit and recovery) *As delay increases, fast retransmit and recovery is detrimental to the performance of TCP.* The efficiency numbers for Reno TCP in table 1 are much lower than those of either SACK or Vanilla TCP. This is a well known problem with the fast retransmit and recovery algorithms in the presence of bursty packet loss. When multiple packets are lost during a single round trip time (the same window), TCP Reno reduces its window by half for each lost packet. The reduced window size is not large enough to send new packets that trigger duplicate acks, resulting in a timeout. The timeout occurs at a very low window (because of multiple decreases during fast recovery), and the congestion avoidance phase triggers at a low window. For large RTT, the increase in window during congestion avoidance is very inefficient, and results in much capacity being unused. For a large number of TCPs, the total throughput is greater, but this reflects a worst case scenario and highlights the inneficiency of the congestion avoidance phase for large round trip times. Vanilla TCP performs better, because the first packet loss triggers a timeout when the window is relatively large. The ensuing slow start phase quickly brings the window to half its original value before congestion avoidance sets in.

Conclusion 3 (Performance of buffer management) *The effect of intelligent buffer management policies studied above is not significant in satellite networks.* It has been shown that both EPD and Selective Drop improve the performance of TCP over UBR for terrestrial networks [16]. However, in these experiments, intelligent drop policies have little effect on the performance of TCP over UBR. The primary reason is that in our simulations, we have used adequate buffer sizes

for high performance. Drop policies play a more significant role in improving performance in cases where buffers are a limited resource. These findings are further corroborated for WWW traffic by [14].

4 Effect of Higher Priority Traffic

In the presence of higher priority traffic sharing the satellite link, UBR traffic can be temporarily starved. This may have adverse effects on TCP depending on the bandwidth and duration of the higher priority traffic. Providing a guaranteed rate to UBR traffic has been suggested as a possible solution to prevent bandwidth starvation. The rate guarantee is provided to the entire UBR service category. Per-VC guarantees to UBR are not provided in this architecture. Such a minimum rate guarantee can be implemented using a simple scheduling mechanism like weighted round robin or weighted fair queuing.

To demonstrate the effect of VBR on TCP over UBR, we simulated a five source configuration with a 30 ms round trip time. An additional variable bit rate (VBR) source-destination pair was introduced in the configuration. The VBR traffic followed the same route as the TCP traffic, except that it was put into a separate queue at the output port in each traveresed switch. The VBR queue was given strict priority over the UBR queue, i.e., the UBR queue was serviced only if the VBR queue was empty. The VBR source behaved as an on-off source sending data at a constant rate during the on period, and not sending any data during the off period. On-off background sources have been used in several studies for TCP over UBR and ABR [26]. Three different VBR on/off periods were simulated - 300ms, 100ms and 50ms. In each case, the on times were equal to the off times and, during the on periods, the VBR usage was 100% of the link capacity. The overall VBR usage was thus 50% of the link capacity.

The effect of UBR starvation is seen in table 2. The table shows the efficiency in the presence of VBR traffic. Note that in calculating efficiency, the bandwidth used by the VBR source is taken into account. From the table we can see that longer VBR bursts (for the same average VBR usage of 50%) result in lower throughput for TCP over UBR. At 300 ms on-off times, the efficiency values

Table 2: LEO: SACK TCP with VBR (strict priority) : Efficiency

Buffer (cells)	VBR period (ms)	Efficiency EPD	SD
12000	300	0.43	0.61
36000	300	0.52	0.96
12000	100	0.58	0.70
36000	100	0.97	0.97
12000	50	0.65	0.73
36000	50	0.98	0.98

were very low, even for large buffer sizes. The reason for low throughput was TCP timeout due to starvation. When no TCP packets were sent for periods longer than the TCP RTO value, the source TCP times out and enters slow start. For large buffer sizes, the efficiency was better, because the packets were all queued during starvation.

We also performed simulations with the GEO configuration in the presence of VBR. The corresponding efficiencies for SACK TCP over GEO were much higher than those for LEO. The results (in table 7 in the appendix) show that SACK TCP over GEO achieves near optimal throughput even in the presence of bursty VBR traffic. The performance of Reno TCP was poor, and that corroborates the poor performance of Reno without the VBR sources. Longer periods of starvation (much more than the round trip times) do reduce throughput even in GEOs, but such starvation periods are unrealistic in high bandwidth links. Starvation due to satellite link outages can be of this duration, but this problem cannot be solved by providing rate guarantees, and its study is beyond the scope of this work.

To improve the performance of TCP over LEO delays, we have proposed the use of Guaranteed Rate (GR) for the UBR service category. Guaranteed Rate provides a minimum rate guarantee to the entire UBR service category. UBR cells are queued onto a single FIFO queue. The guarantee is provided using a service discipline like weighted round robin that reserves a least a minimum fraction of the link capacity for the UBR queue.

Figures 2,3,4, and 5 show the key results on effect of a minimum rate guarantee to UBR in the

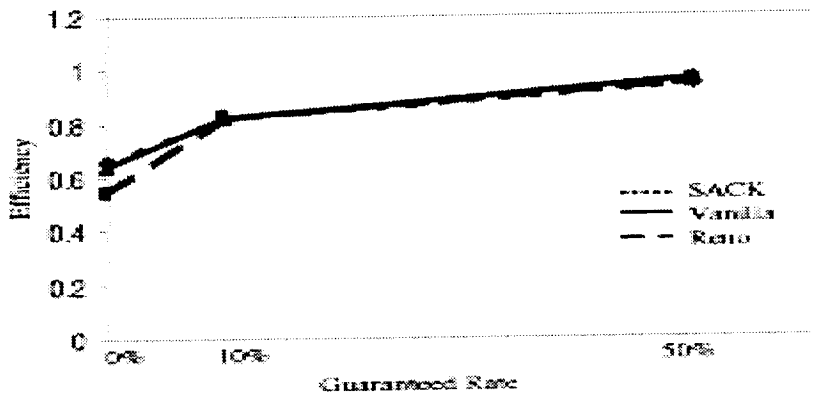


Figure 2: LEO: Guaranteed Rate vs TCP

presence of high priority Variable Bit Rate (VBR) traffic (Table 6 in the appendix lists the complete results in the figures). The VBR traffic is modeled as a simple on-off model with an on-off period of 300ms. The table shows the values of efficiency for 5 and 15 TCP sources and a single VBR source running over a satellite network.

The following parameter values were used for this experment:

- *Number of Sources*: 5 and 15.
- *Buffer size*: 1RTT and 3RTT (delay-bandwidth product).
- *TCP version*: Vanilla, Reno and SACK.
- *Switch Drop Policy*: Selective Drop and Early Packet Discard.
- *Guaranteed Rate*: 0%, 10% and 50% of the total link capacity.
- *Round trip latency*: 30 ms (LEO) and 550 ms (GEO)

In this experiment, we are mainly interested in the effects of TCP, Guaranteed Rate and buffer size. Also, preliminary analysis from the table has shown that the switch drop polices do not have a significant effect on performance. The relative change in efficiencies due to changes in the number of sources is also not significant. The key factors whose effects on efficiency are under study are Guaranteed Rate, buffer size and TCP version.

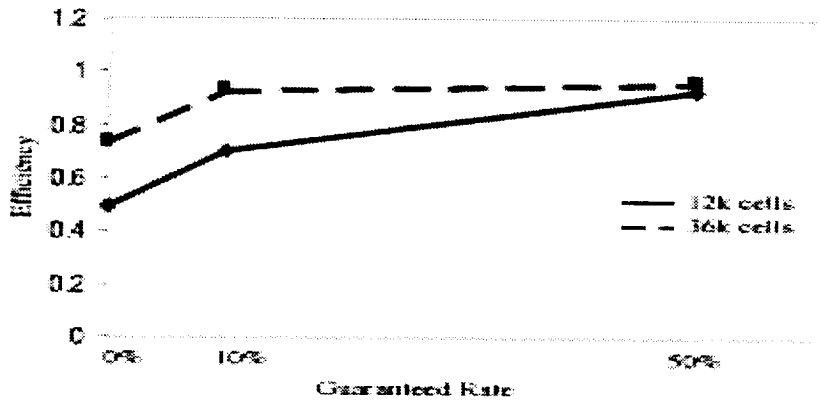


Figure 3: LEO: Guaranteed Rate vs Buffer Size

Figure 2 shows the relative effects of GR and TCP mechanisms on the efficiency for 30 ms RTT. Each point in the figure represents the efficiency value averaged over all the other factors above (Number of sources, Buffer Size and Switch Drop Policy). The figure illustrates that in the presence of high priority traffic, the effect of TCP for smaller round trip times is largely inconsequential. The key determinant is the amount of constant bandwidth allocated to the TCP traffic. The figure shows that even a 10% bandwidth reservation can increase the overall throughput by about 25%.

Figure 3 shows the relative effects of GR and buffer size mechanisms on LEO efficiency. Each point in the figure represents the efficiency value averaged over all the other factors (Number of sources, drop policy, TCP mechanism). The figure shows that a 10% GR allocation increases the efficiency by about 20%. A larger buffer size (36k cells) along with 10% GR can provide high efficiency.

Figures 4, and 5 illustrate the corresponding results for GEO delays. Both figures show that the effect of GR and buffer are incsignificant relative to the effect of TCP. Reno performs very poorly, while SACK performs the best.

The following conclusions can be drawn from the experiments so far:

Conclusion 4 (End system policies vs drop policies) *For longer delays, end system policies are more important than network based drop policies.* For short delays, drop policies have some

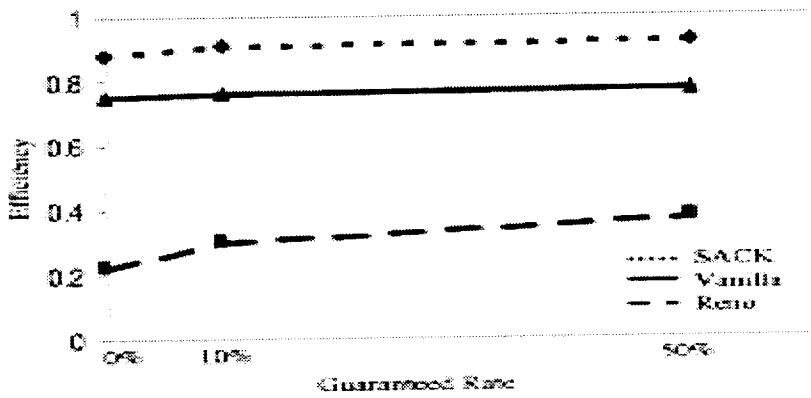


Figure 4: GEO: Guaranteed Rate vs TCP

effect. For long delays, TCP SACK provides the best performance among all TCP mechanisms studied.

Conclusion 5 (Guaranteed rates vs end system policies) *Guaranteed rate helps in the presence of high priority VBR traffic.* The effect of guaranteed rate is more significant for shorter delays (LEO). For longer (GEO) delays, TCP SACK is the most important factor.

In the remainder of the paper, we will present results using TCP SACK. Although SACK is currently not widely deployed, it is quickly becoming the protocol of choice in many new implementations. Moreover, several satellite systems are considering the use of Performance Enhancing Proxies (PEP) [25] over the satellite segment. These proxies will invariably use SACK (or an improvement) on the satellite link, and it is interesting to assess performance using SACK as the desired TCP behavior.

5 Buffer Requirements for TCP over UBR

Our results have shown that small switch buffer sizes can result low TCP throughput over UBR. It is also clear, that the buffer requirements increase with increasing delay-bandwidth product of the connections (provided the TCP window can fill up the pipe). However, the studies have not quantitatively analyzed the effect of buffer sizes on performance. *As a result, it is not clear how*

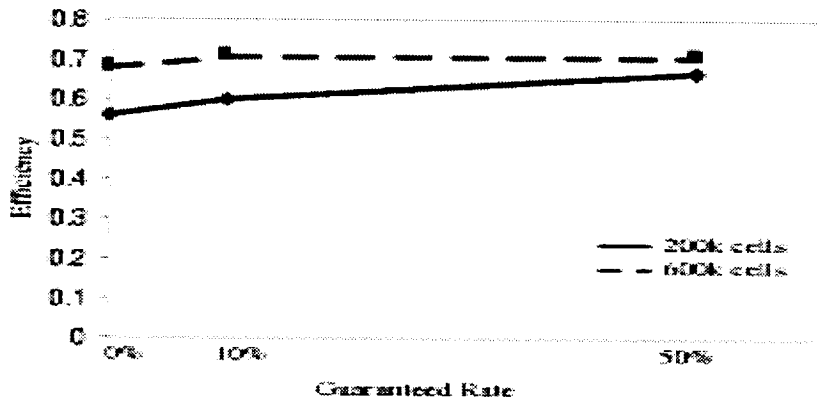


Figure 5: GEO: Guaranteed Rate vs Buffer Size

the increase in buffers affects throughput, and what buffer sizes provide the best cost-performance benefits for TCP/IP over UBR. In this section, we present simulation experiments to assess the buffer requirements for various satellite delay-bandwidth products for TCP/IP over UBR.

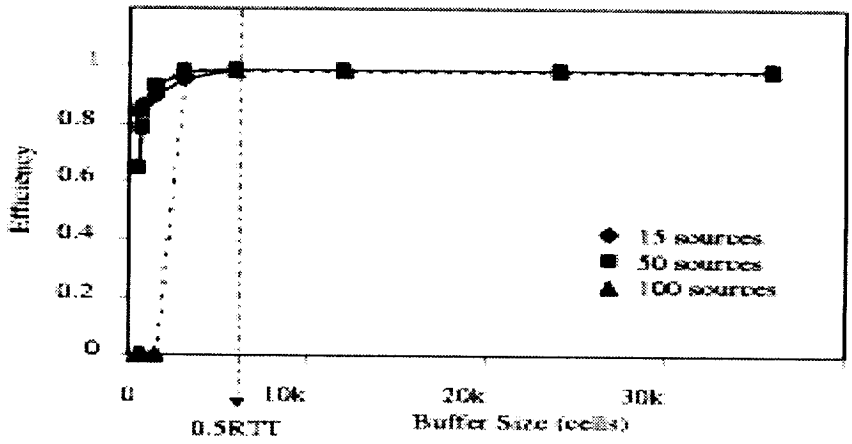


Figure 6: Buffer requirements for single hop LEO

5.1 Parameters

We study the effects of the following parameters:

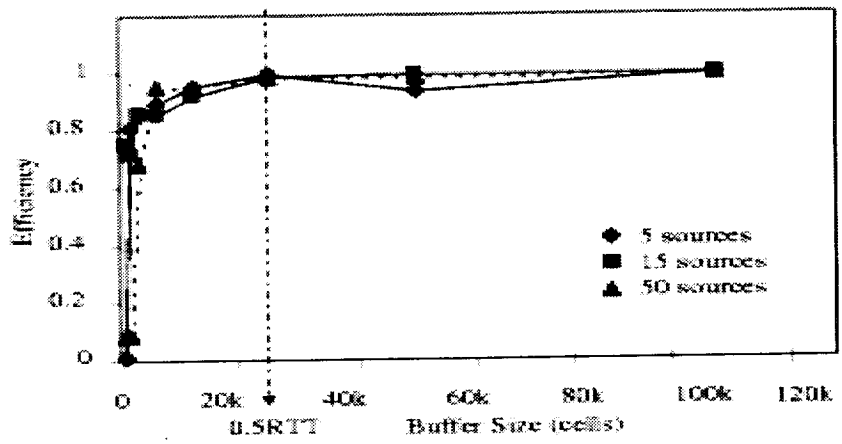


Figure 7: Buffer requirements for multiple hop LEO

Round trip latency. In addition GEO (550 ms round trip) and LEO (30 ms round trip), we also study a multi-hop LEO with an intersatellite one way delay of 50 ms. This results in a round trip time of 120 ms.

Number of sources. To ensure that the results are scalable and general with respect to the number of connections, we will use configurations with 5, 15 and 50 TCP connections on a single bottleneck link. For the single hop LEO configuration, we use 15, 50 and 100 sources.

Buffer size. This is the most important parameter of this study. The set of values chosen are $2^{-k} \times$ Round Trip Time (RTT), $k = -1..6$, (i.e., 2, 1, 0.5, 0.25, 0.125, 0.0625, 0.031, 0.016 multiples of the round trip delay-bandwidth product of the TCP connections.)

The buffer sizes (in cells) used in the switch are the following:

- *LEO (30 ms)*: 375, 750, 1500, 3 K, 6 K, 12 K (=1 RTT) , 24 K and 36 K.

  Multiple LEO (120 ms): 780, 1560, 3125, 6250, 12.5 K, 50 K (=1 RTT) , and 100 K.

- *GEO (550 ms)*: 3375, 6750, 12500, 25 K, 50 K, 100 K, 200 K (=1 RTT) , and 400 K.

Switch drop policy. We use the per-VC buffer allocation policy, Selective Drop (see [16]) to fairly allocate switch buffers to the competing connections.

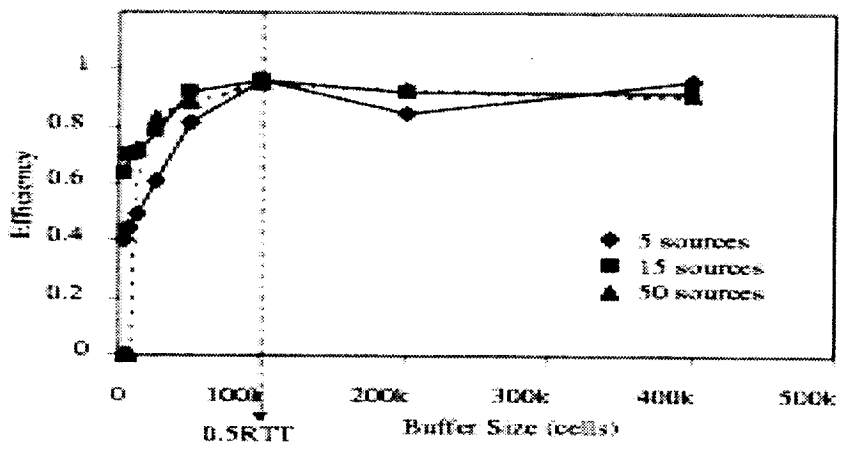


Figure 8: Buffer requirements for GEO

End system policies. We use SACK TCP for this study. Further details about our SACK TCP implementation can be found in [6].

The maximum value of the TCP receiver window is 600000 bytes, 2500000 bytes and 8704000 bytes for single hop LEO, multi-hop LEO and GEO respectively. These window sizes are sufficient to fill the 155.52 Mbps links. The TCP maximum segment size is 9180 bytes. The duration of simulation is 100 seconds for multi-hop LEO and GEO and 20 secs for single hop LEO configuration. These are enough for the simulations to reach steady state. All link bandwidths are 155.52 Mbps, and peak cell rate at the ATM layer is 149.7 Mbps after the SONET overhead.

We plot the buffer size against the achieved TCP throughput for different delay-bandwidth products and number of sources. The asymptotic nature of this graph provides information about the optimal buffer size for the best cost-performance ratio.

5.2 Simulation Results

Figures 6, 7 and 8 show the resulting TCP efficiencies for the 3 different latencies. Each point in the figure shows the efficiency (total achieved TCP throughput divided by maximum possible throughput) against the buffer size used. Each figure plots a different latency, and each set of points (connected by a line) in a figure represents a particular value of N (the number of sources).

For very small buffer sizes, (0.016×RTT, 0.031×RTT, 0.0625×RTT), the resulting TCP throughput is very low. In fact, for a large number of sources (N=50) , the throughput is sometimes close to zero. For moderate buffer sizes (less then 1 round trip delay-bandwidth), TCP throughput increases with increasing buffer sizes. TCP throughput asymptotically approaches the maximal value with further increase in buffer sizes. TCP performance over UBR for sufficiently large buffer sizes is scalable with respect to the number of TCP sources. The throughput is never 100%, but for buffers greater than 0.5×RTT, the average TCP throughput is over 98% irrespective of the number of sources. Fairness (not shown here) is high for a large number of sources. This shows that TCP sources with a good per-VC buffer allocation policy like selective drop, can effectively share the link bandwidth.

The knee of the buffer versus throughput graph is more pronounced for larger number of sources. For a large number of sources, TCP performance is very poor for small buffers, but jumps dramatically with sufficient buffering and then stays about the same. For smaller number of sources, the increase in throughput with increasing buffers is more gradual.

For large round trip delays, and a small number of sources, a buffer of 1 RTT or more can result in a slightly reduced throughput (see figures 7 and 8). This is because of the variability in the TCP retransmission timer value. When the round trip is of the order of the TCP timer granularity (100 ms in this experiment), and the queuing delay is also of the order of the round trip time, the retransmission timeout values become very variable. This may result in false timeouts and retransmissions thus reducing throughput.

Conclusion 6 (Buffer requirements for TCP over satellite) *The simulations show that a buffer size of 0.5RTT is sufficient to provide high efficiency and fairness to TCPs over UBR+ for satellite networks.*

6 The Guaranteed Frame Rate Service

The enhancements to TCP over UBR can provide high throughput to TCP connections over satellite networks. However, UBR does not provide any guarantees to its VCs. The service received by

UBR connection is implementation dependent. Service guarantees may be useful for a satellite-ATM network connecting multiple network clouds of Virtual Private Networks. It may be desirable to provide minimum rate guarantees to VCs of each VPN. Per-VC minimum rate guarantees can be implemented using either the Guaranteed Frame Rate (GFR) service or the Available Bit Rate (ABR) service. In this section we will describe how to implement per-VC minimum rate guarantees for the GFR service over satellite networks.

Guaranteed Frame Rate provides a minimum rate guarantee to VCs, and allows for the fair usage of any extra network bandwidth. GFR is a frame based service and uses AAL5 which enables frame boundaries to be visible at the ATM layer. The service requires the specification of a maximum frame size (MFS) of the VC. If the user sends packets (or frames) smaller than the maximum frame size, at a rate less than the minimum cell rate (MCR), then all the packets are expected to be delivered by the network with minimum loss. If the user sends packets at a rate higher than the MCR, it should still receive at least the minimum rate. A leaky bucket like mechanism called Frame-GCRA is used to determine fir a frame is eligible for MCR guarantees. Such frames are called QoS eligible. The minimum rate is guaranteed to the CLP=0 frames of the connection. In addition, a connection sending in excess of the minimum rate should receive a fair share of any unused network capacity. The exact specification of the fair share has been left unspecified by the ATM Forum.

GFR requires minimum signaling and connection management functions, and depends on the network's ability to provide a minimum rate to each VC. GFR is likely to be used by applications that can neither specify the traffic parameters needed for a VBR VC, nor have capability for ABR (for rate based feedback control). Current internetworking applications fall into this category, and are not designed to run over QoS based networks. These applications could benefit from a minimum rate guarantee by the network, along with an opportunity to fairly use any additional bandwidth left over from higher priority connections. The detailed GFR specification is provided in [1], but the above discussion captures the essence of the service.

6.1 GFR Implementation Options

There are three basic design options that can be used by the *network* to provide the per-VC minimum rate guarantees for GFR - tagging, buffer management, and queueing:

Tagging: Network based tagging (or policing) can be used to mark non-eligible packets before they enter the network. Network based tagging on a per-VC level requires some per-VC state information to be maintained by the network and increases the complexity of the network element. Tagging can isolate eligible and non-eligible traffic of each VC so that other rate enforcing mechanisms can use this information to schedule the conforming traffic in preference to non-conforming traffic.

Buffer management: Buffer management is typically performed by a network element to control the number of packets entering its buffers. In a shared buffer environment, where multiple VCs share common buffer space, per-VC accounting can control the buffer occupancies of individual VCs. Per-VC accounting introduces overhead, but without per-VC accounting it is difficult to control the buffer occupancies of individual VCs (unless non-conforming packets are dropped at the entrance to the network by the policer). Note that per-VC buffer management uses a single FIFO queue for all the VCs. This is different from per-VC queuing and scheduling discussed below.

Scheduling: While tagging and buffer management control the entry of packets into a network element, queuing strategies determine how packets are scheduled onto the next hop. Per-VC queuing maintains a separate queue for each VC in the buffer. A scheduling mechanism can select between the queues at each scheduling time. However, scheduling adds the cost of per-VC queuing and the service discipline. For a simple service like GFR, this additional cost may be undesirable.

A desirable implementation of GFR is to use a single queue for all GFR VCs, and provide minimum rate guarantees by means of intelligent buffer management policies on the FIFO. Several proposals have been made [2, 3, 4] to provide rate guarantees to TCP sources with FIFO queuing in the

network. The bursty nature of TCP traffic makes it difficult to provide per-VC rate guarantees using FIFO queuing. In these proposals, per-VC scheduling was recommended to provide rate guarantees to TCP connections. However, all these studies were performed at high target network utilization, i.e., most of the network capacity was allocated to the MCRs. The designers of the GFR service have intended to allocate MCRs conservatively. Moreover, these proposals are very aggressive in dropping TCP packets causing TCP to timeout and lose throughput. All the above studies have examined TCP traffic with a single TCP per VC. However, routers that use GFR VCs, will multiplex many TCP connections over a single VC. For VCs with several aggregated TCPs, per-VC control is unaware of each TCP in the VC. Moreover, aggregate TCP traffic characteristics and control requirements may be different from those of single TCP streams.

In the next subsection, we will briefly describe a buffer management policy called Differential Fair Buffer Allocation (DFBA) that provides per-VC minimum rate guarantees. We present the performance of DFBA for LEO and GEO systems. A complete analysis of DFBA for terrestrial networks is presented in [21].

6.2 The Differential Fair Buffer Allocation Scheme

The Differential Fair Buffer Allocation (DFBA) scheme is based on per-VC accounting on a FIFO buffer. The scheme maintains efficiency and fairness in the network by selectively accepting or discarding incoming cells of a VC. Once the cells are queued, they are serviced in a FIFO manner from the GFR queue. DFBA recognizes frame boundaries using the EOM bit in the last cell of a frame. As a result, DFBA is fully compliant with the GFR requirements specified by the ATM forum.

DFBA uses the current queue length (buffer occupancy) as an indicator of network load. The scheme tries to maintain an optimal load so that the network is efficiently utilized, yet not congested. Figure 9 illustrates the operating region for DFBA. The high threshold (H) and the low threshold (L) represent the cliff and the knee respectively of the classical load versus delay/throughput graph. The goal is to operate between the knee and the cliff.

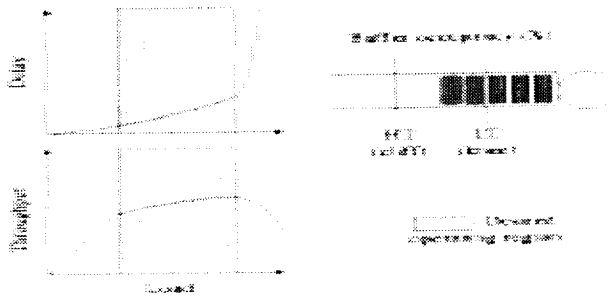


Figure 9: DFBA Target Operating Region

In addition to efficient network utilization, DFBA is designed to allocate buffer capacity fairly amongst competing VCs. This allocation is proportional to the MCRs of the respective VCs. The following variables are used by DFBA to fairly allocate buffer space:

X = Total buffer occupancy at any given time

L = Low buffer threshold

H = High buffer threshold

MCR_i = MCR guaranteed to VC_i

W_i = Weight of $VC_i = MCR_i/(\text{GFR capacity})$

$W = \Sigma W_i$

X_i = Per-VC buffer occupancy ($X = \Sigma X_i$)

Z_i = Parameter ($0 \leq Z_i \leq 1$)

DFBA maintains the total buffer occupancy (X) between L and H. When X falls below L, the scheme attempts to bring the system to efficient utilization by accepting all incoming packets. When X rises above H, the scheme tries to control congestion by performing EPD. When X is between L and H, DFBA attempts to allocate buffer space in proportional to the MCRs, as determined by the W_i for each VC. When X is between L and H, the scheme also drops low priority (CLP=1) packets so as to ensure that sufficient buffer occupancy is available for CLP=0 packets.

Figure 10 illustrates the four operating regions of DFBA. The graph shows a plot of the current buffer occupancy X versus the normalized fair buffer occupancy ($\bar{X}_i$) for VC_i. If VC_i has a weight W_i, then its target buffer occupancy (X_i) should be $X \times W_i/W$. Thus, the normalized buffer occupancy of VC_i can be defined as $\bar{X}_i = X_i \times W/W_i$. The goal is to keep $\bar{X}_i$ as close to X as possible, as indicated by the solid $y = x$ line in the graph. Region 1 is the underload region, in which the current buffer occupancy is less than the low threshold L. In this case, the scheme tries to improve efficiency. Region 2 is the region with mild congestion because X is above L. As a result, any incoming packets with CLP=1 are dropped. Region 2 also indicates that VC_i has a larger buffer occupancy than its fair share (since $X_i > X \times W_i/W$). As a result, in this region, the scheme drops some incoming CLP=0 packets of VC_i, as an indication to the VC that it is using more than its fair share. In region 3, there is mild congestion, but VC_i's buffer occupancy is below its fair share. As a result, only CLP=1 packets of a VC are dropped when the VC is in region 3. Finally, region 4 indicates severe congestion, and EPD is performed here.

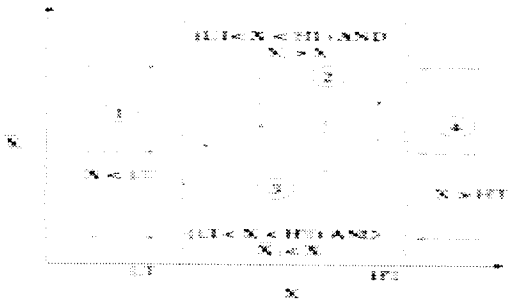


Figure 10: DFBA Drop Regions

In region 2, the packets of VC_i are dropped in a probabilistic manner. This drop behavior is controlled by the drop probability function P{drop}. This is further discussed below.

The probability for dropping packets from a VC when it is in region 2 can be based on several factors. Probabilistic drop is used by several schemes including RED and FRED. The purpose of probabilistic drop is to notify TCP of congestion so that TCP backs off without a timeout. An aggressive drop policy will result in a TCP timeout. Different drop probability functions have different effects on TCP behavior. In general, a simple probability function can use RED like drop,

while a more complex function can depend on all the variables defined above. The drop probability used in our simulations is described in detail in [21] and is given by:

$$P\{drop\} = Z_i \times (\alpha \times \frac{X_i - X \times W_i/W}{X \times (1 - W_i/W)} + (1 - \alpha)\frac{X - L}{H - L})$$

For satellite latencies, an important parameter in this equation is Z_i. It has been shown [22] that for a given TCP connection, a higher packet loss rate results in a lower average TCP window. As a result, a higher drop probability also results in a lower TCP window. In fact, it has been shown [22], that for random packet loss, the average TCP window size is inversely proportional to the square root of the packet loss probability. As a result, the average TCP data rate D is given by

$$D \propto \frac{MSS}{RTT\sqrt{P\{\text{drop}\}}}$$

The data rate is in fact determined by the window size and the RTT of the connection. To maintain a high data rate, the desired window size should be large. As a result, the drop probability should be small. Similarly when the RTT is large, a larger window is needed to support the same data rate (since the delay-bandwidth product increases). As a result, a smaller drop rate should be used. DFBA can be tuned to choose a small Z_i for large latency VCs, as in the case of switches connected to satellite hops, or for VCs with high MCRs. The inherent limitation of any buffer management scheme that depends only on local state is seen here. In general, the switch does not know the RTT of a VC. The switch must estimate a connection's RTT using local state such as the propagation delay of its outgoing links. In case of satellite switches, this propagation delay is likely to be the dominant delay in the VCs path. As a result, the local state provides a pretty good estimate of the today delay. Terrestrial switches are limited in this respect. This limititation is also discussed in [23].

Another potential limitation of any such scheme is that the granularity of fairness is limited by the granularity of flows. The fairness is guaranteed between VCs but not within the TCPs of each. This limitation is not only peculiar to ATM but also to IP. IP routers typically define flows according to IP address or network address source-destination pairs. TCP/UDP port level granularities are not a

scalable solution for backbone networks. As as result, the TCP connections within an IP flow suffer the same kind of unfairness as TCP connections within ATM VCs. However, the probabilistic drop randomizes the packets dropped within a VC. Thus, the scheme can maintain RED like fairness among the TCPs within a VC. This can be accomplished by using a RED like drop probability for drop.

6.3 Simulation Results

The test results presented here are with DFBA for Satellite-ATM interconnected TCP/IP networks. Figure 11 illustrates the basic test configuration. The figure shows 5 local IP/ATM edge switches connected to backbone ATM switches that implement GFR. Each local switch carries traffic from multiple TCPs as shown in the figure. The backbone link carries 5 GFR VCs, one from each local network. Each VC thus carries traffic from several TCP connections. We used 20 TCPs per VC for a total of 100 TCPs. The GFR capacity was fixed to the link rate of 155.52 Mbps (approx. 353207 cells per sec). The MCRs were 20, 40, 60, 80 and 100 kcells/sec for VCs 1...5 respectively, giving a total MCR allocation of 85% of the GFR capacity. At the TCP layer, these MCR's translated to expected TCP throughputs of 6.91, 13.82, 20.74, 27.65, 34.56 Mbps respectively. Note that, in GFR deployments, MCRs are expected to be allocated more conservatively, and 85% allocation reflects an upper bound on MCR allocation. Also, these numbers are aggregate numbers for all 20 TCPs for VCs 1 through 5. All TCP sources are persistent TCPs with SACK. Based on previous studies, [5], we set the thresholds L and H to 0.5 and 0.9 of the buffer capacity respectively. A complete parameter study of DFBA is presented in [21].

In figure 11, the access hop is denoted by x, and the backbone hop is denoted by y. Three different simulation configurations are presented below:

WAN with homogeneous RTT. We first present DFBA results with one way backbone delay = 5 ms, and negligible access delay. In this case, three different buffer sizes were simulated in the bottleneck backbone switch – 25000, 6000 and 3000 cells. The goal of this experiment is to illustrate that DFBA achieves the unequal MCR guarantees for each VC. Table 3 lists the expected and achieved throughputs for each VC in the configuration. The achieved

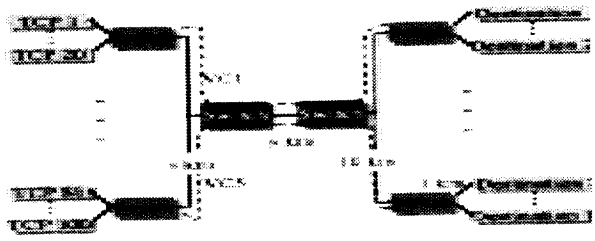


Figure 11: DFBA Simulation Confuration

throughput for a VC is the sum of all the TCP throughputs in that VC. The table illustrates that for each of the buffer sizes, the achieved throughputs exceed the expected throughputs for all VCs. As a result, DFBA provides MCR guarantees to aggregated TCP traffic. The overall efficiency of the system is also more than 95% resulting in high network utilization. In the simulations, the excess capacity (GFR capacity - MCR allocation) is almost equally distributed among the five VCs. This allocation may or may not be considered fair because VCs with higher MCRs may demand a higher portion of the excess. [21] discusses techniques to provide MCR proportional allocation of excess capacity using DFBA.

LEO Access with heterogenous RTT. In this configuration, the access hop (x) for VC 3, is a LEO link with a 25 ms one way delay. This results in a round trip delay of 60 ms for VC3. All other VCs still have negligible access delay, and the backbone delay is also 5 ms one way. The results of this simulation with buffer size = 6000 cells is shown in table 4. The table again shows that DFBA provides the allocated rates to VCs with different MCRs.

GEO backbone Finally, we present the case where the backbone hop is a GEO link. The round trip delay in this case is abount 550 ms. The GEO hop is the most dominant hop with respect to latency, and thus, in the simulation the access hops had negligible latency. Figure 5 shows the achieved throughputs for three different buffer sizes. Again, the table shows that DFBA provides MCR guarantees to VCs over long delay networks.

The ideas and results from this section can be summarized as follows:

Table 3: Minimum rate guarantees with DFBA.

Expected Throughput (Mbps)	Achieved Throughput (Mbps) 25k buffer	6K buffer	3k buffer
6.91	11.29	11.79	10.02
13.82	18.19	18.55	19.32
20.74	26.00	25.13	25.78
27.65	32.35	32.23	32.96
34.56	39.09	38.97	38.56

Table 4: Minimum rate guarantees with DFBA. VC3 = LEO access

Expected Throughput (Mbps)	Achieved Throughput (Mbps)
6.91	10.55
13.82	17.06
20.74	24.22
27.65	33.74
34.56	41.10

Table 5: Minimum rate guarantees with DFBA. GEO backbone

Expected Throughput (Mbps)	Achieved Throughput (Mbps) 200k buffer	150K buffer	100k buffer
6.91	12.4	12.8	11.4
13.82	14.96	16.17	16.99
20.74	21.86	21.63	24.56
27.65	32.10	30.25	33.72
34.56	40.21	39.84	35.52

Conclusion 7 (GFR Service) *The Guaranteed Frame Rate service is designed for frame based best effort applications, and supports per-VC minimum cell rate guarantees.*

Conclusion 8 (GFR Implementation Options) *GFR can be implemented using tagging, buffer management and per-VC scheduling. A desirable implementation of GFR is by using a FIFO buffer with intelligent buffer management.*

Conclusion 9 (DFBA Results) *The Differential Fair Buffer Allocation (DFBA) scheme is a FIFO scheme that provides per-VC MCR guarantees to VCs carrying TCP traffic. Simulations with DFBA show that DFBA can provide such guarantees for terrestrial as well as satellite latencies.*

Conclusion 10 (Limitations) *In general, buffer management schemes for TCP/IP are limited by TCPs dependency on RTT, and the granularity of IP or ATM flows.*

7 Summary of Results

This paper describes a set of techniques for improving the performance of TCP/IP over Asynchronous Transfer Mode (ATM) based satellite networks. Among the service categories provided by ATM networks, the most commonly used category for data traffic is the unspecified bit rate (UBR) service. UBR allows sources to send data into the network without any network guarantees or control.

Several issues arise in optimizing the performance of Transmission Control Protocol (TCP) when ATM-UBR service is used over satellite links. In this paper, we studied several TCP mechanisms as well as ATM-UBR mechanisms to improve TCP performance over long-delay ATM networks. The UBR mechanisms that we studied in this project are:

- UBR with frame level discard policies,
- UBR with intelligent buffer management,
- UBR with guaranteed rate,
- Guaranteed Frame Rate (GFR).

The following TCP mechanisms were studied:

- Vanilla TCP with slow start and congestion avoidance,
- TCP Reno with fast retransmit and recovery,
- TCP with selective acknowledgements (SACK)

We studied several combinations of these mechanisms using an extensive set of simulations and quantified the effect of each of these mechanisms. The following summarizes the list of conclusions drawn from our simulations:

1. In several cases, Vanilla TCP over the UBR service category achieves low throughput and low fairness over satellite networks. This is because during packet loss, TCP loses significant amount of time waiting for retransmission timeout.

2. In the presence of bursty packet losses, fast retransmit and recovery (FRR) (without SACK) further hurts TCP performance over UBR for long delay-bandwidth product networks.

3. Frame level discard policies such as early packet discard (EPD) improve the throughput over cell-level discard policies. However, the fairness is not guaranteed unless intelligent buffer management with per virtual circuit (VC) accounting is used.

4. Throughput increases further with more aggressive New Reno and SACK. SACK gives the best performance in terms of throughput. We found that for long delay paths, the throughput improvement due to SACK is more than that from discard policies and buffer management.

5. A buffer size equal to about half the round-trip delay-bandswidth product of the TCP connections was found to be sufficient for high TCP throughput over satellite-UBR.

6. The presence of bursty high priority cross traffic can degrade the performance of TCP over UBR for terrestrial and low delay satellite networks. The effect of cross traffic is not very significant for GEO because the starvation time is relatively small compared to the round trip time for GEOs

7. Providing guaranteed rate to UBR helps in the presence of a high load of higher priority traffic. We found that reserving just a small fraction, say 10For GEO sytems, the effect of TCP SACK was more significant than other factors.

8. The GFR service category can provide per-VC MCR guarantees. The Differential Fair Buffer Allocation (DFBA) scheme provides MCR guarantees to GFR with a single queue using only per-VC accounting.

The results described above have been based on simulations using persistent TCP traffic. In [14], we have shown that the results also hold for world-wide web TCP traffic.

References

[1] ATM Forum, "ATM Traffic Management Specification Version 4.1," December 1998.

[2] Debashis Basak, Surya Pappu, "GFR Implementation Alternatives with Fair Buffer Allocation Schemes," ATM Forum 97-0528.

[3] Olivier Bonaventure, "A simulation study of TCP with the proposed GFR service category," DAGSTUHL Seminar 9725, High Performance Networks for Multimedia Applications, June 1997, Germany

[4] R. Goyal, R. Jain et.al, "Simulation Experiments with Guaranteed Frame Rate for TCP/IP Traffic," ATM Forum 97-0607.

[5] R. Goyal, R. Jain, S. Kalyanaraman, S. Fahmy and Seong-Cheol Kim, "UBR+: Improving Performance of TCP over ATM-UBR Service," Proc. ICC'97, June 1997.

[6] R. Goyal, R. Jain et.al., "Selective Acknowledgements and UBR+ Drop Policies to Improve TCP/UBR Performance over Terrestrial and Satellite Networks," To appear, Proc. IC3N'97, September 1997. [1]

[1] All our papers and ATM Forum contributions are available from http://www.cis.ohio-state.edu/~jain

[7] Sastri Kota, Rohit Goyal, Raj Jain, "Satellite ATM Network Architectural Considerations and TCP/IP Performance", Proceedings of the 3rd Ka Band Utilization Converence, Italy, September 15-18, 1997, pp. 481-488, http://www.cis.ohio-state.edu/ jain/papers/kaband.htm

[8] Roch Guerin, and Juha Heinanen, "UBR+ Service Category Definition," ATM FORUM 96-1598, December 1996.

[9] Roch Guerin, and Juha Heinanen, "UBR+ Enhancements," ATM FORUM 97-0015, February 1997.

[10] Allyn Romanov, Sally Floyd, "Dynamics of TCP Traffic over ATM Networks," IEEE Journal of Selected Areas In Telecommunications, May 1995.

[11] Dong Lin, Robert Morris, "Dynamics of Random Early Detection," Proceedings of SIGCOMM97, 1997.

[12] Sally Floyd, Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transaction on Networking, August 1993.

[13] C. Ponzoni, "High Data Rate On-board Switching," 3rd Ka-band Utilization Conference, September 1997.

[14] Mukul Goyal, Rohit Goyal, Raj Jain, Bobby Vandalore, Sonia Fahmy, Tom VonDeak, Kul Bhasin, Norm Butts, and Sastri Kota, "Performance Analysis of TCP Enhancements for WWW Traffic using UBR+ with Limited Buffers over Satellite Links", ATM Forum/98-0876R1, December 1998, http://www.cis.ohio-state.edu/ jain/atmf/a98-0876.htm

[15] H. Li, K.Y. Siu, H.T. Tzeng, C. Ikeda and H. Suzuki "TCP over ABR and UBR Services in ATM," Proc. IPCCC'96, March 1996.

[16] Rohit Goyal, Raj Jain, Shiv Kalyanaraman, Sonia Fahmy, Bobby Vandalore, "Improving the Performance of TCP over the ATM-UBR service", Computer Communications, Vol. 21, No. 10, July 1998, pp. 898-911, http://www.cis.ohio-state.edu/ jain/papers/cc.htm

[17] Rohit Goyal, Xiangrong Cai, Raj Jain, Sonia Fahmy, Bobby Vandalore " Per-VC Rate Allocation Techniques for ATM-ABR Virtual Source Virtual Destination Networks," Proceedings of Globecom '98, Australia, November 1998, http://www.cis.ohio-state.edu/ jain/papers/globecom98.htm

[18] Mark Allman, editor, "Ongoing TCP Research Related to Satellites," IETF Draft, Work in Progress.

[19] Raj Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Simulation, and Modeling*, Wiley-Interscience, New York, NY April 1991.

[20] Satellite altitudes taken from "Lloyd's satellite constellation." http://www.ee.surrey.ac.uk/Personal/L.Wood/constellations/overview.html

[21] Rohit Goyal, Raj Jain, Sonia Fahmy, Bobby Vandalore, "Buffer Management for the GFR Service," ATM Forum/98-0405, July 1998, http://www.cis.ohio-state.edu/ jain/atmf/a98-0405.htm

[22] M. Mathis, J. Semke, J. Mahdavi, T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm",Computer Communication Review, volume 27, number 3, pp. 67-82, July 1997.

[23] Floyd, S., "Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic," Computer Communication Review, Vol.21, No.5, October 1991, p. 30-47.

[24] http://www-nrg.ee.lbl.gov/floyd/epd.html

[25] http://tcppep.lerc.nasa.gov/tcppep/

[26] Shivkumar Kalyanaraman, "Traffic Management for the Available Bit Rate (ABR) Service in Asynchronous Transfer Mode (ATM) Networks," PhD Dissertation, The Ohio State University, 1997, xxiv + 429.

Table 6: LEO: TCP with VBR (300ms on/off) over UBR+ with GR : Efficiency

TCP	Buffer	GR	Selective Drop	EPD	
5	12000	SACK	0.5	0.93	0.94
5	12000	SACK	0.1	0.66	0.69
5	12000	SACK	0.0	0.43	0.61
5	36000	SACK	0.5	0.99	0.99
5	36000	SACK	0.1	0.98	0.96
5	36000	SACK	0.0	0.52	0.96
15	12000	SACK	0.5	0.85	0.90
15	12000	SACK	0.1	0.61	0.76
15	12000	SACK	0.0	0.48	0.58
15	36000	SACK	0.5	0.95	0.97
15	36000	SACK	0.1	0.94	0.97
15	36000	SACK	0.0	0.72	0.95
5	12000	Reno	0.5	0.96	0.94
5	12000	Reno	0.1	0.79	0.71
5	12000	Reno	0.0	0.45	0.33
5	36000	Reno	0.5	0.97	0.93
5	36000	Reno	0.1	0.96	0.75
5	36000	Reno	0.0	0.92	0.33
15	12000	Reno	0.5	0.94	0.97
15	12000	Reno	0.1	0.66	0.79
15	12000	Reno	0.0	0.53	0.51
15	36000	Reno	0.5	0.97	0.98
15	36000	Reno	0.1	0.96	0.97
15	36000	Reno	0.0	0.66	0.59
5	12000	Vanilla	0.5	0.97	0.96
5	12000	Vanilla	0.1	0.70	0.69
5	12000	Vanilla	0.0	0.36	0.42
5	36000	Vanilla	0.5	0.97	0.97
5	36000	Vanilla	0.1	0.90	0.94
5	36000	Vanilla	0.0	0.33	0.92
15	12000	Vanilla	0.5	0.92	0.96
15	12000	Vanilla	0.1	0.66	0.74
15	12000	Vanilla	0.0	0.61	0.67
15	36000	Vanilla	0.5	0.97	0.97
15	36000	Vanilla	0.1	0.96	0.97
15	36000	Vanilla	0.0	0.93	0.93

Table 7: GEO: TCP with VBR (300ms on/off) over UBR+ with GR

TCP	Buffer	GR	Selective Drop	EPD
SACK	200000	0.5	0.87	0.84
SACK	200000	0.1	0.78	0.88
SACK	200000	0.0	0.74	0.82
SACK	600000	0.5	0.99	0.99
SACK	600000	0.1	0.99	0.99
SACK	600000	0.0	0.99	0.99
Reno	200000	0.5	0.33	0.46
Reno	200000	0.1	0.24	0.26
Reno	200000	0.0	0.16	0.17
Reno	600000	0.5	0.35	0.36
Reno	600000	0.1	0.39	0.34
Reno	600000	0.0	0.30	0.28
Vanilla	200000	0.5	0.83	0.71
Vanilla	200000	0.1	0.71	0.76
Vanilla	200000	0.0	0.81	0.68
Vanilla	600000	0.5	0.79	0.78
Vanilla	600000	0.1	0.80	0.80
Vanilla	600000	0.0	0.76	0.77

II. UBR switch drop policies and the minimum rate guarantee interaction with TCP congestion control algorithms.

II.A

"TCP Selective Acknowledgements and UBR Drop Policies to Improve ATM-UBR Performance over Terrestrial and Satellite Networks.

II.B

"Buffer Management for the GFR Service"

II.C

"Buffer Management for the GFR Service"

II.D

"GFR Implementation Options"

TCP Selective Acknowledgments and UBR Drop Policies to Improve ATM-UBR Performance over Terrestrial and Satellite Networks *

Rohit Goyal, Raj Jain, Shivkumar Kalyanaraman,
Sonia Fahmy, Bobby Vandalore, Sastri Kota†
Department of Computer Information Science
The Ohio State University
2015 Neil Avenue, DL 395
Columbus, OH 43210
E-mail: {*goyal,jain*}@cis.ohio-state.edu

Abstract

We study the performance of Selective Acknowledgments with TCP over the ATM-UBR service category. We examine various UBR drop policies, TCP mechanisms and network configurations to recommend optimal parameters for TCP over UBR. We discuss various TCP congestion control mechanisms compare their performance for LAN and WAN networks. We describe the effect of satellite delays on TCP performance over UBR and present simulation results for LAN, WAN and satellite networks. SACK TCP improves the performance of TCP over UBR, especially for large delay networks. Intelligent drop policies at the switches are an important factor for good performance in local area networks.

1 Introduction

The Unspecified Bit Rate (UBR) service in ATM networks does not have any explicit congestion control mechanisms [2]. In the simplest form of UBR, switches drop cells whenever their buffers overflow. As a result, TCP connections using ATM-UBR service with limited switch buffers experience low throughput [3, 4, 5, 9, 13]. In our previous paper [9] we analyzed several enhancements to the UBR drop policies, and showed that these enhancements can improve the performance of TCP over UBR. We also analyzed the performance of Reno TCP (TCP with fast retransmit and recovery) over UBR, and concluded that fast retransmit and recovery hurts the performance of TCP in the presence of congestion losses over wide area networks.

This paper discusses the performance of TCP with selective acknowledgments (SACK TCP) over the UBR service category. We compare the performance of SACK TCP with vanilla TCP (TCP with slow start) and Reno TCP (TCP with slow start and fast retransmit and recovery). Simulation results of the performance the SACK TCP with several UBR drop policies over terrestrial and satellite links are presented.

Section 2 describes the TCP congestion control mechanisms including the Selective Acknowledgments (SACK) option for TCP. Section 3 describes our implementation of SACK TCP and Section 4 analyzes the features and retransmission properties of SACK TCP. We also describe a change to TCP's fast retransmit and recovery, proposed in [18, 22] and named "New Reno" in [18]. Section 7 discusses some issues relevant to the performance of TCP over satellite networks. The remainder of the paper presents simulation results comparing the performance of various TCP congestion avoidance methods.

2 TCP Congestion Control

TCP's congestion control mechanisms are described in detail in [15, 21]. TCP uses a window based flow control policy. The variable RCVWND is used as a measure of the receiver's buffer capacity. When a destination TCP host receives a segment, it sends an acknowledgment (ACK) for the next expected segment. TCP congestion control is built on this window based flow control. The following subsections describe the various TCP congestion control policies.

2.1 Slow Start and Congestion Avoidance

The sender TCP maintains a variable called congestion window (CWND) to measure the network capacity. The number of unacknowledged packets in the network is limited to CWND or RCVWND whichever is lower. Initially, CWND is set to one segment and it increases by one segment on the receipt of each new ACK until it reaches a maximum (typically 65536 bytes). It

*Proceedings of ICCCN97, Las Vegas, NV, September 22-25, 1997

†Sastri Kota is with Lockheed Martin Telecommunications, Sunnyvale, CA. Email: sastri.kota@lmco.com

can be shown that in this way, CWND doubles every round trip time, and this corresponds to an exponential increase in the CWND every round trip time [15].

The sender maintains a retransmission timeout for the last unacknowledged packet. Congestion is indicated by the expiration of the retransmission timeout. When the timer expires, the sender saves half the CWND in a variable called SSTHRESH, and sets CWND to 1 segment. The sender then retransmits segments starting from the lost segment. CWND is increased by one segment on the receipt of each new ACK until it reaches SSTHRESH. This is called the slow start phase. After that, CWND increases by one segment every round trip time. This results in a linear increase of CWND every round trip time, and is called the congestion avoidance phase. Figure 1 shows the slow start and congestion avoidance phases for at typical TCP connection.

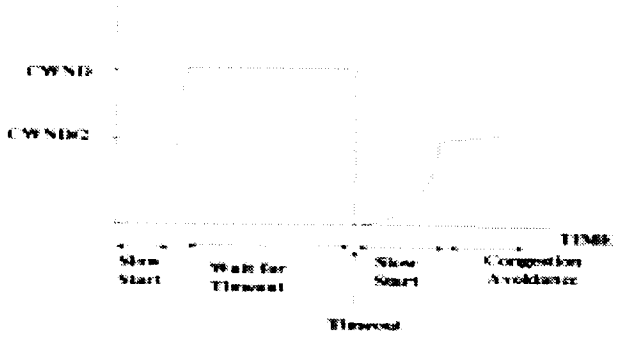


Figure 1: TCP Slow Start and Congestion Avoidance

2.2 Fast Retransmit and Recovery

Current TCP implementations use a coarse granularity (typically 500 ms) timer for the retransmission timeout. As a result, during congestion, the TCP connection can lose much time waiting for the timeout. In Figure 1, the horizontal CWND line shows the time lost in waiting for a timeout to occur. During this time, the TCP neither sends new packets nor retransmits lost packets. Moreover, once the timeout occurs, the CWND is set to 1 segment, and the connection takes several round trips to efficiently utilize the network. TCP Reno implements the fast retransmit and recovery algorithms that enable the connection to quickly recover from isolated segment losses [21].

If a segment is dropped by the network, the subsequent segments that arrive at the receiver are out-of-order segments. For each out-of-order segment, the TCP receiver immediately sends and ACK to the sender indicating the sequence number of the missing segment. This ACK is called a duplicate ACK. When the sender receives three duplicate ACKs, it concludes that the segment indicated by the ACKs has been lost, and immediately retransmits the lost segment. The sender then reduces CWND to half (plus 3 segments) and also saves half the original CWND value in SSTHRESH. Now for each subsequent duplicate ACK, the sender inflates CWND by one and tries to send a new segment. Effectively, the sender waits for half a round trip before sending one segment for each subsequent duplicate ACK it receives. As a result, the sender maintains the network pipe at half of its capacity at the time of fast retransmit.

Approximately one round trip after the missing segment is retransmitted, its ACK is received (assuming the retransmitted segment was not lost). At this time, instead of setting CWND to one segment and proceeding to do slow start until CWND reaches SSTHRESH, the TCP sets CWND to SSTHRESH, and then does congestion avoidance. This is called the fast recovery algorithm.

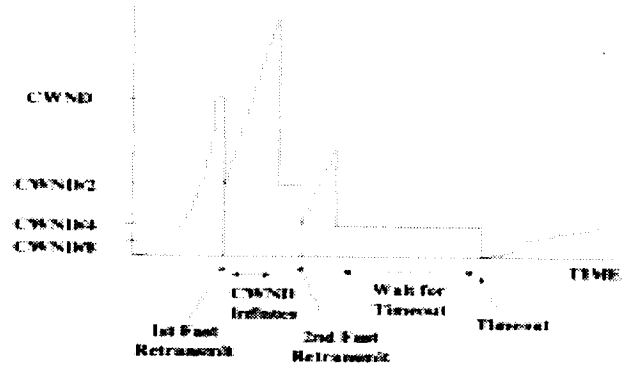


Figure 2: TCP Fast Retransmit and Recovery

2.3 A Modification to Fast Retransmit and Recovery: TCP New Reno

It has been known that fast retransmit and recovery cannot recover from multiple packet losses. Figure 2 shows a case when three consecutive packets are lost from a window, the sender TCP incurs fast retransmit twice and then times out. At that time, SSTHRESH is set to one-eighth of the original congestion window value (CWND in the figure). As a result, the exponential phase lasts a very short time, and the linear increase begins at a very small window. Thus, the TCP sends at a very low rate and loses much throughput.

The "fast-retransmit phase" was introduced in [22], in which the sender remembers the highest sequence number sent (RECOVER) when the fast retransmit was first triggered. After the first unacknowledged packet is retransmitted, the sender follows the usual fast recovery algorithm and inflates the CWND by one for each duplicate ACK it receives. When the sender receives an acknowledgment for the retransmitted packet, it checks if the ACK acknowledges all seg-

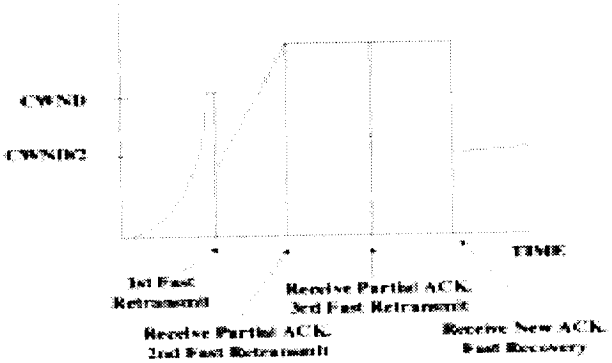


Figure 3: TCP with the fast retransmit phase

ments including RECOVER. If so, the ACK is a new ACK, and the sender exits the fast retransmit-recovery phase, sets its CWND to SSTHRESH and starts a linear increase. If on the other hand, the ACK is a partial ACK, i.e., it acknowledges the retransmitted segment, and only a part of the segments before RECOVER, then the sender immediately retransmits the next expected segment as indicated by the ACK. This continues until all segments including RECOVER are acknowledged. **This mechanism ensures that the sender will recover from N segment losses in N round trips.**

As a result, the sender can recover from multiple packet losses without having to timeout. In case of small propagation delays, and coarse timer granularities, this mechanism can effectively improve TCP throughput over vanilla TCP. Figure 3 shows the congestion window graph of a TCP connection for three contiguous segment losses. The TCP retransmits one segment every round trip time (shown by the CWND going down to 1 segment) until a new ACK is received.

2.4 Selective Acknowledgments

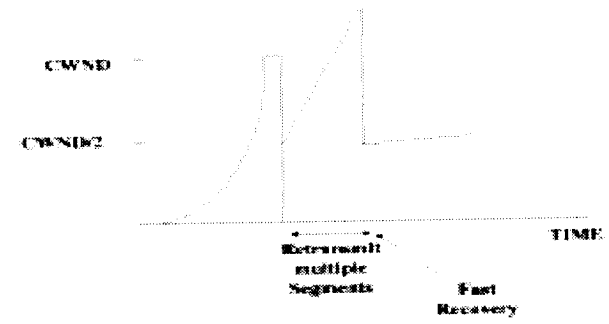


Figure 4: SACK TCP Recovery from packet loss

TCP with Selective Acknowledgments (SACK TCP) has been proposed to efficiently recover from multiple segment losses [20]. In SACK TCP, acknowledgments contain additional information about the segments that have been received by the destination. When the destination receives out-of-order segments, it sends duplicate ACKs (SACKs) acknowledging the out-of-order segments it has received. From these SACKs, the sending TCP can reconstruct information about the segments not received at the destination. When the sender receives three duplicate ACKs, it retransmits the first lost segment, and inflates its CWND by one for each duplicate ACK it receives. This behavior is the same as Reno TCP. However, when the sender is allowed to send a segment, it uses the SACK information to retransmit lost segments before sending new segments. As a result, the sender can recover from multiple dropped segments in about one round trip. Figure 4 shows the congestion window graph of a SACK TCP recovering from segment losses. During the time when the congestion window is inflating (after fast retransmit has incurred), the TCP is sending missing packets before any new packets.

3 SACK TCP Implementation

In this subsection, we describe our implementation of SACK TCP and some properties of SACK. Our implementation is based on the SACK implementation described in [18, 19, 20].

The SACK option is negotiated in the SYN segments during TCP connection establishment. The SACK information is sent with an ACK by the data receiver to the data sender to inform the sender of out-of-sequence segments received. The format of the SACK packet is described in [20]. The SACK option is sent whenever out of sequence data is received. All duplicate ACK's contain the SACK option. The option contains a list of some of the contiguous blocks of data already received by the receiver. Each data block is identified by the sequence number of the first byte in the block (the left edge of the block), and the sequence number of the byte immediately after the last byte of the block. Because of the limit on the maximum TCP header size, at most three SACK blocks can be specified in one SACK packet.

The receiver keeps track of all the out-of-sequence data blocks received. When the receiver generates a SACK, the first SACK block specifies the block of data formed by the most recently received data segment. This ensures that the receiver provides the most up-to-date information to the sender. After the first SACK block, the remaining blocks can be filled in any order.

The sender also keeps a table of all the segments sent but not ACKed. When a segment is sent, it is entered into the table. When the sender receives an ACK with the SACK option, it marks in the table all

the segments specified in the SACK option blocks as SACKed. The entries for each segment remain in the table until the segment is ACKed. The remaining behavior of the sender is very similar to Reno implementations with the modification suggested in Section 2.3 [1]. When the sender receives three duplicate ACKs, it retransmits the first unacknowledged packet. During the fast retransmit phase, when the sender is sending one segment for each duplicate ACK received, it first tries to retransmit the holes in the SACK blocks before sending any new segments. When the sender retransmits a segment, it marks the segment as retransmitted in the table. If a retransmitted segment is lost, the sender times out and performs slow start. When a timeout occurs, the sender resets the table.

During the fast retransmit phase, the sender maintains a variable PIPE that indicates how many bytes are currently in the network pipe. When the third duplicate ACK is received, PIPE is set to the value of CWND and CWND is reduced by half. For every subsequent duplicate ACK received, PIPE is decremented by one segment because the ACK denotes a packet leaving the pipe. The sender sends data (new or retransmitted) only when PIPE is less than CWND. This implementation is equivalent to inflating the CWND by one segment for every duplicate ACK and sending segments if the number of unacknowledged bytes is less than the congestion window value.

When a segment is sent, PIPE is incremented by one. When a partial ACK is received, PIPE is decremented by two. The first decrement is because the partial ACK represents a retransmitted segment leaving the pipe. The second decrement is done because the original segment that was lost, and had not been accounted for, is now actually considered to be lost.

4 TCP: Analysis of Recovery Behavior

In this section, we discuss the behavior of SACK TCP. We first analyze the properties of Reno TCP and then lead into the discussion of SACK TCP. Vanilla TCP without fast retransmit and recovery (we refer to TCP with only slow start and congestion avoidance as vanilla TCP), will be used as the basis for comparison. Every time congestion occurs, TCP tries to reduce its CWND window by half and then enters congestion avoidance. In the case of vanilla TCP, when a segment is lost, a timeout occurs, and the congestion window reduces to one segment. From there, it takes about $log_2(\text{CWND}/(2 \times \text{TCP segment size}))$ round trip times (RTTs) for CWND to reach the target value. This behavior is unaffected by the number of segments lost from a particular window.

[1] It is not clear to us whether the SACK option provides better performance with or without New Reno. This is under further study.

4.1 Reno TCP

When a single segment is lost from a window, Reno TCP recovers within approximately one RTT of knowing about the loss or two RTTs after the lost packet was first sent. The sender receives three duplicate ACKs about one RTT after the dropped packet was sent. It then retransmits the lost packet. For the next round trip, the sender receives duplicate ACKs for the whole window of packets sent after the lost packet. The sender waits for half the window and then transmits a half window worth of new packets. All of this takes about one RTT after which the sender receives a new ACK acknowledging the retransmitted packet and the entire window sent before the retransmission. CWND is set to half its original value and congestion avoidance is performed.

When multiple packets are dropped, Reno TCP cannot recover and may result in a timeout. The fast retransmit phase modification can recover from multiple packet losses by retransmitting a single packet every round trip time.

4.2 SACK TCP

In this subsection we show that SACK TCP can recover from multiple packet losses more efficiently than Reno or vanilla TCP.

Suppose that at the instant when the sender learns of the first packet loss (from three duplicate ACKs), the value of the congestion window is CWND. Thus, the sender has CWND bytes of data waiting to be acknowledged. Suppose also that the network drops a block of data which is CWND/n bytes long (This will typically result in several segments being lost). After one RTT of sending the first dropped segment, the sender receives three duplicate ACKs for this segment. It retransmits the segment, sets PIPE to CWND − 3, and sets CWND to CWND/2. For each duplicate ACK received, PIPE is decremented by 1. When PIPE reaches CWND, then for each subsequent duplicate ACK received, another segment can be sent. All the ACKs from the previous window take 1 RTT to return. For one half RTT nothing is sent (since PIPE > CWND). For the next half RTT, if CWND/n bytes were dropped, then only CWND/2 − CWND/n bytes (of retransmitted or new segments) can be sent. Thus, all the dropped segments can be retransmitted in 1 RTT if

$$\text{CWND}/2 - \text{CWND}/n \geq \text{CWND}/n$$

i.e., $n \geq 4$. Therefore, for SACK TCP to be able to retransmit all lost segments in one RTT, the network can drop at most CWND/4 bytes from a window of CWND.

Now, we calculate the maximum amount of data that can be dropped for SACK TCP to be able to retransmit everything in two RTTs. Suppose again that CWND/n bytes are dropped from a window of

size CWND. Then, in the first RTT from receiving the 3 duplicate ACKs, the sender can retransmit upto CWND/2 – CWND/n bytes. In the second RTT, the sender can retransmit 2(CWND/2 – CWND/n) bytes. This is because for each retransmitted segment in the first RTT, the sender receives a partial ACK that indicates that the next segment is missing. As a result, PIPE is decremented by 2, and the sender can send 2 more segments (both of which could be retransmitted segments) for each partial ACK it receives. Thus, all the dropped segments can be retransmitted in 2 RTTs if

$$\frac{CWND}{2} - \frac{CWND}{n} + 2(\frac{CWND}{2} - \frac{CWND}{n}) \geq \frac{CWND}{n}$$

i.e. $n \geq 8/3$. This means that at most 3×CWND/8 bytes can be dropped from a window of size CWND for SACK TCP to be able to recover in 2 RTTs.

Generalizing the above argument, we have the following result: **The number of RTTs needed by SACK TCP to recover from a loss of CWND/n is at most $\lceil \log(n/(n-2)) \rceil$ for $n > 2$.** If more than half the CWND is dropped, then there will not be enough duplicate ACKs for PIPE to become large enough to transmit any segments in the first RTT. Only the first dropped segment will be retransmitted on the receipt of the third duplicate ACK. In the second RTT, the ACK for the retransmitted packet will be received. This is a partial ACK and will result in PIPE being decremented by 2 so that 2 packets can be sent. As a result, PIPE will double every RTT, and **SACK will recover no slower than slow start [18, 19]**. SACK would still be advantageous because timeout would be still avoided unless a retransmitted packet were dropped.

5 The ATM-UBR Service

The basic UBR service can be enhanced by implementing intelligent drop policies at the switches. A comparative analysis of various drop policies on the performance of Vanilla and Reno TCP over UBR is presented in [9]. Section 5.3 briefly summarizes the results of our earlier work. This section briefly describes the drop policies.

5.1 Early Packet Discard

The Early Packet Discard policy [1] maintains a threshold R, in the switch buffer. When the buffer occupancy exceeds R, then all new incoming packets are dropped. Partially received packets are accepted if possible. It has been shown [9] that EPD improves the efficiency of TCP over UBR but does not improve fairness. The effect of EPD is less pronounced for large delay-bandwidth networks. In satellite networks, EPD has little or no effect in the performance of TCP over UBR.

5.2 Selective Packet Drop and Fair Buffer Allocation

These schemes use per-VC accounting to maintain the current buffer utilization of each UBR VC. A fair allocation is calculated for each VC, and if the VC's buffer occupancy exceeds its fair allocation, its subsequent incoming packet is dropped. Both schemes maintain a threshold R, as a fraction of the buffer capacity K. When the total buffer occupancy exceeds R×K, new packets are dropped depending on the VC_i's buffer occupancy (Y_i). In the Selective Drop scheme, a VC's *entire packet* is dropped if

$$(X > R) \text{ AND } (Y_i \times N_a / X > Z)$$

where N_a is the number of active VCs (VCs with at least one cell the buffer), and Z is another threshold parameter ($0 < Z \leq 1$) used to scale the effective drop threshold.

The Fair Buffer Allocation proposed in [8] is similar to Selective Drop and uses the following formula:

$$(X > R) \text{ AND } (Y_i \times N_a / X > Z \times ((K - R)/(X - R)))$$

5.3 Performance of TCP over UBR: Summary of Earlier Results

In our earlier work [9, 10] we discussed the following results:

- For multiple TCP connections, the switch requires a buffer size of the sum of the receiver windows of the TCP connections.
- With limited buffers, TCP over plain UBR results in poor performance.
- TCP performance over UBR can be improved by intelligent drop policies like Early Packet Discard, Selective Drop and Fair Buffer Allocation.
- TCP fast retransmit and recovery improves TCP performance over LANs, and actually degrades performance over WANs in the presence of congestion losses.

6 Simulation Results with SACK TCP over UBR

This section presents the simulation results of the various enhancements of TCP and UBR presented in the previous sections.

6.1 The Simulation Model

All simulations use the N source configuration shown in Figure 5. All sources are identical and persistent TCP sources i.e., the sources always send a segment as long as it is permitted by the TCP window. Moreover, traffic is unidirectional so that only the sources send data. The destinations only send ACKs. The performance of TCP over UBR with bidirectional traffic is a topic of further study. The delayed acknowledgment

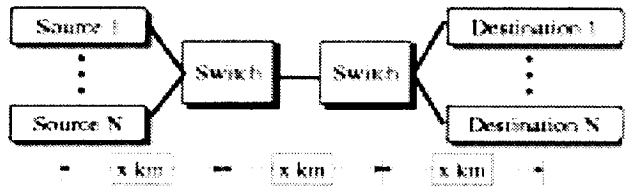


Figure 5: The N source TCP configuration

timer is deactivated, i.e., the receiver sends an ACK as soon as it receives a segment.

Link delays are 5 microseconds for LAN configurations and 5 milliseconds for WAN configurations. This results in a round trip propagation delay of 30 microseconds for LANs and 30 milliseconds for WANs respectively. The TCP segment size is set to 512 bytes. For the LAN configurations, the TCP maximum window size is limited by a receiver window of 64K bytes. This is the default value specified for TCP implementations. For WAN configurations, a window of 64K bytes is not sufficient to achieve 100% utilization. We therefore use the window scaling option to specify a maximum window size of 600,000 Bytes. This window is sufficient to provide full utilization with each TCP source.

All link bandwidths are 155.52 Mbps, and Peak Cell Rate at the ATM layer is 155.52 Mbps. The duration of the simulation is 10 seconds for LANs and 20 seconds for WANs. This allows enough round trips for the simulation to give stable results.

The configurations for satellite networks are discussed in Section 7.

6.2 Performance Metrics

The performance of the simulation is measured at the TCP layer by the Efficiency and Fairness as defined below.

$$\text{Efficiency} = \frac{\text{(Sum of TCP throughputs)}}{\text{(Maximum possible TCP throughput)}}$$

TCP throughput is measured at the destination TCP layer as the total number of bytes delivered to the application divided by the simulation time. This is divided by the maximum possible throughput attainable by TCP. With 512 bytes of TCP data in each segment, 20 bytes of TCP header, 20 bytes of IP header, 8 bytes of LLC header, and 8 bytes of AAL5 trailer are added. This results in a net possible throughput of 80.5% of the ATM layer data rate or 125.2 Mbps on a 155.52 Mbps link.

$$\text{Fairness Index} = (\Sigma x_i)^2 / (N \times \Sigma x_i^2)$$

Where x_i is the ratio of the achieved throughput to the expected throughput of the ith TCP source, and N is the number of TCP sources. Fairness values close to 0.99 indicate near perfect fairness.

6.3 Simulation Results

We performed simulations for the LAN and WAN configurations for three drop policies – vanilla UBR (switches drop incoming cells when their buffers overflow), Early Packet Discard (EPD) and Selective Drop. For LANs, we used buffer sizes of 1000 and 3000 cells. These are representative of the typical buffer sizes in current switches. For WANs, we chose buffer sizes of approximately one and three times the bandwidth – round trip delay product. Tables 1 and 2 show the efficiency and fairness values of SACK TCP with various UBR drop policies. Several observations can be made from these tables:

- For most cases, for a given drop policy, **SACK TCP provides higher efficiency than either the corresponding drop policy in vanilla or Reno TCP.** This confirms the intuition provided by the analysis of SACK that SACK recovers at least as fast as slow start when multiple packets are lost. In fact, for most cases, SACK recovers faster than both fast retransmit/recovery and slow start algorithms.
- **For LANs, the effect of drop policies is very important and can dominate the effect of SACK.** For UBR with tail drop, SACK provides a significant improvement over Vanilla and Reno TCPs. However, as the drop policies get more sophisticated, the effect of TCP congestion mechanism is less pronounced. This is because, the typical LAN switch buffer sizes are small compared to the default TCP maximum window of 64K bytes, and so buffer management becomes a very important factor. Moreover, the degraded performance of SACK in few cases can be attributed to excessive timeout due to the retransmitted packets being lost. In this case SACK loses several round trips in retransmitting parts of the lost data and then times out. After timeout, much of the data is transmitted again, and this results in wasted throughput. This result reinforces the need for a good drop policy for TCP over UBR.
- **The throughput improvement provided by SACK is more significant for wide area networks.** When propagation delay is large, a timeout results in the loss of a significant amount of time during slow start from a window of one segment. With Reno TCP (with fast retransmit and recovery), performance is further degraded (for multiple packet losses) because timeout occurs at a much lower window than vanilla TCP. With SACK TCP, a timeout is avoided at many times, and recovery is complete within a short number of roundtrips. Even if timeout occurs, the recovery

Table 1: SACK TCP over UBR : Efficiency

Configuration	Num of Srcs	Buffer (cells)	UBR	EPD	Sel Drop
LAN	5	1000	0.76	0.85	0.94
LAN	5	3000	0.98	0.97	0.98
LAN	15	1000	0.57	0.78	0.91
LAN	15	3000	0.86	0.94	0.97
SACK Column Average			0.79	0.89	0.95
Vanilla TCP Average			0.34	0.67	0.84
Reno TCP Average			0.69	0.97	0.97
WAN	5	12,000	0.90	0.88	0.95
WAN	5	36,000	0.97	0.99	1.00
WAN	15	12,000	0.93	0.80	0.88
WAN	15	36,000	0.95	0.95	0.98
SACK Column Average			0.94	0.91	0.95
Vanilla TCP Average			0.91	0.9	0.91
Reno TCP Average			0.78	0.86	0.81

Table 2: SACK TCP over UBR : Fairness

Configuration	Num of Srcs	Buffer (cells)	UBR	EPD	Sel Drop
LAN	5	1000	0.22	0.88	0.98
LAN	5	3000	0.92	0.97	0.96
LAN	15	1000	0.29	0.63	0.95
LAN	15	3000	0.74	0.88	0.98
SACK Column Average			0.54	0.84	0.97
Vanilla TCP Average			0.69	0.69	0.92
Reno TCP Average			0.71	0.98	0.99
WAN	5	12,000	0.96	0.98	0.95
WAN	5	36,000	1.00	0.94	0.99
WAN	15	12,000	0.99	0.99	0.99
WAN	15	36,000	0.98	0.98	0.96
Column Average			0.98	0.97	0.97
Vanilla TCP Average			0.76	0.95	0.94
Reno TCP Average			0.90	0.97	0.99

is as fast as slow start but a little time may be lost in the earlier retransmission.

- **The performance of SACK TCP can be improved by intelligent drop policies like EPD and Selective Drop**. This is consistent with our earlier results in [9]. Thus, we recommend that intelligent drop policies be used in UBR service.
- **The fairness values for selective drop are comparable to the values with the other TCP versions**. Thus, SACK TCP does not hurt the fairness in TCP connections with an intelligent drop policy like selective drop. The fairness of tail drop and EPD are sometimes a little lower for SACK TCP. This is again because retransmitted packets are lost and some connections timeout. Connections which do not timeout do not have to go through slow start, and thus can utilize more of the link capacity. The fairness among a set of hybrid TCP connections is a topic of further study.

7 Effects of Satellite Delays on TCP over UBR

Since TCP congestion control is inherently limited by the round trip time, long delay paths have significant effects on the performance of TCP over ATM. A large delay-bandwidth link must be utilized efficiently to be cost effective. This section discusses some of the issues that arise in the congestion control of large delay-bandwidth links. Simulation results of TCP over UBR with satellite delays are also presented. Related results in TCP performance over satellite are available in [23].

7.1 Window Scale Factor

The default TCP maximum window size is 65535 bytes. For a 155.52 Mbps ATM satellite link (with a propagation RTT of about 550 ms), a congestion window of about 8.7M bytes is needed to fill the whole pipe. As a result, the TCP window scale factor must be used to provide high link utilization. In our simulations, we use a receiver window of 34,000 and a window scale factor of 8 to achieve the desired window size.

7.2 Large Congestion Window and the congestion avoidance phase

During the congestion avoidance phase, CWND is incremented by 1 segment every RTT. Most TCP implementations follow the recommendations in [15], and increment by CWND by 1/CWND segments for each ACK received during the congestion avoidance. Since CWND is maintained in bytes, this increment translates to an increment of MSS×MSS/CWND bytes on the receipt of each new ACK. All operations are done on integers, and this expression avoids the need for floating point calculations. However, in the case of large delay-bandwidth paths where the window scale factor is used, MSS×MSS may be less than CWND. For example, with MSS = 512 bytes, MSS×MSS = 262144, and when CWND is larger than this value, the expression MSS×MSS/CWND yields zero. As a result, CWND is never increases during the congestion avoidance phase.

There are several solutions to this problem. The most intuitive is to use floating point calculations. This increases the processing overhead of the TCP layer and is thus undesirable. A second option is to not increment CWND for each ACK, but to wait for N ACKs such that N×MSS×MSS > CWND and then increment CWND by N×MSS×MSS/CWND. We call this the ACK counting option.

Another option would be to increase MSS to a larger

Table 3: TCP over UBR with Satellite Delays: Efficiency

TCP	Num of Srcs	Buffer (cells)	UBR	EPD	Sel Drop
SACK	5	200,000	0.86	0.6	0.72
SACK	5	600,000	0.99	1.00	1.00
Reno	5	200,000	0.84	0.12	0.12
Reno	5	600,000	0.30	0.19	0.22
Vanilla	5	200,000	0.70	0.73	0.73
Vanilla	5	600,000	0.88	0.81	0.82

Table 4: SACK TCP over UBR with Satellite Delays: Fairness

Configuration	Num of Srcs	Buffer (cells)	UBR	EPD	Sel Drop
SACK	5	200,000	1.00	0.83	0.94
SACK	5	600,000	1.00	1.00	1.00
Reno	5	200,000	0.96	0.97	0.97
Reno	5	600,000	1.00	1.00	1.00
Vanilla	5	200,000	1.00	0.87	0.89
Vanilla	5	600,000	1.00	1.00	1.00

value so that MSS×MSS would be larger than CWND at all times. The MSS size of the connection is limited by the smallest MTU of the connection. Most future TCPs are expected to use Path-MTU discovery to find out the largest possible MSS that can be used. This value of MSS may or may not be sufficient to ensure the correct functioning of congestion avoidance without ACK counting. Moreover, if TCP is running over a connectionless network layer like IP, the MTU may change during the lifetime of a connection and segments may be fragmented. In a cell based network like ATM, TCP could used arbitrary sized segments without worrying about fragmentation. The value of MSS can also have an effect on the TCP throughput, and larger MSS values can produce higher throughput. The effect of MSS on TCP over satellite is a topic of current research.

8 Simulation Results of TCP over UBR in Satellite networks

The satellite simulation model is very similar to the model described in section 6.1. The differences are listed below:

- The link between the two switches in Figure 5 is now a satellite link with a one-way propagation delay of 275 ms. The links between the TCP sources and the switches are 1 km long. This results in a round trip propagation delay of about 550 ms.
- The maximum value of the TCP receiver window is now 8,704,000 bytes. This window size is sufficient to fill the 155.52 Mbps pipe.
- The TCP maximum segment size is 9180 bytes. A larger value is used because most TCP connections over ATM with satellite delays are expected to use larger segment sizes.
- The buffer sizes used in the switch are 200,000 cells and 600,000 cells. These buffer sizes reflect buffers of about 1 RTT and 3 RTTs respectively.
- The duration of simulation is 40 seconds.

Tables 3 and 4 show the efficiency and fairness values for Satellite TCP over UBR with 5 TCP sources and buffer sizes of 200,000 and 600,000 cells. Several observations can be made from the tables:

- **Selective acknowledgments significantly improve the performance of TCP over UBR for satellite networks**. The efficiency and fairness values are typically higher for SACK than for Reno and vanilla TCP. This is because SACK often prevents the need for a timeout and can recover quickly from multiple packet losses.
- **Fast retransmit and recovery is detrimental to the performance of TCP over large delay-bandwidth links**. The efficiency numbers for Reno TCP in table 3 are much lower than those of either SACK or Vanilla TCP. This reinforces the WAN results in table 1 for Reno TCP. Both the tables are also consistent with analysis in Figure 2, and show that fast retransmit and recovery cannot recover from multiple losses in the same window.
- Intelligent drop policies have little effect on the performance of TCP over UBR satellite networks. Again, these results are consistent with the WAN results in tables 1 and 2. The effect of intelligent drop policies is most significant in LANs, and the effect decreases in WANs and satellite networks. This is because LAN buffer sizes (1000 to 3000 cells) are much smaller compared to the default TCP maximum window size of 65535 bytes. For WANs and satellite networks, the switch buffer sizes and the TCP maximum congestion window sizes are both of the order of the round trip delays. As a result, efficient buffer management becomes more important for LANs than WANs and satellite networks.

9 Summary

This paper describes the performance of SACK TCP over the ATM UBR service category. SACK TCP is seen to improve the performance of TCP over UBR. UBR drop policies are also essential to improving the performance of TCP over UBR. As a result, TCP performance over UBR can be improved by either improving TCP using selective acknowledgments, or by introducing intelligent buffer management policies at the switches. Efficient buffer management has a more sig-

nificant influence on LANs because of the limited buffer sizes in LAN switches compared to the TCP maximum window size. In WANs and satellite networks, the drop policies have a smaller impact because both the switch buffer sizes and the TCP windows are of the order of the bandwidth-delay product of the network. SACK TCP is especially helpful in satellite networks, and provides a large gain in performance over fast retransmit and recovery and slow start algorithms.

References

[1] Allyn Romanov, Sally Floyd, "Dynamics of TCP Traffic over ATM Networks," IEEE JSAC, May 1995.

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

[3] Chien Fang, Arthur Lin, "On TCP Performance of UBR with EPD and UB R-EPD with a Fair Buffer Allocation Scheme," ATM Forum 95-1645, December 1995.

[4] Hongqing Li, Kai-Yeung Siu, and Hong-Ti Tzeng, "TCP over ATM with ABR service versus UBR+EPD service," ATM Forum 95-0718, June 1995.

[5] H. Li, K.Y. Siu, H.T. Tzeng, C. Ikeda and H. Suzuki "TCP over ABR and UBR Services in ATM," Proc. IPCCC'96, March 1996.

[6] Hongqing Li, Kai-Yeung Siu, Hong-Yi Tzeng, Brian Hang, Wai Yang, " Issues in TCP over ATM," ATM Forum 95-0503, April 1995.

[7] J. Jaffe, "Bottleneck Flow Control," IEEE Transactions on Communications, Vol. COM-29, No. 7, pp. 954-962.

[8] Juha Heinanen, and Kalevi Kilkki, "A fair buffer allocation scheme," Unpublished Manuscript.

[9] R. Goyal, R. Jain, S. Kalyanaraman, S. Fahmy and Seong-Cheol Kim, "UBR+: Improving Performance of TCP over ATM-UBR Service," Proc. ICC'97, June 1997. [2]

[10] R. Goyal, R. Jain, S. Kalyanaraman and S. Fahmy, "Further Results on UBR+: Effect of Fast Retransmit and Recovery," ATM Forum 96-1761, December 1996.

[11] Shiv Kalyanaraman, Raj Jain, Sonia Fahmy, Rohit Goyal, Fang Lu and Saragur Srinidhi, "Performance of TCP/IP over ABR," Proc. IEEE Globecom'96, November 1996.

[12] Shivkumar Kalyanaraman, Raj Jain, Rohit Goyal, Sonia Fahmy and Seong-Cheol Kim, "Performance of TCP over ABR on ATM backbone and with various VBR background traffic patterns," Proc. ICC'97, June 1997.

[13] Stephen Keung, Kai-Yeung Siu, "Degradation in TCP Performance under Cell Loss," ATM Forum 94-0490, April 1994.

[14] Tim Dwight, "Guidelines for the Simulation of TCP/IP over ATM," ATM Forum 95-0077r1, March 1995.

[15] V. Jacobson, "Congestion Avoidance and Control," Proceedings of the SIGCOMM'88 Symposium, pp. 314-32, August 1988.

[16] V. Jacobson, R. Braden, "TCP Extensions for Long-Delay Paths," Internet RFC 1072, October 1988.

[17] V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance," Internet RFC 1323, May 1992.

[18] Kevin Fall, Sally Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," Computer Communications Review, July 1996

[19] Sally Floyd, "Issues of TCP with SACK," Lawrence Berkeley Labs, Technical report, December 1995

[20] M. Mathis, J. Madhavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgment Options," Internet RFC 2018, October 1996.

[21] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," Internet RFC 2001, January 1997.

[22] Janey C. Hoe, "Start-up Dynamics of TCP's Congestion Control and Avoidance Schemes," MS Thesis, Massachusetts Institute of Technology, June 1995.

[23] Mark Allman, Chris Hayes, Hans Kruse, Shawn Ostermann, "TCP Performance over Satellite Links," Proc. 5th International Conference on Telecommunications Systems, 1997.

[2] All our papers and ATM Forum contributions are available from http://www.cis.ohio-state.edu/~jain

****

ATM Forum Document Number: ATM_Forum/98-0405

****

TITLE: Buffer Management for the GFR Service

****

SOURCE: Rohit Goyal, Raj Jain, Sonia Fahmy, Bobby Vandalore
The Ohio State University,
Department of Computer and Information Science,
2015 Neil Ave, DL 395, Columbus, OH 43210-1277
Phone: 614-688-4482
{goyal,jain}@cis.ohio-state.edu

This work is partially sponsored by the NASA Glenn Research Center Under Contract Number NAS3-97198

****

DISTRIBUTION: ATM Forum Technical Committee
Traffic Management Working Group

****

DATE: July, 1998 (Portland)

****

ABSTRACT: In this contribution, we present a buffer management scheme called Differential Fair Buffer Allocation (DFBA) that provides MCR guarantees to GFR VCs carrying TCP/IP traffic. DFBA can be used on a FIFO buffer shared by several VCs. Each VC can carry traffic from one or more TCP connections. We discuss the features of DFBA and present simulation results to analyze its performance.

****

****

1 Introduction

The Guaranteed Frame Rate service has been designed to support non-real-time applications that can send data in the form of frames. IP routers separated by ATM clouds can benefit from the MCR guarantees provided by the GFR service. As a result, GFR implementations must be able to efficiently support MCR guarantees for TCP/IP traffic. These guarantees should be provided on a per-VC basis, where each GFR VC may contain traffic from several TCP connections.

While per-VC rate guarantees can be provided with per-VC queuing and scheduling, for most best effort traffic, it may be cost-effective to be able to provide minimum rate guarantees using a single queue. Intelligent buffer management techniques can be used to provide minimum rate guarantees. Such buffer management schemes must work with TCP traffic, and take into account the conservative slow start mechanism used by TCP on packet loss. Modern TCP implementations are expected to use Selective Acknowledgements (SACK) to minimize the occurrence of timeouts that trigger slow start. However, even with SACK, large losses due to severe congestion or very aggressive switch drop policies, can trigger timeouts. In addition to MCR guarantees, the GFR VCs should also be able to fairly share any excess capacity. As a result, the design of a good buffer management scheme for providing minimum rate guarantees to TCP/IP traffic is an important step towards the successful deployment of GFR.

In this contribution, we present the Differential Fair Buffer Allocation buffer management scheme. This buffer management work is an extension of our previous work on buffer management presented in [GOYALa]. The DFBA scheme presented here is an improved version of the scheme presented in [GOYALb]. We first overview some of the previous results on TCP over GFR. We then discuss the DFBA scheme, and present simulation results for this scheme. We conclude this contribution with a discussion of some key buffer management policies and their limitations.

2 Previous Results on TCP/IP over GFR

Several proposals have been made ([BASAK],[BONAVEN97],[GOYALa])to provide rate guarantees to TCP sources with FIFO queuing in the network. The bursty nature of TCP traffic makes it difficult to provide per-VC rate guarantees to TCP sources using FIFO queuing. Per-VC scheduling was recommended to provide rate guarantees to TCP connections. However, all these studies did not consider the impact of TCP dynamics, and used aggressive drop policies. We show that rate guarantees are achievable with a FIFO buffer using DFBA.

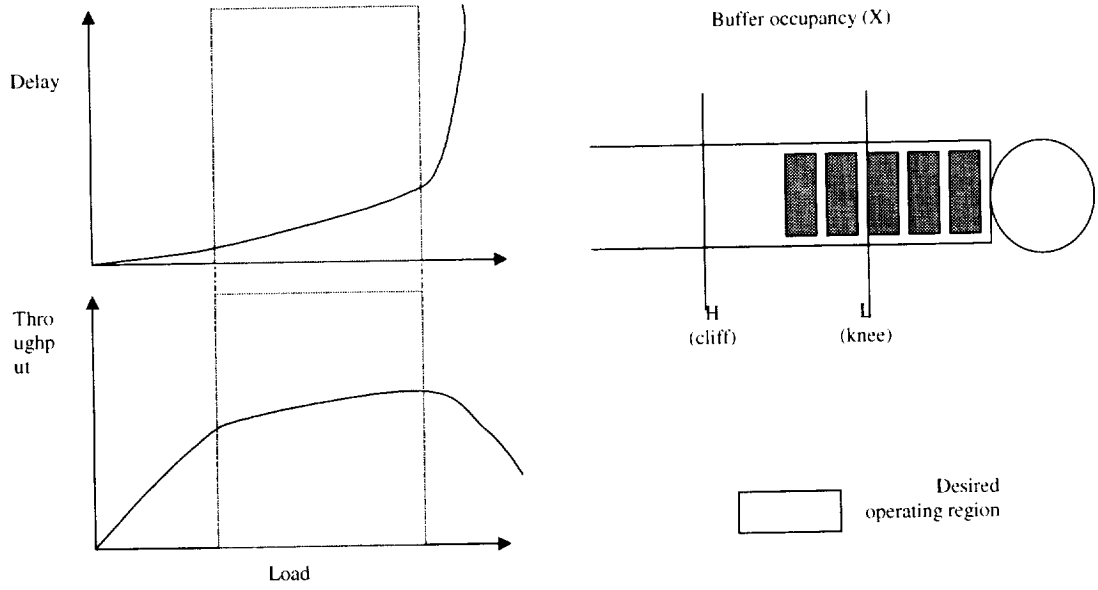
Many of the previous studies have examined TCP traffic with a single TCP connection over a VC. Per-VC buffer management for such cases, reduces to per-TCP buffer management. However, routers using GFR VCs would typically multiplex many TCP connections over a single VC. For VCs with several aggregated TCPs, per-VC control is unaware of each TCP in the VC. Moreover, aggregate TCP traffic characteristics and control requirements may be different from those of single TCP streams.

In [GOYALb], we have used FIFO buffers to control SACK TCP rates by buffer management using a preliminary version of DFBA. The scheme could allocate MCRs to TCP sources when the total MCR allocation was low (typically less than 50% of the GFR capacity). However, it was not clear how to allocate buffers based on the MCRs allocated to the respective VCs. Several other schemes have recently been presented for MCR

guarantees to GFR VCs carrying TCP traffic ([BONAVENb],[CHAO],[ELLOUMI]). In the following sections, we further describe the DFBA scheme, and discuss its design choices. We then present simulation results for both low and high MCR allocations using DFBA.

3 Differential Fair Buffer Allocation

DFBA uses the current queue length as an indicator of network load. The scheme tries to maintain an optimal load so that the network is efficiently utilized, yet not congested. The figure below illustrates the operating region for DFBA. The high threshold (H) and the low threshold (L) represent the cliff and the knee respectively of the load versus delay/throughput graph. The goal is to operate between the knee and the cliff. The scheme also assumes that the delay/throughput versus load curve behaves in a linear fashion between the knee and the cliff.

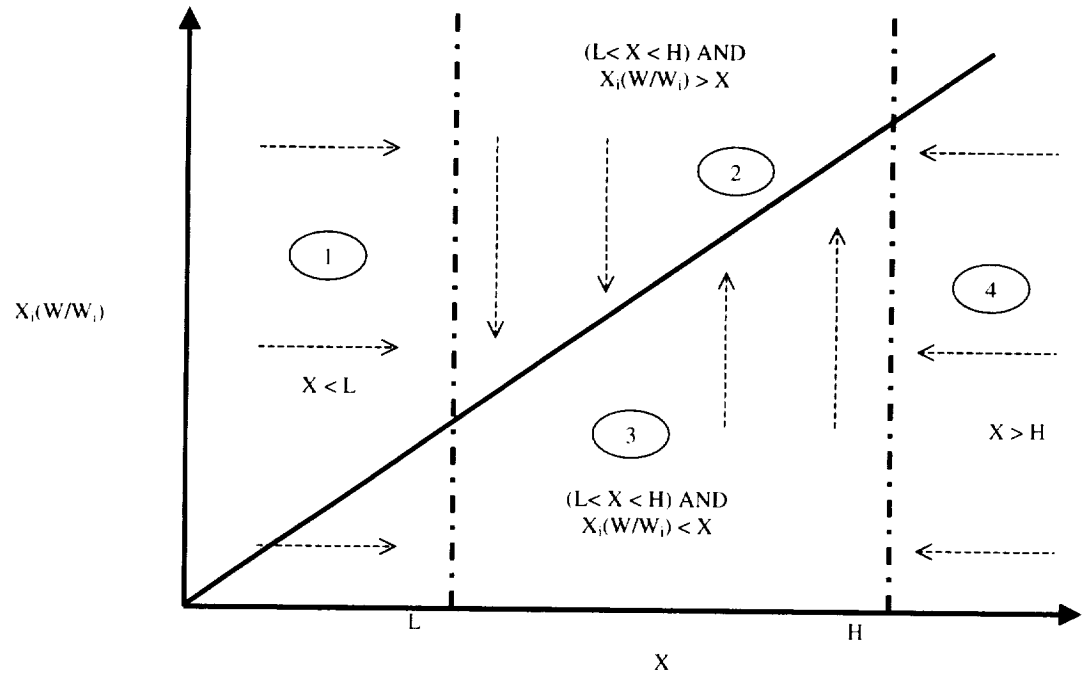


In addition to efficient network utilization, DFBA is designed to allocate buffer capacity fairly amongst competing VCs. This allocation is proportional to the MCRs of the respective VCs. The following variables are used by DFBA to fairly allocate buffer space:

- X = Total buffer occupancy at any time
- L = Low buffer threshold
- H = High buffer threshold
- MCR_i = MCR guaranteed to VC_i
- W_i = Weight of VC_i = MCR_i/(GFR capacity)
- $W = \Sigma W_i$
- X_i = Per-VC buffer occupancy ($X = \Sigma X_i$)
- Z_i = Parameter ($0 <= Z_i <= 1$)

DFBA tries to keep the total buffer occupancy (X) between L and H. When X falls below L, the scheme attempts to bring the system to efficient utilization by accepting all incoming packets. When X rises above H, the scheme tries to control congestion by performing EPD. When X is between L and H, DFBA attempts to allocate buffer space in proportional to the MCRs, as determined by the W_i for each VC. When X is between L

and H, the scheme also drops low priority (CLP=1) packets so as to ensure that sufficient buffer occupancy is available for CLP=0 packets.



The figure above illustrates the four operating regions of DFBA. The graph shows a plot of the current buffer occupancy X versus the normalized fair buffer occupancy for VC_i. If VC_i has a weight W_i, then its target buffer occupancy (X_i) should be $X*W_i/W$. Thus, the normalized buffer occupancy of VC_i is X_i*W/W_i. The goal is to keep this normalized occupancy as close to X as possible, as indicated by the solid line in the graph. Region 1 is the underload region, in which the current buffer occupancy is less than the low threshold L. In this case, the scheme tries to improve efficiency. Region 2 is the region with mild congestion because X is above L. As a result, any incoming packets with CLP=1 are dropped. Region 2 also indicates that VC_i has a larger buffer occupancy than its fair share (since $X_i > X*W_i/W$). As a result, in this region, the scheme drops some incoming CLP=0 packets of VC_i, as an indication to the VC that it is using more than its fair share. In region 3, there is mild congestion, but VC_i's buffer occupancy is below its fair share. As a result, only CLP=1 packets of a VC are dropped when the VC is in region 3. Finally, region 4 indicates severe congestion, and EPD is performed here.

In region 2, the packets of VC_i are dropped in a probabilistic manner. This drop behavior is controlled by the parameter Z_i, whose value depends on the connection characteristics. This is further discussed below. The figure below illustrates the drop conditions for DFBA.

The probability for dropping packets from a VC when it is in region 2 depends on several factors. The drop probability has two main components – the fairness component, and the efficiency component. Thus, P{drop} = fn(Fairness component, Efficiency component). The contribution of the fairness component increases as the VC's buffer occupancy X_i increases above its fair share. The contribution of the efficiency component increases as the total buffer occupancy increases above L. Since we assume that the system is linear between regions L and H, we choose to increase the drop probability linearly as X_i increases from $X*W_i/W$ to X, and as X increases from L to H. As a result, the drop probability is given by

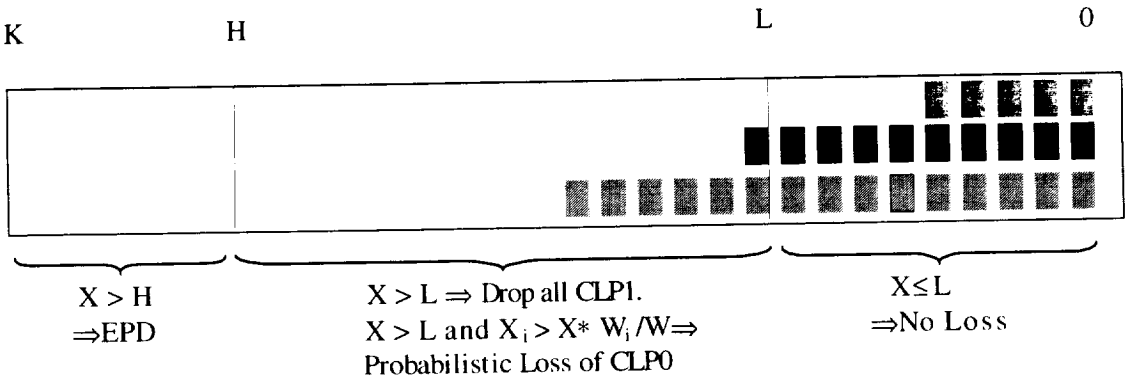
$$P\{drop\} = Z_i\left(\alpha\frac{X_i - X\times W_i/W}{X(1-W_i/W)} + (1-\alpha)\frac{X-L}{H-L}\right)$$

The parameter α is used to assign appropriate weights to the fairness and efficiency components of the drop probability. Z_i allows the scaling of the complete probability function based on per-VC characteristics.

It is well known that for a given TCP connection, a higher packet loss rate results in a lower average TCP window. As a result, a higher drop probability also results in a lower TCP window. In fact, it has been shown that for random packet loss, the average TCP window size is inversely proportional to the square root of the packet loss probability. As a result,

$$\text{TCP data rate} \propto \frac{\text{MSS}}{\text{RTT}\times\sqrt{\text{P\{drop\}}}}$$

This relationship can have a significant impact on TCP connections with either a high data rate or a large latency or both. To maintain high TCP data rate or when the RTT is large, one must choose a large TCP MSS, and/or must ensure that the average loss rate is low. As a result, DFBA can be tuned to choose a small Z_i for large latency VCs, as in the case of satellite VCs, or for VCs with high MCRs.



The following DFBA algorithm is executed when the first cell of a frame arrives at the buffer.

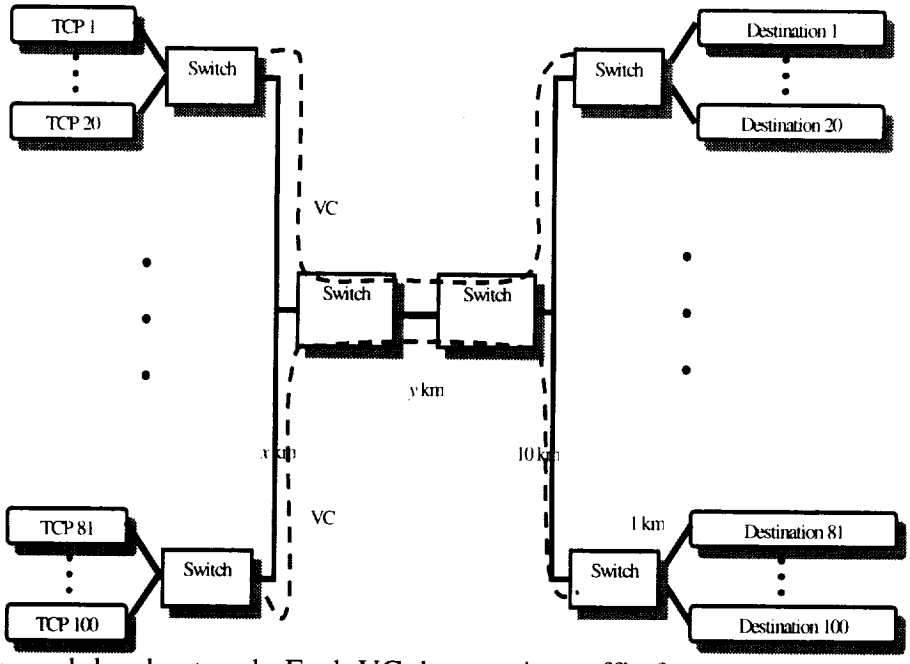
BEGIN
IF (X < L) THEN
Accept frame
ELSE IF (X > H) THEN
Drop frame
ELSE IF ((L < X < H) AND (X_i < X*W_i/W)) THEN
Drop CLP1 frame
ELSE IF ((L < X < H) AND (X_i > X*W_i/W)) THEN
Drop CLP1 frame
Drop CLP0 frame with

$$P\{drop\} = Z_i\left(\alpha\frac{X_i - X\times W_i/W}{X(1-W_i/W)} + (1-\alpha)\frac{X-L}{H-L}\right)$$

ENDIF
END

4 Simulation Configuration

We tested DFBA for ATM interconnected LANs with several scenarios. The following figure illustrates the basic test configuration. The figure shows 5 local switch pairs interconnected by two backbone switches that implement GFR. Each local switch carries traffic from multiple TCPs as shown in the figure. The backbone link carries 5 GFR VCs,



one from each local network. Each VC thus carries traffic from several TCP connections. The length of the local hop is denoted by x km, and the length of the backbone hop is denoted by y km. In this contribution, we present results with x=10 km and y=1000 km. The GFR capacity was fixed to the link rate of 155.52 Mbps (≈ 353,207 cells per sec). α was fixed to 0.5 in this study. All TCP sources were persistent TCPs with SACK. The SACK implementation is based on [FALL]. In our simulations, we varied four key parameters:

1. *Number of TCPs.* We used 10 TCPs per VC and 20 TCPs per VC for a total of 50 and 100 TCPs respectively.

2. *Per-VC MCR allocations.* Two sets of MCRs were chosen. In the first set, the MCR values were 12, 24, 36, 48 and 69 kcells/sec for VCs 1...5 respectively. This resulted in a total MCR allocation of about 50% of the GFR capacity. In the second set, the MCRs were 20, 40, 60, 80 and 100 kcells/sec for VCs 1...5 respectively, giving a total MCR allocation of 85% of the GFR capacity.

3. *Buffer size.* We first used a large buffer size of 25 kcells in the bottleneck backbone switch. We also analyzed DFBA performance with buffer sizes of 6 kcells, and 3 kcells.

4. Z_i. In most cases, the value of Z_i was chosen to be 1. We studied the effect of Z_i by decreasing it with increasing W_i.

5 Simulation Results

Table 1 shows achieved throughput for a 50 TCP configuration. The total MCR allocation is 50% of the GFR capacity. The W_i values for the VCs are 0.034, 0.068, 0.102, 0.136, and 0.170. The "achieved throughput" column shows the total end to end TCP throughput for all the TCP's over the respective VC. The table shows that the VCs achieve the guaranteed MCR. Although the VCs with larger MCRs get a larger share of the unused capacity, the last column of the table indicates that the excess bandwidth is however not shared in proportional to MCR. This is mainly because the drop probabilities are not scaled with respect to the MCRs, i.e., because $Z_i = 1$ for all i. The total efficiency (achieved throughput over maximum possible throughput) is close to 100%.

Table 1 50 TCPs, 5 VCs, 50% MCR Allocation

MCR	Achieved Throughput	Excess	Excess / MCR
4.61	11.86	7.25	1.57
9.22	18.63	9.42	1.02
13.82	24.80	10.98	0.79
18.43	32.99	14.56	0.79
23.04	38.60	15.56	0.68
69.12	126.88	57.77	

Table 2 illustrates the performance of DFBA when 85% of the GFR capacity is allocated as the MCR values. In this case, the W_i's are 0.057, 0.113, 0.17, 0.23, and 0.28 for VC's 1...5 respectively. The table again shows that DFBA meets the MCR guarantees for VCs carrying TCP/IP traffic.

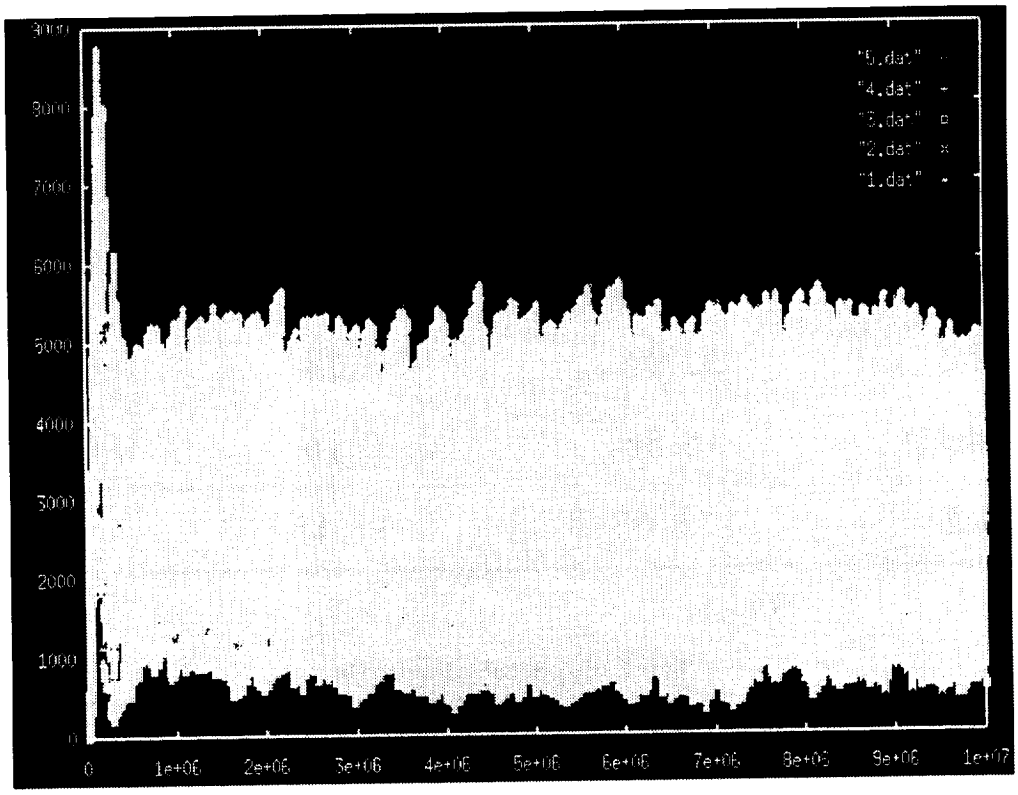
Table 2 50 TCPs, 5 VCs, 85% MCR Allocation

MCR	Achieved Throughput	Excess	Excess/MCR
7.68	12.52	4.84	0.63
15.36	18.29	2.93	0.19
23.04	25.57	2.53	0.11
30.72	31.78	1.06	0.03
38.40	38.72	0.32	0.01
115.2	126.88	11.68	

Table 3 validates the scheme for a larger number of TCPs. Each VC now carries traffic from 20 TCP connections, for a total of 100 TCPs. The total MCR allocation is 85% of the GFR capacity. All MCRs guarantees are met for a large number of TCPs and high MCR allocation.

Table 3 100 TCPs, 5 VCs, 85% MCR Allocation

MCR	Achieved Throughput	Excess	Excess/MCR
7.68	11.29	3.61	0.47
15.36	18.19	2.83	0.18
23.04	26.00	2.96	0.13
30.72	32.35	1.63	0.05
38.40	39.09	0.69	0.02
115.2	126.92	11.72	



The figure above illustrates the buffer occupancies of the 5 VCs in the bottleneck backbone switch. The figure shows that DFBA controls the switch buffer occupancy so that VCs with a lower MCR have a lower buffer occupancy than VCs with a higher MCR. In fact the average buffer occupancies are in proportion to the MCR values, so that FIFO scheduling can ensure a long-term MCR guarantee.

Table 4 and Table 5 show that DFBA provides MCR guarantees even when the bottleneck backbone switch has small buffers (6 kcells and 3 kcells respectively). The configuration uses 100 TCPs with 85% MCR allocation.

Table 4 Effect of Buffer Size (6 kcells)

MCR	Achieved Throughput	Excess	Excess/MCR
7.68	10.02	2.34	0.30
15.36	19.31	3.95	0.26
23.04	25.78	2.74	0.12
30.72	32.96	2.24	0.07
38.40	38.56	0.16	0.00
115.2	126.63	11.43	

Table 5 Effect of Buffer Size (3 kcells)

MCR	Achieved Throughput	Excess	Excess/MCR
7.68	11.79	4.11	0.54
15.36	18.55	3.19	0.21
23.04	25.13	2.09	0.09
30.72	32.23	1.51	0.05
38.40	38.97	0.57	0.01
115.2	126.67	11.47	

Table 6 shows the effect of Z_i on the fairness of the scheme in allocating excess bandwidth. We selected 2 values of Z_i based on the weights of the VCs. In the first experiment, Z_i was selected to be $(1-W_i/W)$ so that VCs with larger MCRs have a lower Z_i. N the second experiment, Z_i was selected to be $(1-W_i/W)^2$. The table shows that in the second case, sharing of the excess capacity is closely related to the MCRs of the VCs. An analytical assessment of the effect of Z_i on the excess capacity allocation by DFBA is a topic of further study.

Table 6 Effect of Z_i

$Z_i = 1-W_i/W$		$Z_i = (1-W_i/W)^2$	
Excess	Excess/ MCR	Excess	Excess/ MCR
3.84	0.50	0.53	0.07
2.90	0.19	2.97	0.19
2.27	0.10	2.77	0.12
2.56	0.08	2.39	0.08
0.02	0.02	3.14	0.08

6 A Framework for Buffer Management Schemes

Several recent papers have focused on fair buffer management schemes for TCP/IP traffic. All these proposals drop packets when the buffer occupancy exceeds a certain threshold. The proposals for buffer management can be classified into four groups based on whether they maintain multiple buffer occupancies (Multiple Accounting -- MA) or a single global buffer occupancy (Single Accounting -- SA), and whether they use multiple discard thresholds (Multiple Thresholds -- MT) or a single global discard Threshold (Single Threshold -- ST). The SA schemes maintain a single count of the number of cells currently in the buffer. The MA schemes classify the traffic into several classes and maintain a separate count for the number of cells in the buffer for each class. Typically, each class corresponds to a single connection, and these schemes maintain per-connection occupancies. In cases where the number of connections far exceeds the buffer size, the added over-head of per-connection accounting may be very expensive. In this case, a set of active connections is defined as those connections with at least one packet in the buffer, and only the buffer occupancies of active connections are maintained.

Schemes with a global threshold (ST) compare the buffer occupancy(s) with a single threshold and drop packets when the buffer occupancy exceeds the threshold. Multiple thresholds (MT) can be maintained corresponding to classes, connections, or to provide differentiated services. Several modifications to this drop behavior can be implemented. Some schemes like RED and FRED compare the average(s) of the buffer occupancy(s) to the threshold(s). Some like EPD maintain static threshold(s) while others like FBA maintain dynamic threshold(s). In some schemes, packet discard may be probabilistic (as in RED) while others drop packets deterministically (EPD/PPD). Finally, some schemes may differentiate packets based on packet tags. Examples of packet tags are the CLP bit in ATM cells or the TOS octet in the IP header of the IETF differentiated services architecture. Table 7 lists the four classes of buffer management schemes and examples of schemes for these classes. The example schemes are briefly discussed below.

The first SA-ST schemes included Early Packet Discard (EPD), Partial Packet Discard (PPD) [ROMANOV95] and Random Early Detection (RED) [FLOYD93]. EPD and PPD improve network efficiency because they minimize the transmission of partial packets by the network. Since they do not discriminate between connections in dropping packets, these schemes are unfair in allocating bandwidth to competing connections. For example when the buffer occupancy reaches the EPD threshold, the next incoming packet is dropped even if the packet belongs to a connection that is has received an unfair share of the bandwidth. Random Early Detection (RED) maintains a global threshold for the average queue. When the average queue exceeds this threshold, RED drops packets probabilistically using a uniform random variable as the drop probability. The basis for this is that uniform dropping will drop packets in proportion to the input rates of the connections. Connections with higher input rates will lose proportionally more packets than connections with lower input rates, thus maintaining equal rate allocation.

Table 7 Classification of Buffer Management Schemes

Group	Examples	Threshold Type (Static/Dynamic)	Drop Type (Deterministic/ Prbabilistic)	Tag/TOS Sensitive (Yes/No)
SAST	EPD, PPD	Static	Deterministic	No
	RED	Static	Probabilistic	No
MAST	FRED	Dynamic	Probabilistic	No
	Selective Drop, FBA, VQ+Dynamic EPD	Dynamic	Deterministic	No
MAMT	PME+ERED	Static	Probabilistic	Yes
	DFBA	Dynamic	Probabilistic	Yes
	VQ+MCR scheduling	Dynamic	Deterministic	No
SAMT	Priority Drop	Static	Deterministic	Yes

However, it has been shown in [LIN97] that proportional dropping cannot guarantee equal bandwidth sharing. The paper also contains a proposal for Flow Random Early Drop (FRED). FRED maintains per-connection buffer occupancies and drops packets probabilistically if the per-connection occupancy exceeds the average queue length. In addition, FRED ensures that each connection has at least a minimum number of packets in the queue. In this way, FRED ensures that each flow has roughly the same number of packets *in the buffer*, and FCFS scheduling guarantees equal sharing of bandwidth. FRED can be classified as one that maintains per-connection queue lengths, but has a global threshold (MA-ST).

The Selective Drop (SD) ([GOYAL98b]) and Fair Buffer Allocation (FBA) [HEIN] schemes are MA-ST schemes proposed for the ATM UBR service category. These schemes use per-connection accounting to maintain the current buffer utilization of each

UBR Virtual Channel (VC). A fair allocation is calculated for each VC, and if the VC's buffer occupancy exceeds its fair allocation, its subsequent incoming packet is dropped. Both schemes maintain a threshold R, as a fraction of the buffer capacity K. When the total buffer occupancy exceeds R*K, new packets are dropped depending on the VC_i's buffer occupancy (X_i). In these schemes, a VC's entire packet is dropped if

$$(X > R) \text{ AND } (X_i * N_a / X > Z) \text{ (Selective Drop)}$$

$$(X > R) \text{ AND } (X_i * N_a / X > Z * ((K - R)/(X - R))) \text{ (Fair Buffer Allocation)}$$

Where N_a is the number of active VCs (VCs with at least one cell the buffer), and Z is another threshold parameter ($0 < Z <= 1$) used to scale the effective drop threshold.

The Virtual Queuing (VQ) [SIU97] scheme is unique because it achieves fair buffer allocation by emulating on a single FIFO queue, a per-VC queued round-robin server. At each cell transmit time, a per-VC accounting variable (X'_i) is decremented in a round-robin manner, and is incremented whenever a cell of that VC is admitted in the buffer. When X'_i exceeds a fixed threshold, incoming packets of the *i*th VC are dropped. An enhancement called Dynamic EPD changes the above drop threshold to include only those sessions that are sending less than their fair shares. Since the above MA-ST schemes compare the per-connection queue lengths (or virtual variables with equal weights) with a global threshold, they can only guarantee equal buffer occupancy (and thus throughput) to the competing connections. These schemes do not allow for specifying a guaranteed rate for connections or groups of connections. Moreover, in their present forms, they cannot support packet priority based on tagging.

Another enhancement to VQ, called MCR scheduling [WU97], proposes the emulation of a weighted scheduler to provide Minimum Cell Rate (MCR) guarantees to ATM connections. In this scheme, a per-VC weighted variable W_i is maintained, and compared with a global threshold. A time interval T is selected, at the end of which, W_i is incremented by $MCR_i * T$ for each VC *i*. The remaining algorithm is similar to VQ. As a result of this weighted update, MCRs can be guaranteed. However, the implementation of this scheme involves the update of W_i for each VC after every time T. To provide tight MCR bounds, a smaller value of T must be chosen, and this increases the complexity of the scheme. For best effort traffic (like UBR), thousands of VC could be sharing the buffer, and this dependence on the number of VCs man not be an efficient solution to the buffer management problem. Since the variable W_i is updated differently for each VC *i*, this is equivalent to having different thresholds for each VC at the start of the interval. These thresholds are then updated in the opposite direction of W_i. As a result, VQ+MCR scheduling can be classified as a MA-MT scheme. The Differential Fair Buffer Allocation Scheme discussed in this contribution is a MA-MT scheme as shown in Table 7.

[FENG] proposes a combination of a Packet Marking Engine (PME) and an Enhanced RED scheme based on per-connection accounting and multiple thresholds (MA-MT). PME+ERED is designed for the IETF differentiated services architecture, and can provide loose rate guarantees to connections. The PME measures per-connection bandwidths and probabilistically marks packets if the measured bandwidths are lower than the target bandwidths (multiple thresholds). High priority packets are marked, and low priority packets are unmarked. The ERED mechanism is similar to RED except that the probability of discarding marked packets is lower than that of discarding unmarked packets. The PME in a node calculates the observed bandwidth over an update interval by counting the number of accepted packets of each connection by the node. Calculating bandwidth can be complex since it may require averaging over several time intervals.

Although it has not been formally proven, Enhanced RED may suffer from the same problem as RED because it does not consider the number of packets actually in the queue.

A simple SA-MT scheme can be designed that implements multiple thresholds based on the packet priorities. When the global queue length (single accounting) exceeds the first threshold, packets tagged as lowest priority are dropped. When the queue length exceeds the next threshold, packets from the lowest and the next priority are dropped. This process continues until EPD/PPD is performed on all packets. The performance of such schemes needs to be analyzed. However, these schemes cannot provide per-connection throughput guarantees and suffer from the same problem as EPD, because they do not differentiate between overloading and underloading connections.

Table 8 illustrates the fairness properties of the four buffer management groups presented above.

Table 8 Properties of Buffer Management Schemes

Group	Equal bandwidth allocation	Weighted bandwidth allocation
SA-ST	6.1.1.1.1 No	No
MA-ST	Yes	No
MA-MT	Yes	Yes
SA-MT	-	-

7 References

[BASAK] Debashis Basak, Surya Pappu, "GFR Implementation Alternatives with Fair Buffer Allocation Schemes," ATM Forum 97-0528.

[BONAVEN97] Olivier Bonaventure, "A simulation study of TCP with the proposed GFR service category," DAGSTUHL Seminar 9725, High Performance Networks for Multimedia Applications, June 1997.

[BONAVENb] Olivier Bonaventure, "Providing bandwidth guarantees to internetwork traffic in ATM networks," Proceedings of ATM'98, May 1998.

[CHAO] Dapeng Wu, H. J. Chao, "TCP/IP over ATM-GFR," Proceedings of ATM'98, May 1998.

[ELLOUMI] Omar Elloumi, Hossam Afifi, "Evaluation of FIFO based Buffer Management Algorithms for TCP over Guaranteed Frame Rate Service," Proceedings of ATM'98, May 1998.

[FALL] Kevin Fall, Sally Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," Computer Communications Review, July 1996

[FENG] Wu-chang Feng, Dilip Kandlur, Debanjan Saha, Kang G. Shin, "Techniques for Eliminating Packet Loss in Congested TCP/IP Networks," _____.

[FLOYD93] Sally Floyd, Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transaction on Networking*, August 1993.

[GOYAL98b] Rohit Goyal, Raj Jain, et. al., "Improving the Performance of TCP over the ATM-UBR Service," To appear in *Computer Communications*, 1998.

[GOYALa] Rohit Goyal, Raj Jain et. al., "Design Issues for providing minimum rate guarantees to the ATM Unspecified Bit Rate Service," Proceedings of ATM'98, May 1998.

[GOYALb] Rohit Goyal, Raj Jain et. al, "Providing Rate Guarantees to TCP over the ATM GFR Service," To appear in the Proceedings of LCN'98, November 1998.

[HEIN] Juha Heinanen, Kalevi Kilkki, "A Fair Buffer Allocation Scheme," Unpublished Manuscript.

[LIN97] Dong Lin, Robert Morris, "Dynamics of Random Early Detection," Proceedings of SIGCOMM97, 1997.

[ROMANOV95] Allyn Romanov, Sally Floyd, "Dynamics of TCP Traffic over ATM Networks," *IEEE Journal of Selected Areas In Telecommunications*, May 1995.

[SIU97] Kai-Yeung Siu, Yuan Wu, Wenge Ren, "Virtual Queuing Techniques for UBR+ Service in ATM with Fair Access and Minimum Bandwidth Guarantee," Proceedings of Globecom'97, 1997.

[WU97] Yuan Wu, Kai-Yeung Siu, Wenge Ren, "Improved Virtual Queuing and Dynamic EPD Techniques for TCP over ATM," Proceedings of ICNP97, 1997.

Buffer Management for TCP over the ATM GFR Service[‡]

Rohit Goyal, Raj Jain, Sonia Fahmy, Bobby Vandalore, Mukul Goyal
Department of Computer and Information Science, The Ohio State University
395 Dreese Lab, 2015 Neil Avenue
Columbus, OH 43210-1277, U.S.A.
E-mail: goyal@cis.ohio-state.edu

Abstract:
In current internetworking architectures, enterprise networks can be interconnected to each other, or to their service providers via backbone ATM VCs. Most ATM networks provide best effort UBR connections for TCP/IP traffic. The ATM Guaranteed Frame Rate (GFR) service is a best effort service that provides minimum rate guarantees to ATM VCs. Edge devices connecting IP LANs to an ATM network can use GFR VCs to transport TCP/IP traffic. Buffer management techniques are essential to the realization of a robust GFR implementation. In this paper we show how rate guarantees can be provided to VCs carrying TCP traffic, using buffer management on a FIFO buffer. We present a buffer management scheme called Differential Fair Buffer Allocation (DFBA) that provides Minimum Cell Rate (MCR) guarantees to ATM VCs carrying TCP/IP traffic. DFBA allocates buffer space in proportion to MCR and probabilistically drops TCP packets to control congestion and maintain MCR. DFBA can be used on a FIFO buffer shared by several VCs. Each VC can carry traffic from one or more TCP connections. We discuss a framework for existing buffer management schemes and briefly describe their properties.

Keywords: ATM, TCP, GFR, buffer management

[‡]This research was supported in part by NASA Glenn Research Center under contract number NAS3-97198.
[†]This is a much enhanced version of the paper presented in LCN'98.

1 Introduction: The Guaranteed Frame Rate Service

Guaranteed Frame Rate (GFR) has been recently proposed in the ATM Forum as an enhancement to the UBR service category. Guaranteed Frame Rate provides a minimum rate guarantee to VCs at the frame level. The GFR service also allows for the fair usage of any extra network bandwidth. GFR requires minimum signaling and connection management functions, and depends on the network's ability to provide a minimum rate to each VC. GFR is likely to be used by applications that can neither specify the traffic parameters needed for a VBR VC, nor have cability for ABR (for rate based feedback control). Current internetworking applications fall into this category, and are not designed to run over QoS based networks. These applications could benefit from a minimum rate guarantee by the network, along with an opportunity to fairly use any additional bandwidth left over from higher priority connections. In the case of LANs connected by ATM backbones, network elements outside the ATM network could also benefit from GFR guarantees. For example, IP routers separated by an ATM network could use GFR VCs to exchange control messages. Figure 1 illustrates such a case where the ATM cloud connects several LANs and routers. ATM end systems may also establish GFR VCs for connections that can benefit from a minimum throughput guarantee.

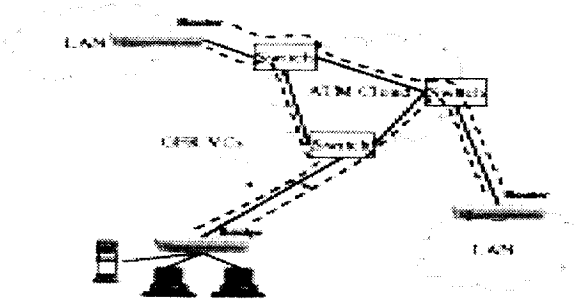


Figure 1: Use of GFR in ATM connected LANs

The original GFR proposals [11, 12] give the basic definition of the GFR service. GFR provides a minimum rate guarantee to the **frames** of a VC. GFR uses AAL5 with enables frame boundaries to be visible at the ATM layer. The service requires the specification of a maximum frame size

(MFS) of the VC. If the user sends packets (or frames) smaller than the maximum frame size, at a rate less than the minimum cell rate (MCR), then all the packets are expected to be delivered by the network with minimum loss. If the user sends packets at a rate higher than the MCR, it should still receive at least the minimum rate. The minimum rate is guaranteed to the CLP=0 frames of the connection. In addition, a connection sending in excess of the minimum rate should receive a fair share of any unused network capacity. The exact specification of the fair share has been left unspecified by the ATM Forum. The detailed GFR specification is provided in [1], but the above discussion captures the essence of the service.

There are three basic design options that can be used by the *network* to provide the per-VC minimum rate guarantees for GFR - tagging, buffer management, and queueing:

1. **Tagging:** *Network based tagging* (or policing) can be used as a means of marking non-conforming packets before they enter the network. This form of tagging is usually performed when the connection enters the network. Figure 2 shows the role of network based tagging in providing a minimum rate service in a network. Network based tagging on a per-VC level requires some per-VC state information to be maintained by the network and increases the complexity of the network element. Tagging can isolate conforming and non-conforming traffic of each VC so that other rate enforcing mechanisms can use this information to schedule the conforming traffic in preference to non-conforming traffic. In a more general sense, policing can be used to discard non-conforming packets, thus allowing only conforming packets to enter the network.

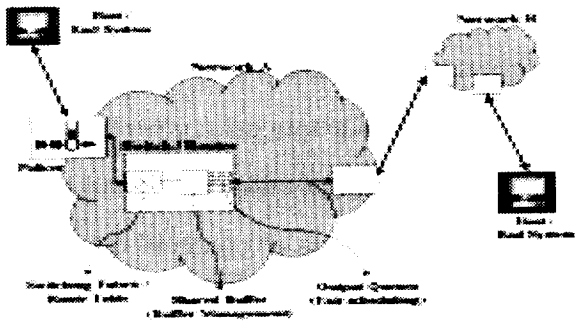


Figure 2: Network Architecture with tagging, buffer management and scheduling

2. **Buffer management:** Buffer management is typically performed by a network element (like a switch or a router) to control the number of packets entering its buffers. In a shared buffer environment, where multiple VCs share common buffer space, per-VC buffer management can control the buffer occupancies of individual VCs. Per-VC buffer management uses per-VC accounting to keep track of the buffer occupancies of each VC. Figure 2 shows the role of buffer management in the connection path. Examples of per-VC buffer management schemes are Selective Drop and Fair Buffer Allocation [9]. Per-VC accounting introduces overhead, but without per-VC accounting it is difficult to control the buffer occupancies of individual VCs (unless non-conforming packets are dropped at the entrance to the network by the policer). Note that per-VC buffer management uses a single FIFO queue for all the VCs. This is different from per-VC queuing and scheduling discussed below.

3. **Scheduling:** Figure 2 illustrates the position of scheduling in providing rate guarantees. While tagging and buffer management control the entry of packets into a network element, queuing strategies determine how packets are scheduled onto the next hop. FIFO queuing cannot isolate packets from various VCs at the egress of the queue. As a result, in a FIFO queue, packets are scheduled in the order in which they enter the buffer. Per-VC queuing, on the other hand, maintains a separate queue for each VC in the buffer. A scheduling mechanism can select between the queues at each scheduling time. However, scheduling adds the cost of per-VC queuing and the service discipline. For a simple service like GFR, this additional cost may be undesirable.

A desirable implementation of GFR is to use a single queue for all GFR VCs, and provide minimum rate guarantees by means of intelligent buffer management policies on the FIFO. Several proposals have been made [3, 4, 8] to provide rate guarantees to TCP sources with FIFO queuing in the network. The bursty nature of TCP traffic makes it difficult to provide per-VC rate guarantees using FIFO queuing. Per-VC scheduling was recommended to provide rate guarantees to TCP connections. However, all these studies were performed at high target network utilization, i.e., most of the network capacity was allocated to the MCRs. Moreover, these proposals are very

aggressive in dropping TCP packets causing TCP to timeout and lose throughput.

All the previous studies have examined TCP traffic with a single TCP per VC. Per-VC buffer management for such cases reduces to per-TCP buffer management. However, routers that use GFR VCs, will multiplex many TCP connections over a single VC. For VCs with several aggregated TCPs, per-VC control is unaware of each TCP in the VC. Moreover, aggregate TCP traffic characteristics and control requirements may be different from those of single TCP streams.

In this paper, we present:

- Issues in providing minimum rate guarantees to TCP traffic with FIFO buffers.
- A buffer management scheme for MCR guarantees to VCs with aggregate TCP flows.

We begin in section 2 by disussing some introductory material on the behavior of TCP traffic with controlled windows. This provides insight into controlling TCP rates by controlling TCP windows. Section 3 describes the effect of buffer occupancy and thresholds on TCP throughput. The focus of these sections is to present empirical simulation based analysis of intuitive ideas on controlling TCP rates using buffer management. Section 4 presents a dynamic threshold-based buffer management policy called DFBA to provide TCP throughputs in proportion to buffer thresholds for low rate allocations. This scheme assumes that each VC may carry multiple TCP connections. We then present simulation results with TCP traffic over LANs interconnected by an ATM network. We conclude with a framework for buffer management, and briefly describe how some key buffer management schemes fit into this framework.

2 TCP Behavior with Controlled Windows

TCP uses a window based mechanism for flow control. The amount of data sent by a TCP connection in one round trip is determined by the window size of the TCP connection. The window size is the minimum of the sender's congestion window (CWND) and the receiver's window (RCVWND). TCP rate can be controlled by controlling the window size of the TCP connection.

However, a window limit is not enforceable by the network to control the TCP rate. The only form of control available to the network is to drop TCP packets. TCP sources respond to packet

loss by reducing the source congestion window by one-half, and then increasing it by one segment size every round trip. As a result, the average TCP window can be controlled by intelligent packet discard.

For TCP window based flow control, the throughput (in Mbps) can be calculated from the average congestion window (in Bytes) and the round trip time (in seconds) as:

$$\text{Throughput (Mbps)} = \frac{8 \times 10^{-6} \times \text{CWND}_{\text{avg}}}{\text{Round Trip Time}} \tag{1}$$

Where CWND_{avg} is the average congestion window in bytes, and Round Trip Time is in seconds. The factor 8×10^{-6} converts the throughput from bytes per sec to Megabits per sec.

Suppose the network capacity allows the TCP window to increase to CWND_{max} at which point TCP detects a packet loss, and reduces its window to $\text{CWND}_{\text{max}}/2$. The window then increases linearly to CWND_{max} when a packet is dropped again. The average TCP CWND during the linear increase phase can be calculated as:

$$\text{CWND}_{\text{avg}} = \frac{\Sigma_{i=1}^{T} \text{CWND}_{\text{max}}/2 + \text{MSS} \times i}{\text{T}} \tag{2}$$

where T is the number of round trip times for the congestion window to increase from $\text{CWND}_{\text{max}}/2$ to CWND_{max}. Note that this equation assumes that during the linear increase phase, the TCP window increases by one segment every round trip time. However, when the TCP delayed acknowledgment option is set, TCP might only send an ACK for every two segments. In this case, the window would increase by 1 segment every 2 RTTs.

Figure 3 shows how the source TCP congestion window varies when a single segment is lost at a particular value of the congestion window. The figure is the CWND plot of the simulation of the configuration shown in Figure 4 with a single SACK TCP source (N=1). The figure shows four different values of the window at which a packet is lost. The round trip latency (RTT) for the connection is 30 ms. The window scale factor is used to allow the TCP window to increase beyond the 64K limit.

From Figure 3 and equation 2, the average congestion windows in the linear phases of the four experiments are approximately 91232 bytes, 181952 bytes, 363392 bytes and over 600000 bytes. As

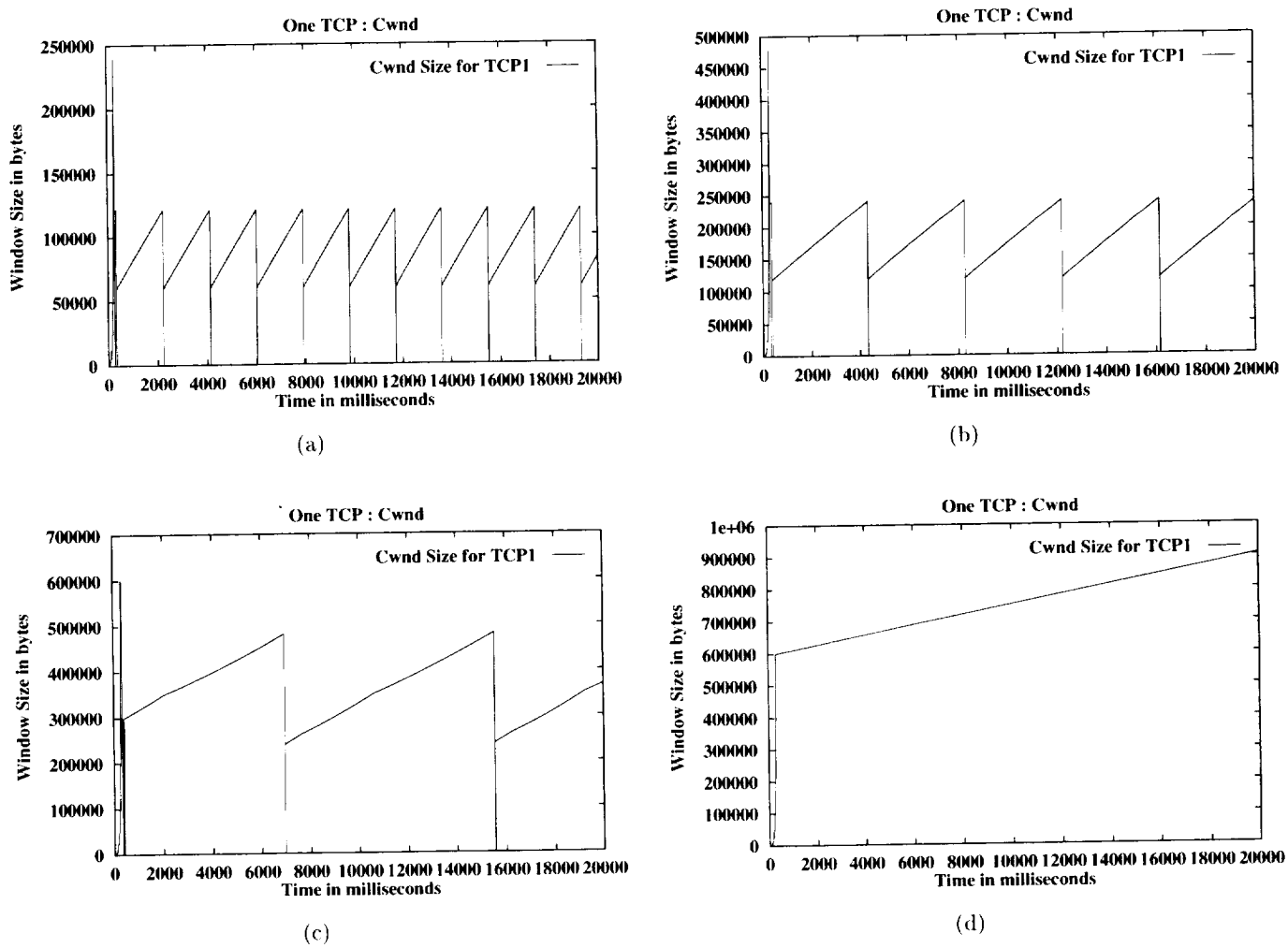


Figure 3: Single TCP Congestion Window Control. Drop thresholds (bytes) = 125000, 250000, 500000, None

a result, the average calculated throughputs from equation 1 are 24.32 Mbps, 48.5 Mbps, 96.9 Mbps, and 125.6 Mbps (126 Mbps is the maximum possible TCP throughput for a 155.52 Mbps link with 1024 byte TCP segments). The empirical TCP throughputs obtained from the simulations of the four cases are 23.64 Mbps, 47.53 Mbps, 93.77 Mbps and 125.5 Mbps respectively. The throughput values calculated from equation 2 are very close to those obtained by simulation. This shows that controlling the TCP window so as to maintain a desired average window size enables the network to control the average TCP throughput.

3 TCP Rate Control using Buffer Management

In the previous section, an artificial simulation was presented where the network controlled the TCP rate by dropping a packet every time the TCP window reached a particular value. In practice, the network knows neither the size of the TCP window, nor the round trip time of the connection. In this section, a switch can use per-VC accounting in its buffer to estimate the bandwidth used by the connection relative to other connections in the same buffer.

In a FIFO buffer, the average output rate of a connection is determined by the relative proportion of packets from the connection in the buffer. Let μ_i and x_i be the output rate and the buffer occupancy respectively of VC_i. Let μ and x be the total output rate and the buffer occupancy (total number of cells from all connections in the buffer) of the FIFO buffer respectively. Note that these numbers are averages over a long enough time period. Then, because the buffer is a FIFO,

$$\mu_i = \frac{x_i}{x}\mu$$

$$\text{or } \frac{x_i/x}{\mu_i/\mu} = 1$$

If the buffer occupancy of every active VC is maintained at a desired relative threshold, then the output rate of each VC can also be controlled. In other words, if a VC always has x_i cells in the buffer with a total occupancy of x cells, its average output rate will be at least $\mu x_i/x$.

Adaptive flows like TCP respond to segment loss by reducing their congestion window. A single packet loss is sufficient to reduce the TCP congestion window by one-half. Consider a drop policy that drops a single TCP packet from a connection every time the connection's buffer occupancy crosses a given threshold from below. The drop threshold for a connection, effectively determines the maximum size to which the congestion window is allowed to grow. Because of TCP's adaptive nature, the buffer occupancy reduces after about 1 RTT. The drop policy drops a single packet when the TCP's buffer occupancy crosses the threshold, and then allows the buffer occupancy to grow by accepting the remainder of the TCP window. On detecting a loss, TCP reduces its congestion window by 1 segment and remains idle for about one-half RTT, during which the buffer occupancy decreases below the threshold. Then the TCP window increases linearly (and so does the buffer occupancy), and a packet is again dropped when the buffer occupancy crosses the threshold. In

this way, TCP windows can be controlled quite accurately to within one round trip time. As a result, the TCP's throughput can also be controlled by controlling the TCP's buffer occupancy.

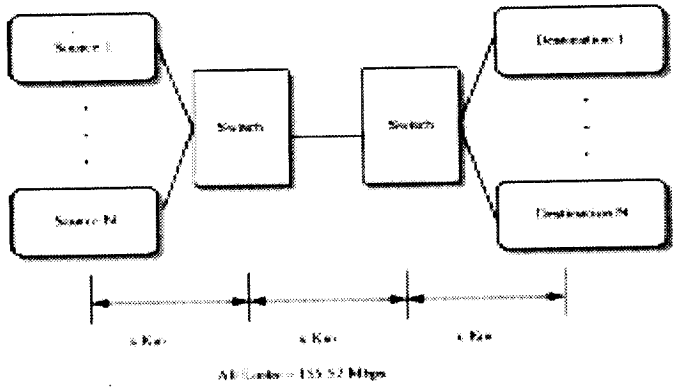


Figure 4: N source configuration

Table 1: Fifteen TCP buffer thresholds

Experiment #	1	2	3	4
TCP number	Threshold per TCP (cells) (r_i)			
1-3	305	458	611	764
4-6	611	917	1223	1528
7-9	917	1375	1834	2293
10-12	1223	1834	2446	3057
13-15	1528	2293	3057	3822
Tot. Thr (r)	13752	20631	27513	34392

Using this drop policy, we performed simulations of the TCP configuration in Figure 4 with fifteen TCP sources divided into 5 groups of 3 each. Each TCP source was a separate UBR VC. Five different buffer thresholds (r_i) were selected, and each of three TCP's in a group had the same buffer threshold. Table 1 lists the buffer thresholds for the VC's in the FIFO buffer of the switches. We performed experiments with 4 different sets of thresholds as shown by the threshold columns. The last row in the table shows the total buffer allocated ($r = \Sigma r_i$) to all the TCP connections for each simulation experiment. The total buffer size was large (48000 cells) so that there was enough space for the buffers to increase after the single packet drop. For a buffer size of 48000 cells, the total target buffer utilizations were 29%, 43%, 57%, 71% in the 4 columns of table 1 respectively. The selected buffer thresholds determine the MCR achieved by each connection. For each connection, the ratios of the thresholds to the total buffer allocation should be proportional to the ratios of the

Table 2: Fifteen TCP throughputs

Experiment #	1	2	3	4	
TCP number	Achieved throughput per TCP (Mbps) (μ_i)				Expected Throughput
					($\mu_i^e = \mu \times r_i/\Sigma r_i$)
1-3	2.78	2.83	2.95	3.06	2.8
4-6	5.45	5.52	5.75	5.74	5.6
7-9	8.21	8.22	8.48	8.68	8.4
10-12	10.95	10.89	10.98	9.69	11.2
13-15	14.34	13.51	13.51	13.93	14.0
Tot. throughput (μ)	125.21	122.97	125.04	123.35	126.0

achieved per-VC throughputs to the total achieved throughput. In other words, if μ_i, μ, r_i and r represent the per-VC achieved throughputs, total throughput, per-VC buffer thresholds, and total buffer threshold respectively, then we should have

$$\mu_i/\mu = r_i/r$$

or the expected per-VC throughput is

$$\mu_i^e = \mu \times r_i/r$$

The above formula holds when all TCPs are greedy, and are trying to use up their allocated thresholds by growing their congestion window. For non-greedy sources, or sources that may be bottlenecked elsewhere in the network the thresholds must be allocated relative to the current buffer occupancy and not statically as above. This is further discussed in section 4.1.

Table 2 shows the average throughput obtained per TCP in each group for each of the four simulations. The TCP throughputs were averaged over each group to reduce the effects of randomness. The last row of the table shows the total throughput obtained in each simulation. Based on the TCP segment size (1024 bytes) and the ATM overhead, it is clear that the TCPs were able to use almost the entire available link capacity (approximately 126 Mbps at the TCP layer).

The proportion of the buffer usable by each TCP (r_i/r) before the single packet drop should determine the proportion of the throughput achieved by the TCP. Table 3 shows the ratios (μ_i/μ_i^e) for each simulation. All ratios are close to 1. This indicates that the TCP throughputs are indeed proportional to the buffer allocations. The variations (not shown in the table) from the mean TCP

Table 3: Fifteen TCP buffer:throughput ratio

Experiment #	1	2	3	4
TCP number	Ratio (μ_i/μ_i^e)			
1-3	1.00	1.03	1.02	1.08
4-6	0.98	1.01	1.03	1.04
7-9	0.98	1.00	1.00	1.02
10-12	0.98	0.99	0.98	0.88
13-15	1.02	0.98	0.97	1.01

throughputs increased as the total buffer thresholds increased (from left to right across the table). This is because the TCPs suffered a higher packet loss due to the reduced room to grow beyond the threshold. Thus, high buffer utilization produced more variation in achieved rate (last column of Table 3), whereas in low utilization cases, the resulting throughputs were in proportion to the buffer allocations.

Figure 5 shows the congestion windows of one TCP from each group for each of the four simulations. The graphs illustrate that the behaviors of the TCP congestion windows are very periodic in these cases. The average throughput achieved by each TCP can be calculated from the graphs using equations 1 and 2. An intersting observation is that for each simulation, the slopes of the graphs during the linear increase are approximately the same for each TCP, i.e., for a given simulation, the rate of increase of CWND is the same for all TCPs regardless of their drop thresholds. We know that TCP windows increase by 1 segment every round trip time. We can conclude that for a given simulation, TCPs sharing the FIFO buffer experience similar queuing delays regardless of the individual per-connection thresholds at which their packets are dropped. This is because, if all TCP's buffer occupancies are close to their respective thresholds (r_i), then when a packet arrives at the buffer, it is queued behind cells from $\Sigma(r_i)$ packets, regardless of the connection to which it belongs. Consequently, each TCP experiences the same average queuing delay.

However, as the total buffer threshold increases (from experiment (a) to (d)), the round trip time for each TCP increases because of the larger total queue size. The larger threshold also results in a larger congestion window at which a packet is dropped. A larger congestion window means that TCP can send more segments in one round trip time. But, the round trip time also increases

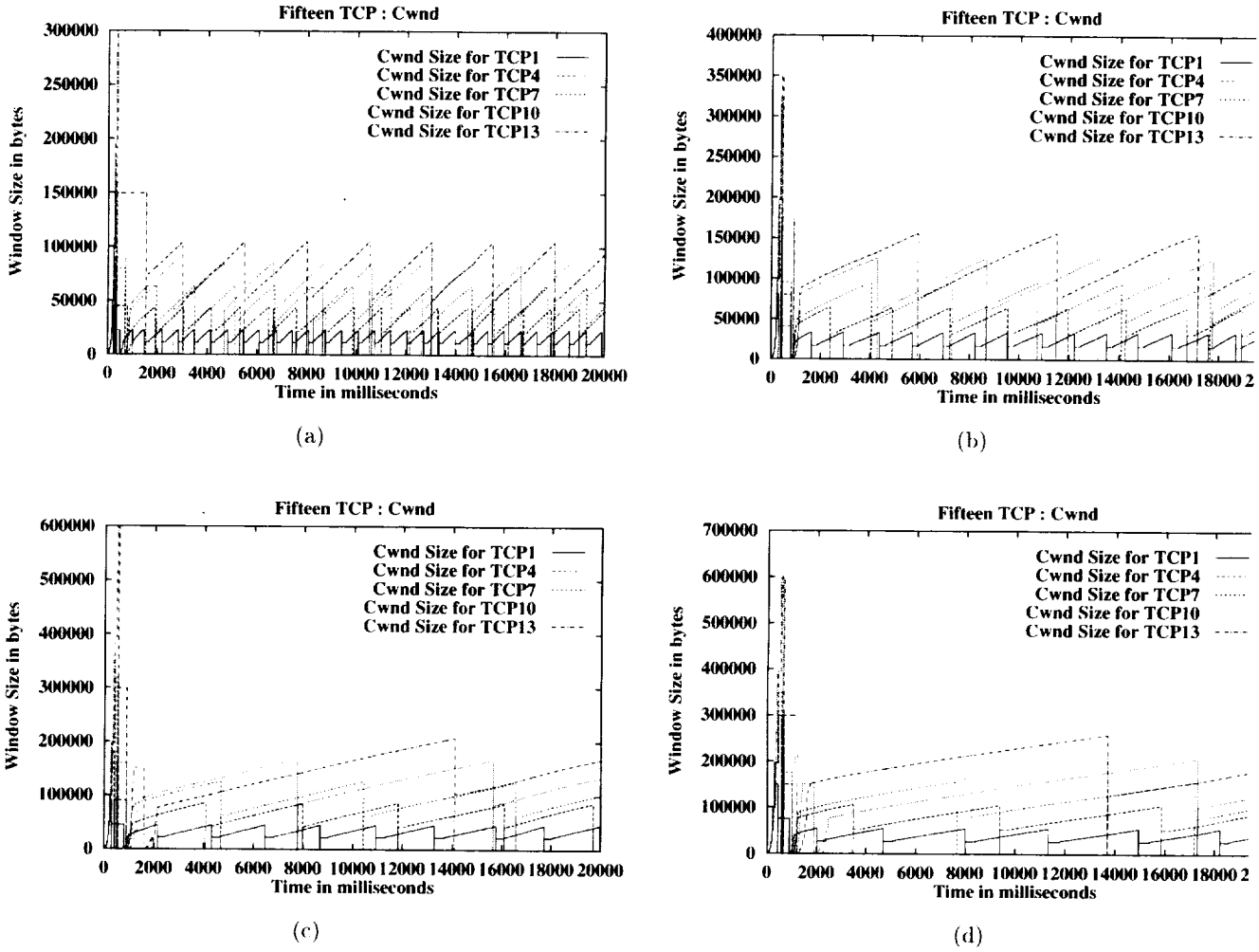


Figure 5: 15 TCP rate control by packet drop

proportionally to the increase in CWND (due to the increasing queuing delay of the 15 TCPs bottlenecked at the first switch). As a result, the average throughput achieved by a single TCP remains almost the same (see table 2) across the simulations.

The following list summarizes the observations from the graphs:

1. TCP throughput can be controlled by controlling its congestion window, which in turn, can be controlled by setting buffer thresholds to drop packets.

2. With a FIFO buffer, the average throughput achieved by a connection is proportional to the fraction of the buffer occupancy of the connection's cells. The achieved TCP throughput is independent of the absolute number of bytes from that TCP in the buffer, but on the relative number of bytes it the buffer.

3. As long as the fraction of buffer occupancy of a TCP can be controlled, its relative throughput is independent of the total number of packets in the buffer, and depends primarily on the fraction of packets of that TCP in the buffer.

4. At a very high buffer utilization, packets may be dropped due to buffer unavailability. This results in larger variations in TCP throughputs. At very high thresholds, the queuing delay also increases significantly, and may cause the TCP sources to timeout.

5. At very low buffer thresholds (high loss rates), TCP sources become unstable and tend to timeout. Also, very low buffer occupancies result in low network utilization. Since TCP can maintain a flow of 1 CWND worth of packets each round trip time, a total buffer occupancy of 1 bandwidth-delay product should provide good utilization [13].

4 The Differential Fair Buffer Allocation Scheme

In this section we describe the DFBA buffer management scheme that provides minimum rate guarantees to TCP/IP traffic. The scheme is based on the principles described above and uses per-VC accounting and thresholds to control TCP rates. The scheme stores the number of cells of each active VC in the buffer, where active VCs are those with at least one cell in the buffer.

As a result, the DFBA scheme is scalable with respect to the number of VCs, and only maintains fairness among the active VCs. Another feature of DFBA is that it uses dynamic thresholds to determine the fairness of the individual VCs. Dynamic thresholds allow the scheme to maintain approximate max-min fairness among remaining VCs when other active VCs are not using their entire guaranteed rates.

4.1 DFBA Description

The Differential Fair Buffer Allocation (DFBA) scheme uses the current queue length (buffer occupancy) as an indicator of network load. The scheme tries to maintain an optimal load so that the network is efficiently utilized, yet not congested. Figure 6 illustrates the operating region for DFBA. The high threshold (H) and the low threshold (L) represent the cliff and the knee respectively of the classical load versus delay/throughput graph. The goal is to operate between the knee and the cliff.

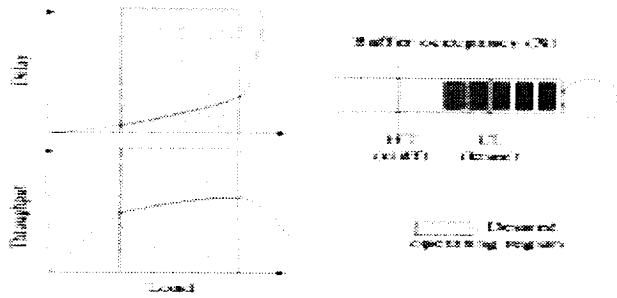


Figure 6: DFBA Target Operating Region

In addition to efficient network utilization, DFBA is designed to allocate buffer capacity fairly amongst competing VCs. This allocation is proportional to the MCRs of the respective VCs. The following variables are used by DFBA to fairly allocate buffer space:

X = Total buffer occupancy at any given time

L = Low buffer threshold

H = High buffer threshold

MCR_i = MCR guaranteed to VC_i

W_i = Weight of VC_i = MCR_i/(GFR capacity)

$W = \Sigma W_i$

X_i = Per-VC buffer occupancy ($X = \Sigma X_i$)

Z_i = Parameter ($0 \leq Z_i \leq 1$)

DFBA maintains the total buffer occupancy (X) between L and H. When X falls below L, the scheme attempts to bring the system to efficient utilization by accepting all incoming packets. When X rises above H, the scheme tries to control congestion by performing EPD. When X is between L and H, DFBA attempts to allocate buffer space in proportional to the MCRs, as determined by the W_i for each VC. When X is between L and H, the scheme also drops low priority (CLP=1) packets so as to ensure that sufficient buffer occupancy is available for CLP=0 packets.

Figure 7 illustrates the four operating regions of DFBA. The graph shows a plot of the current buffer occupancy X versus the normalized fair buffer occupancy ($\bar{X}_i$) for VC_i. If VC_i has a weight W_i, then its target buffer occupancy (X_i) should be $X \times W_i/W$. Thus, the normalized buffer occupancy of VC_i can be defined as $\bar{X}_i = X_i \times W/W_i$. The goal is to keep $\bar{X}_i$ as close to X as possible, as indicated by the solid $y = x$ line in the graph. Region 1 is the underload region, in which the current buffer occupancy is less than the low threshold L. In this case, the scheme tries to improve efficiency. Region 2 is the region with mild congestion because X is above L. As a result, any incoming packets with CLP=1 are dropped. Region 2 also indicates that VC_i has a larger buffer occupancy than its fair share (since $X_i > X \times W_i/W$). As a result, in this region, the scheme drops some incoming CLP=0 packets of VC_i, as an indication to the VC that it is using more than its fair share. In region 3, there is mild congestion, but VC_i's buffer occupancy is below its fair share. As a result, only CLP=1 packets of a VC are dropped when the VC is in region 3. Finally, region 4 indicates severe congestion, and EPD is performed here.

In region 2, the packets of VC_i are dropped in a probabilistic manner. This drop behavior is controlled by the drop probability function P{drop}. This is further discussed below. Figure 8 illustrates the drop conditions for DFBA.

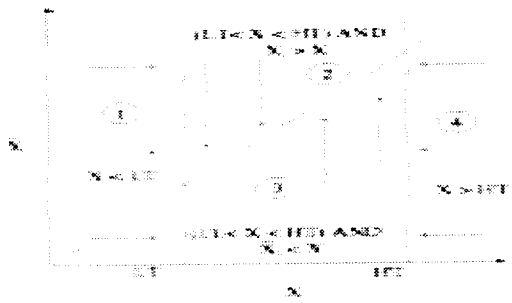


Figure 7: DFBA Drop Regions

Figure 8: DFBA Buffer Occupancies for Drop

Figures 11, 13 and 12 in the appendix contain the complete pseudocode for DFBA. The pseudocode is triggered by three events

- **Switch Initialization** (Figure 11): This sets the buffer occupancy variables (global and per-VC) to 0. The number of active VCs (N) is also set to zero. The variable 'Middle' indicates if a frame is being received for that VC, and the variable 'Drop' indicates if the frame being received (if Middle is true) is being discarded or accepted.

- **Dequeuing of a cell** (Figure 12): The global and individual buffer occupancies are decremented. If a VC becomes inactive then the number of active VCs and the total weights are also updated.

- **Receipt of a cell** (Figure 13): The DFBA algorithm is only applied to the first cell of every packet. If the first cell passes the DFBA tests, it is accepted, else it is discarded. For the

remaining cells of the frame, the DFBA tests are not performed. The cells are accepted or discarded depending on the action of the first cell. An exception is made for the last cell of a frame. The GFR standards advise not to discard the last cell of a frame so that frame delineation information can be carried on. As result, the last cell of a frame is not discarded in the pseudocode. The last cell can be discarded if all the previous cells in the frame were also discarded. This does not affect the properties of the scheme especially for larger frame sizes.

4.2 DFBA Drop Probability

The probability for dropping packets from a VC when it is in region 2 can be based on several factors. Probabilistic drop is used by several schemes including RED and FRED. The purpose of probabilistic drop is to notify TCP of congestion so that TCP backs off without a timeout. An aggressive drop policy will result in a TCP timeout. Different drop probability functions have different effects on TCP behavior. In general, a simple probability function can use RED like drop, while a more complex function can depend on all the variables defined above.

A sample drop probability can be defined using two main components

1. A fairness component, and

2. An efficiency component.

Thus, P{drop} = fn(Fairness component, Efficiency component). The contribution of the fairness component increases as the VC's buffer occupancy X_i increases above its fair share. The contribution of the efficiency component increases as the total buffer occupancy increases above L. A sample function could increase linearly as X_i increases from $X \times W_i/W$ to X, and as X increases from L to H. As a result, the drop probability is given by

$$P\{drop\} = Z_i \times (\alpha \times \frac{X_i - X \times W_i/W}{X \times (1 - W_i/W)} + (1 - \alpha)\frac{X - L}{H - L})$$

The parameter α is used to assign appropriate weights to the fairness and efficiency components of the drop probability. Z_i allows the scaling of the complete probability function based on per-VC characteristics. It has been shown that for a given TCP connection, a higher packet loss rate results

in a lower average TCP window. As a result, a higher drop probability also results in a lower TCP window. In fact, it has been shown that for random packet loss, the average TCP window size is inversely proportional to the square root of the packet loss probability. As a result, the average TCP data rate D is given by

$$D \propto \frac{1}{\sqrt{P\{\text{drop}\}}}$$

The data rate is in fact determined by the window size and the RTT of the connection. To maintain a high data rate, the desired window size should be large. As a result, the drop probability should be small. Similarly when the RTT is large, a larger window is needed to support the same data rate (since the delay-bandwidth product increases). As a result, a smaller drop rate should be used. DFBA can be tuned to choose a small Z_i for large latency VCs, as in the case of switches connected to satellite hops, or for VCs with high MCRs.

The probabilistic drop also randomizes the packets dropped within a VC. As a result, the scheme can maintain fairness among the TCPs within a VC. This can be accomplished by using an RED like drop probability.

4.3 DFBA Thresholds

The operation of DFBA is based on two static thresholds (L and H) and the per-VC dynamic thresholds $X \times W_i/W$. These thresholds determine the overall and per-VC buffer occupancies. To maintain high throughput, the average total buffer occupancy must be close to the bandwidth-delay products of the TCP connections [13].

On a per-VC level, DFBA employs a dynamic threshold strategy as opposed to a static threshold. When all sources are infinitely greedy, static thresholds can sometimes provide equivalent guarantees. However, when the number and the data rate of sources is dynamic, static thresholds cannot provide max-min fairness among competing connections. As an example, consider a scheme that allocates a static fraction of the buffer capacity to VCs depending on their MCRs. Consider a buffer size of 100 cells, a link data rate of 100 cells per sec, and three active VCs to be allocated 0.5, 0.25 and 0.25 of the capacity. The static thresholds in this case are 50, 25 and 25 cells respectively.

Consider two possible schemes. The first scheme drops incoming packets of a given VC as

soon as the VCs buffer occupancy exceeds its static threshold. Suppose that the first two VCs are bottlenecked elsewhere in the network, and only have 1 cell each in the buffer (so they are counted as active). Regardless of the traffic condition, VC3's cells are dropped probabilistically as soon as its buffer occupancy exceeds 25 cells. However, if the bandwidth-delay product of VC3 is more than 25 cells, then the buffer will empty out before the TCPs in VC3 have a chance to replenish it with the next window of packets. As a result, the link will be underutilized.

The second scheme fixes the underutilization problem by using a low threshold L like DFBA, so that if the total buffer occupancy is less than L, then all packets are accepted. When the total buffer occupancy exceeds L, then incoming packets are checked, and if their per-VC buffer occupancies exceed the static threshold, then the packets are dropped. Consider the above example again, and this time suppose only VC1 is bottlenecked elsewhere, while VC2 and VC3 are vying for a fair share of the capacity. Suppose that based on the network configuration, the threshold L is set to 60 cells, and both VC2 and VC3 currently have 25 cells each in the buffer while VC1 has no cells (inactive). Now more cells from VC2 are received, and since the total buffer occupancy (50 cells) is less than L, these cells are accepted. When the buffer occupancy crosses L, VC2 has 35 cells and VC1 has 25 cells in the buffer. Now if cells of VC3 are received, these cells are dropped because X > L and VC3's buffer occupancy is more than the static threshold. This will result in unfairness between VC2 and VC3 because VC2 will get more than VC3 although both were guaranteed an equal amount of the capacity.

In case of a dynamic threshold like in DFBA, VC3's cells will not be dropped because its per-VC buffer occupancy (25 cells) is less than its proportional share (since only two VCs are active) of the total buffer occupancy (60 cells). It will be able to share the usused capacity equally with VC2.

5 Simulation Configuration

The test results presented here are with DFBA for ATM interconnected TCP/IP networks. Figure 4 illustrates the basic test configuration. The figure shows 5 local IP/ATM edge switches connected to backbone ATM switches that implement GFR. Each local switch carries traffic from multiple TCPs as shown in the figure. The backbone link carries 5 GFR VCs, one from each local network.

Each VC thus carries traffic from several TCP connections. The length of the local hop is denoted by x km, and the length of the backbone hop is denoted by y km.

In our simulations, we varied five key parameters:

1. Number of TCPs. We used 10 TCPs per VC and 20 TCPs per VC for a total of 50 and 100 TCPs respectively.

2. Per-VC MCR allocations. Two sets of MCRs were chosen. In the first set, the MCR values were 12, 24, 36, 48 and 69 kcells/sec for VCs 1...5 respectively. This resulted in a total MCR allocation of about 50% of the GFR capacity. In the second set, the MCRs were 20, 40, 60, 80 and 100 kcells/sec for VCs 15 respectively, giving a total MCR allocation of 85% of the GFR capacity.

3. Buffer size. We first used a large buffer size of 25 kcells in the bottleneck backbone switch. We also analyzed DFBA performance with buffer sizes of 6 kcells, and 3 kcells.

4. Z_i. In most cases, the value of Z_i was chosen to be 1. We studied the effect of Z_i by decreasing it with increasing W_i.

5. In most cases, we fixed x=10 km and y=1000 km. This reflects a typical topology with local networks connected through a backbone GFR VC. We also analysed the effect of heterogenous TCP RTTs by setting x=5000km for VC3 in one simulation configuration.

The GFR capacity was fixed to the link rate of 155.52 Mbps (approx. 353207 cells per sec). α is fixed to 0.5 in this study. All TCP sources were persistent TCPs with SACK. The SACK implementation is based on [14]. Based on previous studies, [9], we set the thresholds L and H to 0.5 and 0.9 of the buffer capacity respectively. In the next section, we present the key results to show that DFBA provides MCR guarantees to TCP traffic. A complete parameter study of DFBA is presented in a future paper.

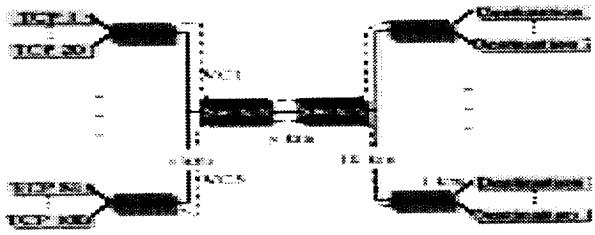


Figure 9: DFBA Simulation Confuration

Table 4: DFBA: 50 TCPs 5 VCs, 50% MCR Allocation

MCR	Throughput	Excess	Excess/MCR
4.61	11.86	7.25	1.57
9.22	18.63	9.42	1.02
13.82	24.80	10.98	0.79
18.43	32.99	14.56	0.79
23.04	38.60	15.56	0.68
69.12	126.88	57.77	

6 Simulation Results

Table 4 shows achieved throughput for a 50 TCP configuration. The total MCR allocation is 50% of the GFR capacity. The W_i values for the VCs are 0.034, 0.068, 0.102, 0.136, and 0.170. The achieved throughput column shows the total end to end TCP throughput for all the TCPs over the respective VC. The table shows that the VCs achieve the guaranteed MCR. Although the VCs with larger MCRs get a larger share of the unused capacity, the last column of the table indicates that the excess bandwidth is however not shared in proportional to MCR. This is mainly because the drop probabilities are not scaled with respect to the MCRs, i.e., because $Z_i = 1$ for all i. The total efficiency (achieved throughput over maximum possible throughput) is close to 100%.

Table 5 illustrates the performance of DFBA when 85% of the GFR capacity is allocated as the MCR values. In this case, the W_i's are 0.057, 0.113, 0.17, 0.23, and 0.28 for VC's 1...5 respectively. The table again shows that DFBA meets the MCR guarantees for VCs carrying TCP/IP traffic.

Table 6 validates the scheme for a larger number of TCPs. Each VC now carries traffic from 20

Table 5: DFBA: 50 TCPs 5 VCs, 85% MCR Allocation

MCR	Throughput	Excess	Excess/MCR
7.68	12.52	4.84	0.63
15.36	18.29	2.93	0.19
23.04	25.57	2.53	0.11
30.72	31.78	1.06	0.03
38.40	38.72	0.32	0.01
115.2	126.88	11.68	

Table 6: DFBA: 100 TCPs 5 VCs, 85% MCR Allocation

MCR	Throughput	Excess	Excess/MCR
7.68	11.29	3.61	0.47
15.36	18.19	2.83	0.18
23.04	26.00	2.96	0.13
30.72	32.35	1.63	0.05
38.40	39.09	0.69	0.02
115.2	126.92	11.72	

TCP connections, for a total of 100 TCPs. The total MCR allocation is 85% of the GFR capacity. All MCRs guarantees are met for a large number of TCPs and high MCR allocation.

Figure 10 illustrates the buffer occupancies of the 5 VCs in the bottleneck backbone switch. The figure shows that DFBA controls the switch buffer occupancy so that VCs with a lower MCR have a lower buffer occupancy than VCs with a higher MCR. In fact the average buffer occupancies are in proportion to the MCR values, so that FIFO scheduling can ensure a long-term MCR guarantee.

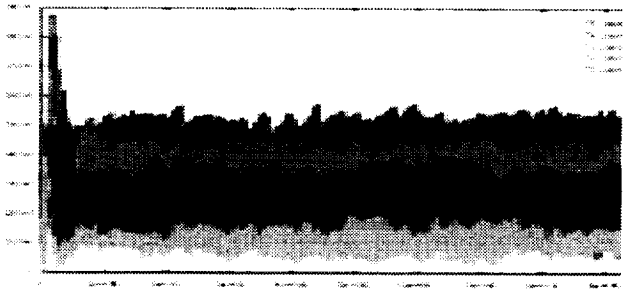


Figure 10: Per-VC Buffer Occupancy Levels

Tables 7 and 8 show that DFBA provides MCR guarantees even when the bottleneck backbone

Table 7: DFBA: Effect of Buffer Size (6 kcells)

MCR	Throughput	Excess	Excess/MCR
7.68	11.79	4.11	0.54
15.36	18.55	3.19	0.21
23.04	25.13	2.09	0.09
30.72	32.23	1.51	0.05
38.40	38.97	0.57	0.01
115.2	126.67	11.47	

Table 8: DFBA: Effect of Buffer Size (3 kcells)

MCR	Throughput	Excess	Excess/MCR
7.68	10.02	2.34	0.30
15.36	19.32	3.95	0.26
23.04	25.78	2.74	0.12
30.72	32.96	2.24	0.07
38.40	38.56	0.16	0.00
115.2	126.63	11.43	

switch has small buffers (6 kcells and 3 kcells respectively). The configuration uses 100 TCPs with 85% MCR allocation.

Table 9 shows the effect of Z_i on the fairness of the scheme in allocating excess bandwidth. We selected 2 values of Z_i based on the weights of the VCs. In the first experiment, Z_i was selected to be $(1 - W_i/W)$ so that VCs with larger MCRs have a lower Z_i. N the second experiment, Z_i was selected to be $(1 - W_i/W)^2$. The table shows that in the second case, sharing of the excess capacity is closely related to the MCRs of the VCs. An analytical assessment of the effect of Z_i on the excess capacity allocation by DFBA is a topic of further study.

Table 9: DFBA: Effect of Z_i

$Z_i = 1 - W_i/W$		$Z_i = (1 - W_i/W)^2$	
Excess	Excess/MCR	Excess	Excess/MCR
3.84	0.50	0.53	0.07
2.90	0.19	2.97	0.19
2.27	0.10	2.77	0.12
2.56	0.08	2.39	0.08
0.02	0.02	3.14	0.08

7 A Framework for Buffer Management

Several recent papers have focussed on fair buffer management for network traffic. All these proposals all drop packets when the buffer occupancy exceeds a certain threshold. The proposals for buffer management can be classified into four groups based on whether they maintain multiple buffer occupancies (Multiple Accounting - MA) or a single global buffer occupancy (Single Accounting - SA), and whether they use multiple discard thresholds (Multiple Thresholds - MT) or a single global discard Threshold (Single Threshold - ST). The SA schemes maintain a single count of the number of cells currently in the buffer. The MA schemes classify the traffic into several classes and maintain a separate count for the number of cells in the buffer for each class. Typically, each class correcponds to a single connection, and these schemes maintain per-connection occupancies. In cases where the number of connections far exceeds the buffer size, the added over-head of per-connection accounting may be very expensive. In this case, a set of active connections is defined as those connections with at least one packet in the buffer, and only the buffer occupancies of active connections are maintained.

Schemes with a global threshold (ST) compare the buffer occupancy(s) with a single threshold and drop packets when the buffer occupancy exceeds the threshold. Multiple thresholds (MT) can be maintained corresponding to classes, connections or to provide differentiated services. Several modifications to this drop behavior can be implemented. Some schemes like RED and FRED compare the average(s) of the buffer occupancy(s) to the threshold(s). Some like EPD maintain static threshold(s) while others like FBA maintain dynamic threshold(s). In some, packet discard may be probabilistic (RED) while others drop packets deterministically (EPD/PPD). Finally, some schemes may differentiate packets based on packet tags. Examples of packet tags are the CLP bit in ATM cells or the TOS octet in the IP header of the IETF's differentiated services architecture. Table 10 lists the four classes of buffer management schemes and examples of schemes for these classes. The example schemes are briefly discussed below.

The first SA-ST schemes included Early Packet Discard (EPD), Partial Packet Discard (PPD) [15] and Random Early Detection (RED) [20]. EPD and PPD improve network efficiency because they minimize the transmission of partial packets by the network. Since they do not discriminate

between connections in dropping packets, these schemes are unfair in allocating bandwidth to competing connections [9]. For example, when the buffer occupancy reaches the EPD threshold, the next incoming packet is dropped even if the packet belongs to a connection that is has received an unfair share of the bandwidth. Random Early Detection (RED) maintains a global threshold for the average queue. When the average queue exceeds this threshold, RED drops packets probabilistically using a uniform random variable as the drop probility. The basis for this is that uniform dropping will drop packets in proportion to the input rates of the connections. Connections with higher input rates will lose proportionally more packets than connections with lower input rates, thus maintaining equal rate allocation.

However, it has been shown in [16] that proportional dropping cannot guarantee equal bandwidth sharing. The paper also contains a proposal for Flow Random Early Drop (FRED). FRED maintains per-connection buffer occupancies and drops packets probabilistically if the per-connection occupancy exceeds the average queue length. In addition, FRED ensures that each connection has at least a minimum number of packets in the queue. In this way, FRED ensures that each flow has roughly the same number of packets *in the buffer*, and FCFS scheduling guarantees equal sharing of bandwidth. FRED can be classified as one that maintains per-connection queue lengths, but has a global threshold (MA-ST).

The Selective Drop (SD) [9] and Fair Buffer Allocation (FBA) [7] schemes are MA-ST schemes proposed for the ATM UBR service category. These schemes use per-connection accounting to

Table 10: Classification of Buffer Management Schemes

Group	Examples	Threshold Type (Static/Dynamic)	Drop Type (Deterministic/Probabilistic)	Tag/TOS sensitive (Yes/No)
SA-ST	EPD, PPD	Static	Deterministic	No
	RED	Static	Probabilistic	No
MA-ST	FRED	Dynamic	Probabilistic	No
	SD, FBA	Dynamic	Deterministic	No
	VQ+Dynamic EPD	Dynamic	Deterministic	No
MA-MT	PME+ERED	Static	Probabilistic	Yes
	DFBA	Dynamic	Probabilistic	Yes
	VQ+MCR scheduling	Dynamic	Deterministic	No
SA-MT	Priority Drop	Static	Deterministic	Yes

maintain the current buffer utilization of each UBR Virtual Channel (VC). A fair allocation is calculated for each VC, and if the VC's buffer occupancy exceeds its fair allocation, its subsequent incoming packet is dropped. Both schemes maintain a threshold R, as a fraction of the buffer capacity K. When the total buffer occupancy exceeds R×K, new packets are dropped depending on the VC_i's buffer occupancy (Y_i). In the Selective Drop scheme, a VC's *entire packet* is dropped if

$$(X > R) \text{ AND } (Y_i \times N_a/X > Z) \text{ (Selective Drop)}$$

$$(X > R) \text{ AND } (Y_i \times N_a/X > Z \times ((K - R)/(X - R))) \text{ (Fair Buffer Allocation)}$$

where N_a is the number of active VCs (VCs with at least one cell the buffer), and Z is another threshold parameter ($0 < Z \leq 1$) used to scale the effective drop threshold.

The Virtual Queuing (VQ) [18] scheme is unique because it achieves fair buffer allocation by emulates on a single FIFO queue, a per-VC queued round-robin server. At each cell transmit time, a per-VC accounting variable ($\hat{Y}_i$) is decremented in a round-robin manner, and is incremented whenever a cell of that VC is admitted in the buffer. When $\hat{Y}_i$ exceeds a fixed threshold, incoming packets of the ith VC are dropped. An enhancement called Dynamic EPD changes the above drop threshold to include only those sessions that are sending less than their fair shares.

Since the above MA-ST schemes compare the per-connection queue lengths (or virtual variables with equal weights) with a global threshold, they can only guarantee equal buffer occupancy (and thus throughput) to the competing connections. These schemes do not allow for specifying a guaranteed rate for connections or groups of connections. Moreover, in their present forms, they cannot support packet priority based on tagging.

Another enhancement to VQ, called MCR scheduling [17], proposes the emulation of a weighted scheduler to provide Minimum Cell Rate (MCR) guarantees to ATM connections. In this scheme, a per-VC weighted variable (W_i) is maintained, and compared with a global threshold. A time interval T is selected, at the end of which, W_i is incremented by $\text{MCR}_i \times T$ for each VC i. The remaining algorithm is similar to VQ. As a result of this weighted update, MCRs can be guaranteed. However, the implementation of this scheme involves the update of W_i for each VC after every time T. To provide tight MCR bounds, a smaller value of T must be chosen, and this increases the

complexity of the scheme. For best effort traffic (like UBR), thousands of VC could be sharing the buffer, and this dependence on the number of VCs is not an efficient solution to the buffer management problem. Since the variable W_i is updated differently for each VC i, this is equivalent to having different thresholds for each VC at the start of the interval. These thresholds are then updated in the opposite direction of W_i. As a result, VQ+MCR scheduling can be classified as a MA-MT scheme.

[19] proposes a combination of a Packet Marking Engine (PME) and an Enhanced RED scheme based on per-connection accounting and multiple thresholds (MA-MT). PME+ERED is designed for the IETF's differentiated services architecture, and can provide loose rate guarantees to connections. The PME measures per-connection bandwidths and probabilistically marks packets if the measured bandwidths are lower than the target bandwidths (multiple thresholds). High priority packets are marked, and low priority packets are unmarked. The ERED mechanism is similar to RED except that the probability of discarding marked mackets is lower that that of discarding unmarked packets. The PME in a node calculates the observed bandwidth over an update interval, by counting the number of accepted packets of each connection by the node. Calculating bandwidth can be complex that may require averaging over several time intervals. Although it has not been formally proven, Enhanced RED can suffer from the same problem as RED because it does not consider the number of packets actually in the queue.

A simple SA-MT scheme can be designed that implements multiple thresholds based on the packet priorities. When the global queue length (single accounting) exceeds the first threshold, packets tagged as lowest priority are dropped. When the queue length exceeds the next threshold, packets from the lowest and the next priority are dropped. This process continues until EPD/PPD is performed on all packets. The performance of such schemes needs to be analyzed. However, these schemes cannot provide per-connection throughput guarantees and suffer from the same problem as EPD, because they do not differentiate between overloading and underloading connections.

Table 11 illustrates the fairness properties of the four buffer management groups presented above.

Table 11: Properties of Buffer Management Schemes

Group	Equal bandwidth allocation	Weighted bandwidth allocation
SA-ST	No	No
MA-ST	Yes	No
MA-MT	Yes	Yes
SA-MT	-	-

8 Concluding Remarks

In this paper, we have used FIFO buffers to control TCP rates by buffer management. An optimal set of thresholds should be selected that is high enough to provide sufficient network utilization, and is low enough to allow stable operation. The achieved TCP throughputs are in proportion to the fraction of the average buffer occupied by the VC.

This paper does not study the effect of non-adaptive traffic (like UDP) on the drop policy. It appears that for non-adaptive traffic, the thresholds must be set lower than those for adaptive traffic (for the same MCR), and the dropping should be more strict when the buffer occupancy crosses the threshold. However, the scheme works well when non-adaptive traffic is queued separately from adaptive traffic. A general scheme can be designed by defining an adaptivity metric that determines the dropping rate - for example, a higher adaptivity metric would mean better response to packet drop, and a lower drop rate (or drop probability). Thus TCP would have a higher adaptivity metric than UDP, and even within TCP, longer lived connections of ftp traffic would have a higher adaptivity metric than short lived WWW connections. This a topic for further study in general behavior of flows in response to various pack drop rates.

In this paper we have not studied the effect of network based tagging in the context of GFR. In the strict sense, GFR only provides a low CLR guarantee to the CLP=0 cell stream i.e., the cells that were not tagged by the source and passed the GCRA conformance test. However, when source (this could be a non-ATM network element like a router) based tagging is not performed, it is not clear if the CLP0 stream has any significance over the CLP1 stream. Moreover, network tagging is an option that must be signaled during connection establishment.

Although buffer management techniques provide a useful traffic management mechanism for

best effort, services they have certain inherent limitations in their ability to recognize flows. Per-flow accounting schemes depend on their ability to recognize flows in the traffic. In ATM, these flows take the form of ATM VCs, while in IP, flows can be constructed from the IP header using the five-tuple consisting of the source IP address, destination IP address, Tranport protocol (TCP or UDP), and TCP/UDP source and destination port numbers. With new developments in Multiprotocol Label Switching (MPLS), and differentiated-services, flow classification can be achieved at a coarser level using the MPLS Label Switch Paths (LSP), and the six bits of the Differentiated Services Code Point (DSCP) in the Type of Service byte of the IP header. For TCP traffic, throughput is inherently dependent on the round trip time (RTT) of the connection. For the same loss rate, TCP connections with a shorter RTT will achieve a higher throughput than those with a longer RTT. If a single ATM VC or IP flow carries TCP connections with different RTTs, then a per-flow accounting scheme that does not differentiate the TCP connections within the flow will not be able to guarantee fair allocation of throughput to the TCPs with different RTTs. RED suffers from these limitation because it does not distinguish TCP connections. FRED provides a solution, but needs to account for each TCP flow. A more scalable version of FRED can be designed to classify flows based on IP headers described above, but would have the same limitations as RED for individual TCPs within a flow. DFBA and other VC based schemes cannot guarantee fairness to TCPs within a VC for the same reason.

References

[1] ATM Forum, "ATM Traffic Management Specification Version 4.1," December 1998.

[2] Debashis Basak, Surya Pappu, "TCP over GFR Implementation with Different Service Disciplines: A Simulation Study"

[3] Debashis Basak, Surya Pappu, "GFR Implementation Alternatives with Fair Buffer Allocation Schemes," ATM Forum 97-0528.

[4] Olivier Bonaventure, "A simulation study of TCP with the proposed GFR service category," DAGSTUHL Seminar 9725, High Performance Networks for Multimedia Applications, June

1997, Germany

[5] John B. Kenney, "Satisfying UBR+ Requirements via a New VBR Conformance Definition," ATM FORUM 97-0185.

[6] John B. Kenney, "Open Issues on GFR Frame Discard and Frame Tagging," ATM FORUM 97-0385.

[7] Juha Heinanen, and Kalevi Kilkki, "A fair buffer allocation scheme," Unpublished Manuscript.

[8] R. Goyal, R. Jain et.al, "Simulation Experiments with Guaranteed Frame Rate for TCP/IP Traffic," ATM Forum 97-0607.

[9] R. Goyal, R. Jain, S. Kalyanaraman, S. Fahmy and Seong-Cheol Kim, "UBR+: Improving Performance of TCP over ATM-UBR Service," Proc. ICC'97, June 1997.

[10] R. Goyal, R. Jain et.al., "Selective Acknowledgements and UBR+ Drop Policies to Improve TCP/UBR Performance over Terrestrial and Satellite Networks," To appear, Proc. IC3N'97, September 1997. [1]

[11] Roch Guerin, and Juha Heinanen, "UBR+ Service Category Definition," ATM FORUM 96-1598, December 1996.

[12] Roch Guerin, and Juha Heinanen, "UBR+ Enhancements," ATM FORUM 97-0015, February 1997.

[13] Sastri Kota, Rohit Goyal, Raj Jain, "Satellite-ATM Network Architectural Considerations and TCP/IP Performance," Proc. of the 3rd Ka-Band Utilization Conference, Italy, 1997.

[14] Kevin Fall, Sally Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," Computer Communications Review, July 1996

[15] Allyn Romanov, Sally Floyd, "Dynamics of TCP Traffic over ATM Networks," IEEE Journal of Selected Areas In Telecommunications, May 1995.

[1] All our papers and ATM Forum contributions are available from http://www.cis.ohio-state.edu/~jain

[16] Dong Lin, Robert Morris, "Dynamics of Random Early Detection," Proceedings of SIGCOMM97, 1997.

[17] Kai-Yeung Siu, Yuan Wu, Wenge Ren, "Virtual Queuing Techniques for UBR+ Service in ATM with Fair Access and Minimum Bandwidth Guarantee," Proceedings of Globecom'97, 1997.

[18] Yuan Wu, Kai-Yeung Siu, Wenge Ren, "Improved Virtual Queuing and Dynamic EPD Techniques for TCP over ATM," Proceedings of ICNP97, 1997.

[19] Wu-chang Feng, Dilip Kandlur, Debanjan Saha, Kang G. Shin, "Techniques for Eliminating Packet Loss in Congested TCP/IP Networks," University of Michigan, CSE-TR-349-97, November 1997.

[20] Sally Floyd, Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transaction on Networking, August 1993.

9 Appendix: Pesudocode

```
        X   0
        N   0
For each VC i
        Middle_i   FALSE
        Drop_i   FALSE
        X_i   0
```

Figure 11: Initialization

```
X_i   X_i − 1
X   X − 1
IF (X_i = 0) THEN
        N   N − 1
        W   W − W_i
ENDIF
```

Figure 12: Cell dequeued

```
IF (cell is NOT the last cell of the pkt) THEN
        IF (NOT (Middle_i)) THEN
                IF ((X ≤ L)∨
                  (X_i > X × W_i/W ∧ X < H ∧ rand_var > P{drop} ∧ CELL.CLP = 0)∨
                  (X_i < X × W_i/W ∧ X < H ∧ CELL.CLP = 0)) THEN
                        Middle_i   TRUE
                        IF (X_i = 0) THEN
                                N   N + 1
                                W   W + W_i
                        ENDIF
                        X_i   X_i + 1
                        X   X + 1
                        Enqueue cell
                ELSE
                        Middle_i   TRUE
                        Drop_i   TRUE
                        Drop cell
                ENDIF
        ELSE
                IF (Drop_i) THEN
                        Drop cell
                ELSE
                        Enqueue cell if possible
                        IF (enqueued) THEN
                                IF (X_i = 0) THEN
                                        N   N + 1
                                        W   W + W_i
                                ENDIF
                                X_i   X_i + 1
                                X   X + 1
                        ELSE
                                Drop_i   1
                        ENDIF
                ENDIF
        ENDIF
ELSE
        Enqueue cell if possible
        Middle_i   FALSE
        Drop_i   FALSE
        IF (Enqueued) THEN
                IF (X_i = 0) THEN
                        N   N + 1
                        W   W + W_i
                ENDIF
                X_i   X_i + 1
                X   X + 1
        ENDIF
ENDIF
```

Figure 13: Cell Received

****

ATM Forum Document Number: ATM_Forum/98-0406

****

TITLE: GFR Implementation Options

****

SOURCE: Rohit Goyal, Raj Jain, Sonia Fahmy, Bobby Vandalore
The Ohio State University,
Department of Computer and Information Science,
2015 Neil Ave, DL 395, Columbus, OH 43210
Phone: 614-688-4482
{goyal,jain}@cis.ohio-state.edu

This work is partially sponsored by the NASA Glenn Research Center Under Contract Number NAS3-97198

****

DISTRIBUTION: ATM Forum Technical Committee
Traffic Management Working Group

****

DATE: July, 1998 (Portland)

****

ABSTRACT: Section VII.2 of the baseline text contains two example implementations of the GFR service. Several other GFR implementations have been proposed in the past meetings and in recent literature. This contribution provides text to enhance section VII.2 by providing the various design options for GFR.

****

****

1 Introduction

Section VII.2 of the baseline text describes two sample implementations of the GFR service. We propose the following text to be added to modify section VII.2.

2 Motion

The following text should be used a replacement for section VII.2 in the baseline text document:

VII.2 Example Designs and Implementations of the GFR Service

There are three basic design options that can be used by the network to provide the per-VC minimum rate guarantees for GFR -- tagging, buffer management, and queueing:

- **Tagging**: Network based tagging (or policing) can be used as a means of marking non-conforming packets before they enter the network. This form of tagging is usually performed when the connection enters the network. Network based tagging on a per-VC level requires some per-VC state information to be maintained by the network. Tagging can isolate conforming and non-conforming traffic of each VC so that other rate enforcing mechanisms can use this information to treat the conforming traffic in preferentially over non-conforming traffic. For example, policing can be used to discard non-conforming packets, thus allowing only conforming packets to enter the network.

- **Buffer management**: Buffer management is typically performed by a network element (like a switch or a router) to control the number of packets entering its buffers. In a shared buffer environment, where multiple VCs share common buffer space, per-VC buffer management can control the buffer occupancies of individual VCs. Per-VC buffer management uses per-VC accounting to keep track of the buffer occupancies of each VC. Examples of per-VC buffer management schemes are Selective Drop and Fair Buffer Allocation. Per-VC accounting introduces overhead, but without per-VC accounting it is difficult to control the buffer occupancies of individual VCs (unless non-conforming packets are dropped at the entrance to the network by the policer).

- **Scheduling**: While tagging and buffer management control the entry of packets into a network element, queuing strategies determine how packets are scheduled onto the next hop. In a FIFO queue, packets are scheduled in the order in which they enter the buffer. As a result, FIFO queuing cannot isolate packets from various VCs at the egress of the queue. Per-VC queuing, on the other hand, maintains a separate queue for each VC in the buffer. A scheduling mechanism can select between the queues at each scheduling time. However, scheduling adds the overhead of per-VC queuing and the service discipline.

Table 1 lists the various options available for queuing, buffer management and support for tagged cells. A switch could use any of the available options in each category for its GFR implementation.

Table 1 GFR Options

Queuing	FIFO	Per-VC
Buffer Management	Global Threshold (No per-VC accounting)	Per-VC Threshold (per-VC accounting)
Tag Sensitive Buffer Management	Supported	Not Supported

The following subsections list some sample GFR implementations based on this framework. Section VII.2.1 presents an implementation that uses Per-VC queuing with per-VC thresholds for untagged cells, as well as support for treating tagged cells separately from untagged cells. Section VII.2.2 presents a sample implementation with FIFO queuing and two global thresholds, i.e., it is sensitive to tags, but does not employ per-VC buffer management. Section VII.2.3 describes the Differential Fair Buffer Allocation Policy that uses FIFO queuing, per-VC thresholds and supports tagging by the source or the network.

VII.2.1 GFR Implementation using Weighted Fair Queuing and per-VC accounting

(Unchanged)

VII.2.2 GFR Implementation Using Tagging and FIFO Queue

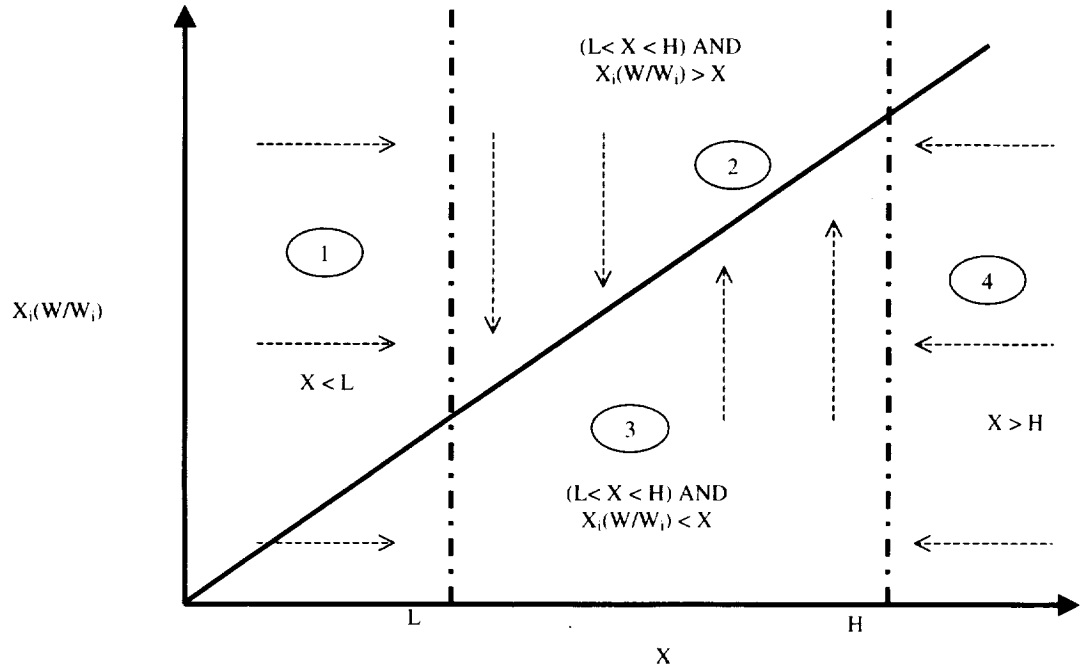
(Unchanged)

VII.2.3 GFR Implementation Using Differential Fair Buffer Allocation

Differential Fair Buffer Allocation (DFBA) uses the current queue length as an indicator of network load. The scheme tries to maintain an optimal load so that the network is efficiently utilized, yet not congested. In addition to efficient network utilization, DFBA is designed to allocate buffer capacity fairly amongst competing VCs. This allocation is proportional to the MCRs of the respective VCs. The following variables are used by DFBA to fairly allocate buffer space:

- X = Total buffer occupancy at any time
- L = Low buffer threshold
- H = High buffer threshold
- MCR_i = MCR guaranteed to VC_i
- W_i = Weight of VC_i = MCR_i/(GFR capacity)
- $W = \Sigma W_i$
- X_i = Per-VC buffer occupancy ($X = \Sigma X_i$)
- Z_i = Parameter ($0 <= Z_i <= 1$)

DFBA tries to keep the total buffer occupancy (X) between L and H. When X falls below L, the scheme attempts to bring the system to efficient utilization by accepting all incoming packets. When X rises above H, the scheme tries to control congestion by performing EPD. When X is between L and H, DFBA attempts to allocate buffer space in proportional to the MCRs, as determined by the W_i for each VC. When X is between L and H, the scheme also drops low priority (CLP=1) packets so as to ensure proportional buffer occupancy for CLP=0 packets.



The figure above illustrates the four operating regions of DFBA. The graph shows a plot of the current buffer occupancy X versus the normalized fair buffer occupancy for VC_i. If VC_i has a weight W_i, then its target buffer occupancy should be $X*W_i/W$. Thus, the normalized buffer occupancy of VC_i is X_i*W/W_i. The goal is to keep this normalized occupancy as close to X as possible, as indicated by the solid line in the graph. Region 1 is the underload region, in which the current buffer occupancy is less than the low threshold L. In this case, the scheme tries to improve efficiency. Region 2 is the region with mild congestion because X is above L. As a result, any incoming packets with CLP=1 are dropped. Region 2 also indicates that VC_i has a larger buffer occupancy than its fair share (since $X_i > X*W_i/W$). As a result, in this region, the scheme drops some incoming CLP=0 packets of VC_i, as an indication to the VC that it is using more than its fair share. In region 3, there is mild congestion, but VC_i's buffer occupancy is below its fair share. As a result, only CLP=1 packets of a VC are dropped when the VC is in region 3. Finally, region 4 indicates severe congestion, and EPD is performed here.

In region 2, the packets of VC_i are dropped in a probabilistic manner. This drop behavior is controlled by the parameter Z_i, whose value depends on the connection characteristics. This is further discussed below.

The probability for dropping CLP=0 packets from a VC when it is in region 2 depends on several factors. The drop probability has two main components – the fairness component,

and the efficiency component. Thus, P{drop} = fn(Fairness component, Efficiency component). The contribution of the fairness component increases as the VC's buffer occupancy X_i increases above its fair share. The drop probability is given by

$$P\{drop\} = Z_i\left(\alpha\frac{X_i - X\times W_i/W}{X(1-W_i/W)} + (1-\alpha)\frac{X-L}{H-L}\right)$$

The parameter α is used to assign appropriate weights to the fairness and efficiency components of the drop probability. Z_i allows the scaling of the complete probability function based on per-VC characteristics.

The following DFBA algorithm is executed when the first cell of a frame arrives at the buffer.

BEGIN

IF (X < L) THEN

Accept frame

ELSE IF (X > H) THEN

Drop frame

ELSE IF (L < X < H) AND (X_i < X*W_i/W)) THEN

Drop CLP1 frame

ELSE IF (L < X < H) AND (X_i > X*W_i/W)) THEN

Drop CLP1 frame

Drop CLP0 frame with

$$P\{drop\} = Z_i\left(\alpha\frac{X_i - X\times W_i/W}{X(1-W_i/W)} + (1-\alpha)\frac{X-L}{H-L}\right)$$

ENDIF

END

VII.2.4 Evaluation Criteria

(From VII.2.3 in the baseline text document.)

III. Buffer requirements as a function of delay-bandwidth products

III.A

"Analysis and Simulation of Delay and Buffer Requirements of Satellite-ATM Networks for TCP/IP Traffic"

III.B

"UBR Buffer Requirements for TCP/IP over Satellite Networks"

Analysis and Simulation of Delay and Buffer Requirements of Satellite-ATM Networks for TCP/IP Traffic

Rohit Goyal[1], Sastri Kota[2], Raj Jain[1], Sonia Fahmy[1], Bobby Vandalore[1], Jerry Kallaus[2]

1. Department of Computer and Information Science, The Ohio State University, 2015 Neil Ave, DL395, Columbus, OH 43210, Phone: 614-688-4482, Fax: 614-292-2911, Email: goyal@cis.ohio-state.edu

2. Lockheed Martin Telecommunications, 1272 Borregas Avenus, Bldg B/551 O/GB-70, Sunnyvale, CA 94089, Phone: 408-543-3140, Fax: 408-543-3104, Email: sastri.kota@lmco.com

Abstract

In this paper we present a model to study the end-to-end delay performance of a satellite-ATM network. We describe a satellite-ATM network architecture. The architecture presents a trade-off between the on-board switching/processing features and the complexity of the satellite communication systems. The end-to-end delay of a connection passing through a satellite constellation consists of the transmission delay, the uplink and downlink ground terminal-satellite propagation delay, the inter-satellite link delays, the on-board switching, processing and buffering delays. In a broadband satellite network, the propagation and the buffering delays have the most impact on the overall delay. We present an analysis of the propagation and buffering delay components for GEO and LEO systems. We model LEO constellations as satellites evenly spaced in circular orbits around the earth. A simple routing algorithm for LEO systems calculates locally optimal paths for the end-to-end connection. This is used to calculate the end-to-end propagation delays for LEO networks. We present a simulation model to calculate the buffering delay for TCP/IP traffic over ATM ABR and UBR service categories. We apply this model to calculate total end-to-end delays for TCP/IP over satellite-ATM networks.

1 Introduction

ATM technology is expected to provide quality of service based networks that support voice, video and data applications. ATM was originally designed for fiber based terrestrial networks that exhibit low latencies and low error rates. With the widespread availability of multimedia technology, and an increasing demand for electronic connectivity across the world, satellite networks will play an indispensable role in the deployment of global networks. Ka-band satellites using the gigahertz frequency spectrum can reach user terminals across most of the populated world. As a result, ATM based satellite networks can be effectively used to provide real time as well as non-real time communications services to remote areas.

Satellite communications technology offers a number of advantages over traditional terrestrial point-to-point networks [AKYL97]. These include,

1. wide geographic coverage including interconnection of "ATM islands",
2. multipoint to multipoint communications facilitated by the inherent broadcasting ability of satellites,

3. bandwidth on demand, or Demand Assignment Multiple Access (DAMA) capabilities, and
4. an alternative to fiber optic networks for disaster recovery options.

However, satellite systems have several inherent constraints. The resources of the satellite communication network, especially the satellite and the earth station are expensive and typically have low redundancy. These must be robust and be used efficiently. Also, satellite systems use a Time Division Multiplexed (TDM) physical layer, where individual earth stations can transmit frames during fixed time slots. The cell based ATM layer must be mapped onto the frame based satellite layer. This involves the use of efficient bandwidth allocation strategies for Demand Assignment Multiple Access (DAMA) based media access techniques.

Current and proposed satellite communications networks use low earth orbit (LEO) constellations as well as geosynchronous (GEO) satellite systems. GEO satellites have a high propagation delay but a few satellites are enough to provide connectivity across the globe. LEO satellites have lower propagation delays due to their lower altitudes, but many satellites are needed to provide global service. While LEO systems have lower propagation delay, they exhibit higher delay variation due to connection handovers and other factors related to orbital dynamics [IQTC97]. The effects of the propagation delays are further intensified by the buffering delays that could be of the order of the propagation delays especially for best effort TCP/IP traffic. The large delays in GEOs, and delay variations in LEOs, affect both real time and non-real time applications. Many real time applications are sensitive to the large delay experienced in GEO systems, as well as to the delay variation experienced in LEO systems. In an acknowledgment and timeout based congestion control mechanism (like TCP), performance is inherently related to the delay-bandwidth product of the connection. Moreover, TCP Round Trip Time (RTT) measurements are sensitive to delay variations that may cause false timeouts and retransmissions. As a result, the congestion control issues for broadband satellite networks are somewhat different from those of low latency terrestrial networks. Both interoperability, as well as performance issues must be addressed before data, voice and video services can be provided over a Satellite-ATM network.

In this paper, we present a model for analyzing the delay performance of LEO and GEO satellite systems. We present an overview of our satellite-ATM network architecture. This model applies both to LEO and GEO systems. We describe the various components of the delay experienced by the cells of a connection over the satellite network. The two most important delay components are propagation and buffering delays. We present a model for calculating the propagation delay in a satellite network. We provide values for the delays experienced by connections traversing sample LEO constellations. We describe a simulation model to compute the buffer requirements of a satellite-ATM network for TCP/IP file transfer traffic. This analysis, performed for TCP/IP over ABR and UBR service categories, provides an estimate of the buffering delay experienced by a TCP/IP connection. A case study of the total delay experienced by a TCP connection over GEO and LEO systems concludes this paper.

2 Satellite-ATM Network Architecture

In this section, we briefly overview the basic architecture of a Satellite-ATM network. We first present a brief overview of the QoS guarantees in ATM networks. This gives the reader an idea of the kinds of

guarantees that are expected of a satellite-ATM network. We then describe the various components of the architecture and overview their functionality.

2.1 Quality of Service in ATM Networks

ATM networks carry traffic from multiple service categories, and support Quality of Service (QoS) requirements for each service category. The ATM-Forum Traffic Management Specification 4.0 [TM4096] defines five service categories for ATM networks. Each service category is defined using a traffic contract and a set of QoS parameters. The *traffic contract* is a set of parameters that specify the characteristics of the source traffic. This defines the requirements for compliant cells of the connection. The *QoS parameters* are negotiated by the source with the network, and are used to define the expected quality of service provided by the network. For each service category, the network guarantees the negotiated QoS parameters if the end system complies with the negotiated traffic contract. For non-compliant traffic, the network need not maintain the QoS objective.

The *Constant Bit Rate (CBR)* service category is defined for traffic that requires a constant amount of bandwidth, specified by a Peak Cell Rate (PCR), to be continuously available. The network guarantees that all cells emitted by the source that conform to this PCR will be transferred by the network with minimal cell loss, and within fixed bounds of cell delay and delay variation. The *real time Variable Bit Rate (VBR-rt)* class is characterized by PCR, Sustained Cell Rate (SCR) and a Maximum Burst Size (MBS) in cells that controls the bursty nature of VBR traffic. The network attempts to deliver cells within fixed bounds of cell delay and delay variation. *Non-real-time VBR* sources are also specified by PCR, SCR and MBS, but are less sensitive to delay and delay variation than the real time sources. The network does not specify any delay and delay variation parameters for the VBR-nrt service.

The *Available Bit Rate (ABR)* service category is specified by a PCR and Minimum Cell Rate (MCR) which is guaranteed by the network. The bandwidth allocated by the network to an ABR connection may vary during the life of a connection, but may not be less than MCR. ABR connections use a rate-based closed-loop feedback-control mechanism for congestion control. The network tries to maintain a low Cell Loss Ratio by changing the allowed cell rates (ACR) at which a source can send. The *Unspecified Bit Rate (UBR)* class is intended for best effort applications, and this category does not support any service guarantees. UBR has no built in congestion control mechanisms. The UBR service manages congestion by efficient buffer management policies in the switch. A new service called Guaranteed Frame Rate (GFR) is being introduced at the ATM Forum and the ITU-T. GFR is based on UBR, but guarantees a minimum rate to connections. The service also recognizes AAL5 frames, and performs frame level dropping as opposed to cell level dropping.

In addition, the ITU-T has specified four QoS classes to be used to deliver network based QoS [I35696]. It is imperative that a broadband satellite network be able to support the various QoS services specified by the standards. Most importantly, the network should be able to support TCP/IP based data applications that constitute the bulk of Internet traffic.

Most of the parameters specified in the standards are relevant only to terrestrial networks. These values have to be re-evaluated for Satellite-ATM networks. For example, the ITU-T specifies a maximum cell transfer delay of 400 ms for the ITU Class 1 stringent service [I35696]. This class is expected to carry CBR traffic for real-time voice communications over ATM. However, the 400ms maximum delay

needs to be reviewed to ensure that it properly accounts for the propagation delays in geosynchronous satellite networks. The peak-to-peak cell delay variation of QoS Class 1 is also specified to be a maximum of 3 ms by the ITU-T [I35696]. This value may be too stringent for many satellite systems. As a result, the QoS parameters are under careful consideration by ITU-4B [IT4B97] In this context, the ITU-4B preliminary draft recommendations on transmission of Asynchronous Transfer Mode (ATM) Traffic via Satellite is in the process of development.

3 Delay Requirements of Applications

We briefly discuss the basic qualitative requirements of three classes of applications, interactive voice/video, non-interactive voice/video and TCP/IP file transfer. Interactive voice requires very low delay (ITU-T specifies a delay of less than 400 ms to prevent echo effects) and delay variation (up to 3 ms specified by ITU-T). GEO systems have a high propagation delay of at least 250 ms from ground terminal to ground terminal. If two GEO hops are involved, then the inter-satellite link delay could be about 240 ms. Other delay components are additionally incurred, and the total end-to-end delay can be higher than 400 ms. Although the propagation and inter-satellite link delays of LEOs are lower, LEO systems exhibit high delay variation due to connection handovers, satellite and orbital dynamics, and adaptive routing. This is further discussed in section 5.3. Non-interactive voice/video applications are real-time applications whose delay requirements are not as stringent as their interactive counterparts. However, these applications also have stringent jitter requirements. As a result, the jitter characteristics of GEO and LEO systems must be carefully studied before they can service real time voice-video applications.

The performance of TCP/IP file transfer applications is throughput dependent and has very loose delay requirements. As a result, both GEOs and LEOs with sufficient throughput can meet the delay requirements of file transfer applications. It is often misconstrued that TCP is throughput limited over GEOs due to the default TCP window size of 64K bytes. The TCP large windows option allows the TCP window to increase beyond 64K bytes and results in the usage of the available capacity even in high bandwidth GEO systems. The efficiency of TCP over GEO systems can be low because the TCP window based flow control mechanism takes several round trips to fully utilize the available capacity. The large round trip time in GEOs results in capacity being wasted during the ramp-up phase. To counter this, the TCP spoof protocol is being designed that splits the TCP control loop into several segments. However this protocol is currently incompatible with end-to-end IP security protocols. Several other mechanisms are being developed to mitigate latency effects over GEOs [GOY97a][TCPS98].

The TCP congestion control algorithm inherently relies on round trip time (RTT) estimates to recover from congestion losses. The TCP RTT estimation algorithm is sensitive to sudden changes in delays as may be experienced in LEO constellations. This may result in false timeouts and retransmits at the TCP layer. More sophisticated RTT measurement techniques are being developed for TCP to counter the effects of delay jitter in LEO systems [TCPS98].

3.1 Architectural Issues

Figure 1 illustrates a satellite-ATM network represented by a ground segment, a space segment, and a network control center. The ground segment consists of ATM networks that may be further connected

to other legacy networks. The network control center (NCC) performs various management and resource allocation functions for the satellite media. Inter-satellite links (ISL) in the space segment provide seamless global connectivity to the satellite constellation. The network allows the transmission of ATM cells over satellite, multiplexes and demultiplexes ATM cell streams from uplinks and downlinks, and maintains the QoS objectives of the various connection types. The satellite-ATM network also includes a satellite-ATM interface device connecting the ATM network to the satellite system. The interface device transports ATM cells over the frame based satellite network, and demultiplexes ATM cells from the satellite frames. The device typically uses a DAMA technique to obtain media access to the satellite physical layer. The interface unit is also responsible for forward error correction techniques to reduce the error rates of the satellite link. The unit must maintain ATM quality of service parameters at the entrance to the satellite network. As a result, it translates the ATM QoS requirements into corresponding requirements for the satellite network. This interface is thus responsible for resource allocation, error control, and traffic control. Details about this model can be obtained from [KOTA97].

This architectural model presents several design options for the satellite and ground network segments. These options include

1. No on-board processing or switching.

2. On-board processing with ground ATM switching.

3. On-board processing and ATM switching.

About 53% of the planned Ka-band satellite networks propose to use on-board ATM like fast packet switching [PONZ97]. An overview of the network architectures of some of the proposed systems can be found in [WUPI94]. In a simple satellite model without on-board processing or switching, minimal on-board buffering is required. However, if on-board processing is performed, then on-board buffering is needed to achieve the multiplexing gains provided by ATM. On-board processing can be used for resource allocation and media access control (MAC). MAC options include TDMA, FDMA, and CDMA and can use contention based, reservation based, or fixed media access control. Demand Assignment Multiple Access (DAMA) [KOT97b] can be used with any of the MAC options. If on-board processing is not performed, DAMA must be done by the NCC. On-board DAMA decreases the response time of the media access policy by half because link access requests need not travel to the NCC on the ground any more. In addition to media access control, ABR explicit rate allocation or EFCI control, and UBR/GFR buffer management can also be performed on-board the satellite. On-board switching may be used for efficient use of the network by implementing adaptive routing/switching algorithms. Trade-offs must be made with respect to the complexity, power and weight requirements for providing on-board buffering, switching and processing features to the satellite network. In addition, on-board buffering and switching will introduce some additional delays within the space segment of the network. For fast packet or cell switched satellite networks, the switching delay is negligible compared to the propagation delay, but the buffering delay can be significant. Buffering also results in delay variations due to the bursty nature of ATM traffic.

The major focus of this paper includes:

1. The development of an end-to-end satellite network delay model.

2. Simulation and analysis of the buffering requirements of the satellite network for TCP/IP traffic over the UBR service category

In this paper, we have assumed that all processing is performed at the ground terminals with the help of the NCC. The simulations of buffer requirements estimate the buffers needed at the ground stations, and assume that no on-board processing or buffering is performed. However, the delay model presented in the next section is applicable for on-board processing and switching systems.

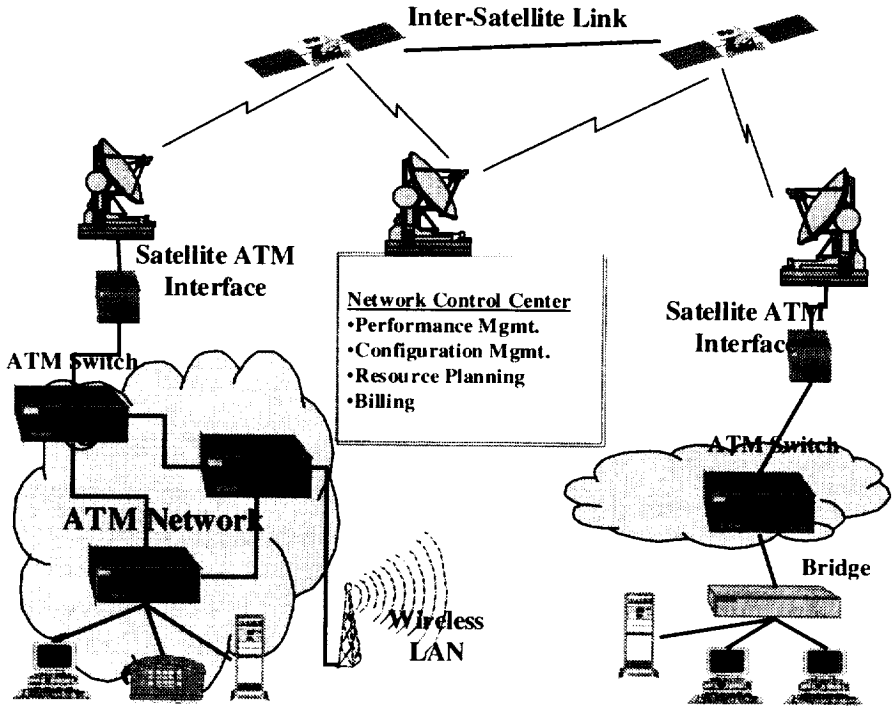


Figure 1: Satellite-ATM network model

4 Satellite Network Delay Model

In this section, we develop a simple delay model of a satellite network. This model can be used to estimate the end-to-end delay of both GEO and LEO satellite networks.

The end-to-end delay (D) experienced by a data packet traversing the satellite network is the sum of the transmission delay (t_t), the uplink (t_{up}) and downlink (t_{down}) ground segment to satellite propagation delays, the inter-satellite link delay (t_i), the on-board switching and processing delay (t_s) and the buffering delay (t_q). The inter-satellite, on-board switching, processing and buffering delays are cumulative over the path traversed by a connection. In this model, we only consider the satellite component of the delay. The total delay experienced by a packet is the sum of the delays of the satellite and the terrestrial networks. This model does not incorporate the delay variation experienced by the

cells of a connection. The delay variation is caused by orbital dynamics, buffering, adaptive routing (in LEOs) and on-board processing. Quantitative analysis of delay jitter in satellite systems is beyond the scope of this study. The end-to-end delay (D) is given by:

$$D = t_t + t_{up} + t_i + t_{down} + t_s + t_q$$

Transmission delay: The transmission delay (t_t) is the time taken to transmit a single data packet at the network data rate.

$$t_t = \frac{packet\_size}{data\_rate}$$

For broadband networks with high data rates, the transmission delays are negligible in comparison to the satellite propagation delays. For example, a 9180 byte TCP packet is transmitted in about 472 microseconds. This delay is much less than the propagation delays in satellites.

Propagation delay: The propagation delay for the cells of a connection is the sum of the following three quantities:

1. The source ground terminal to source satellite propagation delay (t_{up})
2. The Inter-satellite link propagation delays (t_i)
3. The destination satellite to destination ground terminal propagation delay (t_{down})

The *uplink and downlink satellite-ground terminal propagation delays* (t_{up} and t_{down} respectively) represent the time taken for the signal to travel from the source ground terminal to the first satellite in the network (t_{up}), and the time for the signal to reach the destination ground terminal from the last satellite in the network (t_{down}).

$$t_{up} = \frac{source\_satellite\_dist}{speed\_of\_signal}$$

$$t_{down} = \frac{dest\_satellite\_dist}{speed\_of\_signal}$$

The *inter-satellite link delay* (t_i) is the sum of the propagation delays of the inter-satellite links (ISLs) traversed by the connection. Inter-satellite links (crosslinks) may be *in-plane* or *cross-plane* links. In-plane links connect satellites within the same orbit plane, while cross-plane links connect satellites in different orbit planes. In GEO systems, ISL delays can be assumed to be constant over a connection's lifetime because GEO satellites are almost stationary over a given point on the earth, and with respect to one another. In LEO constellations, the ISL delays depend on the orbital radius, the number of satellites-per-orbit, and the inter-orbital distance (or the number of orbits). Also, the ISL delays change over the life of a connection due to satellite movement and adaptive routing techniques in LEOs. As a result, LEO systems can exhibit a high variation in ISL delay.

$$t_i = \frac{\sum ISL\_lengths}{speed\_of\_signal}$$

Buffering delay: Buffering delay (t_q) is the sum of the delays that occur at each hop in the network due to cell queuing. Cells may be queued due to the bursty nature of traffic, congestion at the queuing points (earth stations and satellites), or due to media access control delays. Buffering delays depend on the congestion level, queuing and scheduling policies, connection priority and ATM service category. CBR and real time VBR connections suffer minimum buffering delays because they receive higher priority than the non-real time connections. Cells from ABR and UBR connections could suffer significant delay at each satellite hop during periods of congestion.

Switching and processing delays: The data packets may incur additional delays (t_s) at each satellite hop depending on the amount of on-board switching and processing. For high data rate networks with packet/cell switching, switching and processing delays are negligible compared to the propagation delays.

5 Propagation Delay Model

In this section, we present a propagation delay model for satellite networks. The GEO model is fairly simple due to the stationary nature of GEO satellites, and the small number of satellites needed to cover the earth. The LEO model assumes a circular multi-orbit constellation with evenly spaced orbits and evenly spaced satellites within the orbits.

5.1 GEO Propagation Delay Model

GEO systems are at an altitude of about 36,000 km above the equator. For GEOs, t_{up} and t_{down} can be approximated to about 125 ms each for ground terminals near the equator. Inter satellite propagation delays are stable and depend on the number of satellites in the constellation. As few as three GEOs are sufficient to cover the earth. Table 1 lists the inter-satellite link distances and propagation delays for GEO systems with N satellites evenly spaced around the equatorial plane. For ground terminals farther away from the equator, the propagation delay from ground station to ground station through a single satellite is about 275 ms.

Table 1 : GEO Inter Satellite Delays

Number of Satellites (N)	Inter-Satellite Link Distance (km)	Inter-Satellite Link Delay (ms)
3	73,030	243
4	59,629	199
5	49,567	165
6	42,164	141
7	36,589	122
8	32,271	108
9	28,842	96
10	26,059	87
11	23,758	79
12	21,826	73

5.2 LEO Propagation Delay Model

In this section, we provide a simple model for the propagation delays of LEO systems. This model calculates the total propagation delay from source ground terminal to the destination ground terminal, through the LEO network. A LEO geometry and network topology model is used to determine the total number of satellites and the total propagation delay for communication between two ground points. The model uses the following information.

- Number of orbit planes
- Number of satellites per orbit plane
- Satellite altitude
- Orbit plane inclination angle[1]
- Ground terminal coordinates

5.2.1 LEO Orbital Model

Low Earth Orbit (LEO) communication satellites are arranged in a constellation in the following manner. The satellites are organized into a set of *number_of_orbit_planes* orbit planes, each containing *number_of_sats_per_plane* satellites. The orbits are assumed to be circular and to have a common, high inclination angle *(inclination)*. The inclination angle, combined with the electronic-horizon reach of the satellites directly determines the range of latitudes for which the system can provide service. The satellites within a given orbit plane are evenly spaced by using a delta anomaly between in-plane satellite orbits:

$$delta\_anomaly = \frac{360^\circ}{number\_of\_sats\_per\_plane}$$

The orbit planes are approximately evenly spaced about the earth polar axis by using a delta right ascension and a correction term between orbit planes:

$$delta\_right\_ascension = \frac{180^\circ}{number\_of\_orbit\_planes} + RA\_correction$$

Spreading the right ascension of the orbit planes over 180 degrees means that the satellites in adjacent planes are roughly traveling in parallel, with the exception that between the last and first planes, the satellites are traveling in opposite directions. The interval between the last and first orbit plane is called the "seam" of the constellation. The fact that the satellites in the last and first orbit planes of the constellation are travelling in opposite directions means that any cross-plane links spanning the seam will have to change connectivity to a different satellite every few minutes. This will result in frequent handovers causing additional delays and delay variations.

[1] Inclination angle is the angle made by the satellite orbital plane with the equatorial plane.

The *RA_correction* term in the previous formula is necessary for the following reason. LEO communication constellations use inclination angles of less than 90 degrees. Because of this, the last orbit plane tends to tilt in the opposite direction as the first orbit plane by roughly twice the complement of the inclination angle (i.e., 2×(90-*inclination*)). Without the correction term for the *delta_right_ascension*, a "hole" results in the ground coverage of the constellation in the two areas of the earth where the seam orbit-planes are tilting away from each other. In the opposite hemisphere of the two holes, the seam orbit-planes are tilting towards each other, resulting in overlapping, or redundant, ground coverage. Trade-offs can be made between how much of the serviced latitude range will be provided continuous, uninterrupted service, and the amount of redundant coverage suffered. The model described here currently uses the following simple correction term.

$$RA\_correction = \frac{1.5 \times (90^\circ - inclination)}{number\_of\_orbit\_planes}$$

The inter-plane satellites are phased by about one-half of the in-plane satellite spacing. This staggered phasing provides a more optimal and uniform coverage pattern, and maximizes inter-plane satellite distances near the extreme latitudes where the orbit planes cross. The current model uses the following delta inter-plane phasing.

$$Delta\_inter\_plane\_phasing = 0.5 \times delta\_anomaly + delta\_right\_ascension \times \sin(90 - inclination)$$

The model uses an Earth Centered Inertial (ECI) right -handed coordinate system. The first and second axes lie on the equatorial plane with the first axis aligned with the prime meridian. The third axis is aligned with the earth's polar axis pointing north. The first satellite of the first orbit plane (satellite(1,1)) is arbitrarily placed at latitude, longitude coordinates of (0,0), giving this satellite a position vector in ECI coordinates of (r, 0, 0), where r is the sum of the equatorial earth radius and the satellite altitude. The position vectors of the remaining satellites are obtained by an appropriate series of rotations of the satellite(1,1) position vector involving the angles described above.

5.2.2 LEO Route Calculation

We use this model to calculate routes and propagation delays across the satellite constellation. We first use the above procedure to create a constellation pattern providing continuous ground coverage for most of the latitude range covered by the constellation. Circular orbits with staggered inter-plane phasing are assumed as described above. Each satellite is assumed to have four crosslinks (inter-satellite links) providing connectivity to the in-plane satellites immediately leading and following, and to the nearest satellites in the two adjacent orbit planes -- in navigational terms, these would be the fore, aft, port, and starboard satellites. Cross-plane connectivity constraints in the area of extreme latitudes or across the constellation seam (i.e., where satellites in the last orbit plane and the first orbit plane are traveling in opposite relative directions) are not considered. Anticipatory routing to reduce hand-offs and routing around congested paths is not considered.

The following simple algorithm is used to determine the route between two ground points through the satellite constellation. One of the ground points is designated as the source node and the other ground point is assigned as the destination node. The satellite nearest to the source node, and the satellite nearest to the destination node are first determined. This assumes minimal redundant satellite ground coverage, which is usually the case for LEO communication systems. Starting at the satellite nearest the source node, of the satellites with which it has connectivity, the satellite nearest to the destination node's satellite is selected. The process is repeatedly applied at each selected satellite, with backtracking precluded, until the destination node's satellite is reached. The algorithm then counts the number of satellites in the end-to-end, source-to-destination path. The distances between successive path-nodes, beginning at the source terminal and ending at the destination terminal, are computed, converted to link propagation delays by dividing by the speed of light, and accumulated to provide the path end-to-end propagation delay. The number of satellites in the route path and the total propagation delay are then reported. While the routing algorithm just described is strictly geometry based and only locally optimal, of the limited cases examined thus far, the results appear to be generally coincident with a globally optimal solution.

The model can also generate three-dimensional orthographic-projection displays showing satellite orbits, satellite positions, cross-links, ground terminal positions, path-links followed, and earth model from any desired viewing direction. Figure 2 shows an example path of a 6-plane, 11-satellites per plane LEO system path from Los Angeles to London. The associated configuration parameters are given in Table 2. Table 3 shows the resultant number of path-satellites, the individual link delays, and the total end-to-end delay for the example. Table 4 shows the end-to-end propagation delays for a 6-plane, 11-satellites per plane constellation, between 10 cities of the world ranked by Gross Domestic Product. Table 5 shows the number of satellites in the path between the same set of cities. Table 6 and Table 7 show the same information for a 12-plane, 24-satellites per plane constellation at an altitude of 1400 km and an inclination angle of 82 degrees.

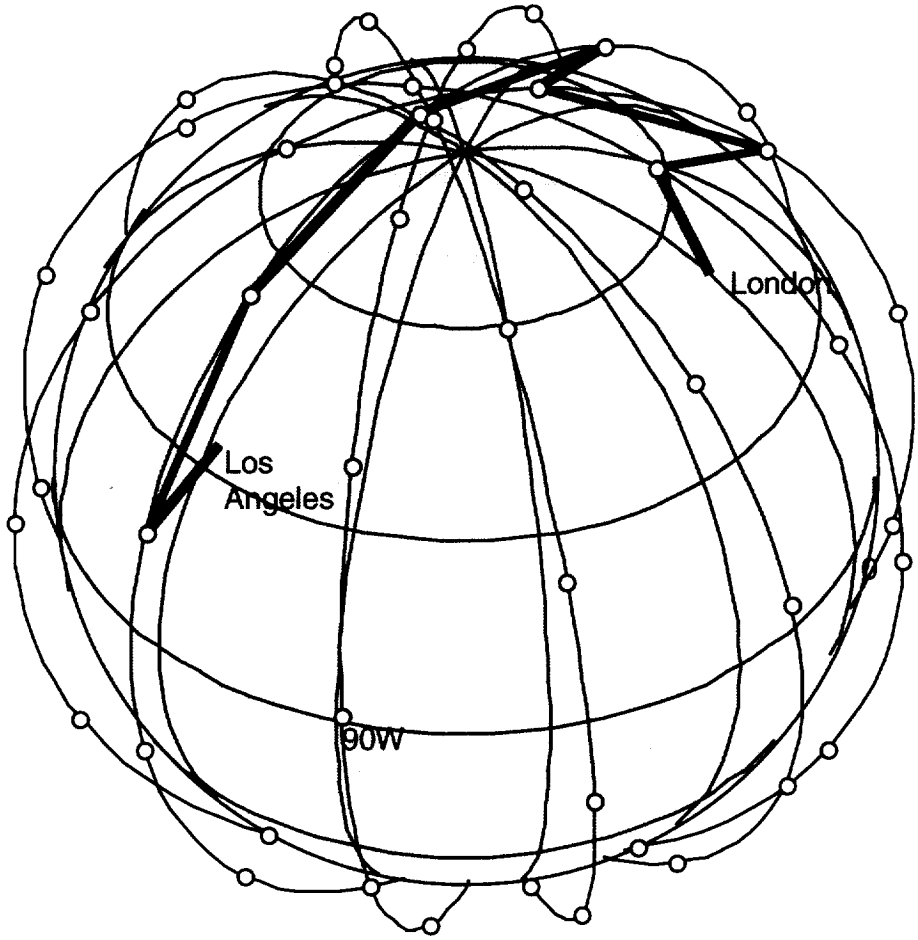


Figure 2: Example path through LEO constellation

Table 2: LEO configuration parameters

Orbit Planes	6
Sats Per Plane	11
Total Sats	66
Altitude(km)	780
Inclination(deg)	86

Table 3: LEO propagation delays

Source	Los Angeles
Destination	London
Satellites In Path	7
Propagation Delays(ms)	
Uplink	5.87
Downlink	6.37
ISL 1	13.44
ISL 2	13.44
ISL 3	13.44
ISL 4	7.80
ISL 5	13.44
ISL 6	9.63
Total Prop. Delay	**83.45**

Table 4: Total propagation delays in milliseconds in city-to-city path for 6x11 constellation

	New York	Tokyo	Paris	London	Seoul	Los Angeles	Toronto	Mexico City	Sydney	Chicago
New York	11									
Tokyo	58	13								
Paris	60	60	13							
London	39	47	26	13						
Seoul	62	37	83	68	11					
Los Angeles	24	71	69	83	71	12				
Toronto	10	57	54	38	63	23	9			
Mexico City	26	73	51	61	79	39	25	14		
Sydney	62	37	91	71	36	49	61	77	7	
Chicago	8	56	53	36	61	22	7	23	59	6

Table 5: Number of satellites in city-to-city path for 6x11 constellation

	New York	Tokyo	Paris	London	Seoul	Los Angeles	Toronto	Mexico City	Sydney	Chicago
New York	1									
Tokyo	5	1								
Paris	5	5	1							
London	4	4	2	1						
Seoul	5	3	7	6	1					
Los Angeles	2	6	6	7	6	1				
Toronto	1	5	5	4	5	2	1			
Mexico City	2	6	4	5	6	3	2	1		
Sydney	5	3	7	6	3	4	5	6	1	
Chicago	1	5	5	4	5	2	1	2	5	1

Table 6: Total propagation delays in milliseconds in city-to-city path for 12x24 constellation.

	New York	Tokyo	Paris	London	Seoul	Los Angeles	Toronto	Mexico City	Sydney	Chicago
New York	10									
Tokyo	57	11								
Paris	77	109	10							
London	78	110	11	12						
Seoul	58	24	75	76	11					
Los Angeles	41	57	99	100	59	10				
Toronto	10	57	73	74	58	44	11			
Mexico City	24	67	87	88	68	30	24	10		
Sydney	92	52	89	89	53	115	92	113	10	
Chicago	22	54	83	84	57	30	23	23	101	10

Table 7: Number of satellites in city-to-city path for 12x24 constellation

	New York	Tokyo	Paris	London	Seoul	Los Angeles	Toronto	Mexico City	Sydney	Chicago
New York	1									
Tokyo	8	1								
Paris	10	15	1							
London	10	15	1	1						
Seoul	8	3	12	12	1					
Los Angeles	6	9	15	15	9	1				
Toronto	1	8	10	10	8	6	1			
Mexico City	3	10	12	12	10	4	3	1		
Sydney	14	7	12	12	7	19	14	18	1	
Chicago	3	8	12	12	8	4	3	3	13	1

5.3 Delay Variation Characteristics

Although LEO networks have relatively smaller propagation delays than GEO networks, the delay variation in LEOs can be significant. The delay variation in LEO systems can arise from several factors:

1. **Handovers:** The revolution of the satellites within their orbits causes them to change position with respect to the ground terminals. As a result, the ground terminal must handover the connections from the satellite descending below the horizon to the satellite ascending from the opposing horizon. Based on the velocity, altitude and the coverage of the satellites, it is estimated that call handovers can occur on an average of every 8 to 11 minutes [IQTC97]. The handover procedure requires a state transfer from one satellite to the next, and will result in a change in the delay characteristic of the connection at least for a short time interval. If the satellites across the seam of the constellation are communicating via crosslinks, the handover rate is much more frequent because the satellites are travelling in opposite directions.

2. **Satellite Motion:** Not only do the satellites move with respect to the ground terminal, they also move relative to each other. When satellites in adjacent orbits cross each other at the poles, they are now traveling in opposite sides of each other. As a result, calls may have to be rerouted accordingly resulting in further changes in delays.

3. **Buffering and Processing:** A typical connection over a LEO system might pass through several satellites, suffering buffering and processing delays at each hop. For CBR traffic, the buffering delays are small, but for bursty traffic over real time VBR (used by video applications), the cumulative effects of the delays and delay variations could be large depending on the burstiness and the amount of overbooking in the network.

4. **Adaptive Routing:** Due to the satellite orbital dynamics and the changing delays, most LEO systems are expected to use some form of adaptive routing to provide end-to-end connectivity. Adaptive routing inherently introduces complexity and delay variation. In addition, adaptive routing may result in packet reordering. These out of order packets will have to be buffered at the edge of the network resulting in further delay and jitter.

GEO systems exhibit relatively stable delay characteristics because they are almost stationary with respect to the ground terminals. Connection handovers are rare in GEO systems and are mainly due to fault recovery reasons. As a result, there is a clear trade-off between delay and jitter characteristics of GEO and LEO systems, especially for interactive real-time applications.

6 Buffering Delays for TCP/IP over Satellite-ATM ABR and UBR Service Classes

The majority of Internet traffic is TCP/IP based data traffic. It is thus important to assess the buffering characteristics for TCP/IP applications over satellite-ATM networks. Most TCP/IP data applications will use the ABR or UBR service categories in ATM networks. The maximum buffering delay can be calculated from an estimate of the buffer size at each queuing point in the connection's path:

$$Buffering\_delay \leq \frac{Buffer\_size}{Buffer\_drain\_rate}$$

The buffer drain rate is the rate at which cells are serviced from the buffer. This rate depends on the link capacity, the scheduling policy, and other higher priority traffic on the link. The queuing points in the network must have sufficient buffer size to ensure good performance of TCP/IP applications. In this section, we present a simulation model to calculate the buffer requirements for TCP/IP traffic over ATM networks for the ABR and UBR service categories. Section 6.1 outlines some known results on buffer requirements of TCP over ABR [SHIV98]. Section 6.2 describes a simulation model for TCP/IP over satellite-ATM UBR, and presents simulation results to estimate the buffer requirements for TCP/IP file transfer traffic. The estimates of buffer requirements are then used to calculate the queuing delays at each queuing point in the connection's path.

6.1 Buffer Requirements for TCP/IP over ABR

An analysis of the buffer requirements of TCP/IP over ABR has been conducted in [SHIV97] and [SHIV98]. ABR uses a rate based, closed loop feedback control model for congestion control. As a result, the performance of the ABR service depends on the ABR congestion control scheme used in the switch, and on the ABR source-end-system parameters. In general, a good ABR switch scheme should be able to control queues within the ATM network. The ERICA+ (Explicit Rate Indication Congestion Avoidance +) scheme [ERIC97]has been specified as a sample scheme by the ATM Forum. [SHIV97] and [SHIV98] show that for the ERICA+ scheme, the buffer requirements for an ABR switch to ensure zero packet loss for TCP/IP traffic over ABR can be bounded by a constant multiple of the round trip propagation delay from the ABR end-system or virtual end-system to the bottleneck ABR node in the network. The delay is called the *feedback delay* of the network, and it specifies the time taken for the effect of the bottleneck feedback to be seen by the network. The feedback delay at a queuing point can be restricted to the round trip propagation delay from the previous end-system by implementing Backward Explicit Congestion Notification (BECN) or Virtual Source/Virtual Destination (VS/VD) at the satellite nodes. For TCP/IP file transfer traffic, the buffer requirements are proportional only to the feedback delay, and are independent of the number of TCP sources and other background traffic in the ATM network. Thus, TCP connections can be transported through a finite buffer ABR network with zero packet loss.

6.2 Buffer Requirements for TCP/IP over UBR

Most ATM networks are expected to be implemented as backbone networks within an IP based Internet where edge devices separate ATM networks from IP networks. Currently, IP networks do not support the rate based flow control mechanisms used by ABR. The above studies have shown that for ATM in the backbone, the buffer requirements of nodes at the periphery of the ATM network (edge devices) for TCP/IP traffic are comparable to buffer requirements for TCP/IP with UBR. Moreover, since TCP has its own flow and congestion control mechanisms, many TCP/IP connections are expected to use the UBR service. As a result, it is important to assess the buffer sizes (and hence delays) at UBR queuing points in a satellite network.

In this subsection, we present a simulation model for calculating the buffer requirements for satellite-UBR networks to efficiently support TCP/IP traffic. We present results of SACK TCP throughput over satellite-UBR for various satellite latencies, buffer sizes and number of sources.

6.2.1 Simulation Model

Figure 3 shows the basic network configuration used in the paper to assess buffer requirements at a single bottleneck node. In the figure, the switches represent the earth stations that connect to the satellite constellation. The earth stations interface the terrestrial network with the satellite network. In general, the satellite network model may include on-board processing and queuing. In the results stated in this section, no on-board processing or queuing is performed. The bottleneck node is the earth station at the entry to the satellite network. As a result, in our experiments, no queuing delays occur in the satellite network. All processing and queuing are performed at the earth stations. The goal of this study is to assess the buffer requirements of the bottleneck node (in this case, the earth station) for good TCP/IP performance.

All simulations use the N source configuration shown in the figure. All sources are identical and persistent TCP sources. The TCP layer always sends a segment as long as it is permitted by the TCP window. Moreover, traffic is unidirectional so that only the sources send data. The destinations only send ACKs. The TCP delayed acknowledgement timer is deactivated, and the receiver sends an ACK as soon as it receives a segment. TCP with selective acknowledgments (SACK TCP) is used in our simulations. All link bandwidths are 155.52 Mbps, and peak cell rate at the ATM layer is 149.7 Mbps. This accounts for a SONET like overhead in the satellite component of the network.

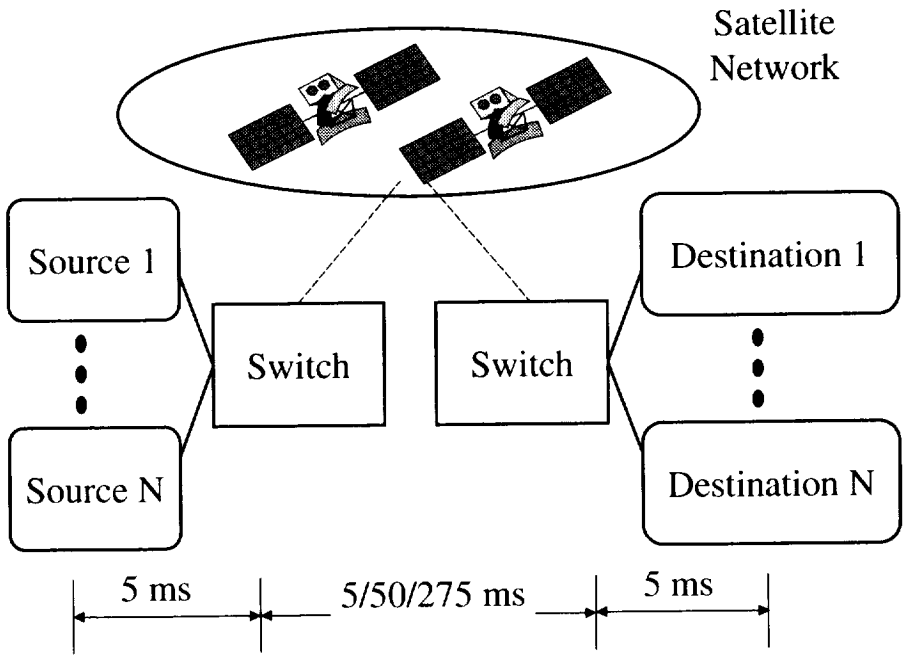


Figure 3: Simulation model for TCP/IP over UBR

The following parameters are used to assess the buffer requirements:

Latency: Our primary aim is to study the buffer requirements for long latency connections. A typical latency from earth station to earth station for a single LEO hop is about 5 ms. The latencies for multiple LEO hops can easily be 50 ms or more from earth station to earth station. GEO latencies are typically 275 ms from earth station to earth station for earth stations that are not on the equator. We study these three latencies (5 ms, 50 ms, and 275 ms) with various number of sources and buffer sizes. The link delays between the switches and the end systems are 5 ms in all configurations. This results in round trip propagation delays (RTT) of 30 ms, 120 ms and 570 ms respectively.

Number of sources: To ensure that the recommendations are scalable and general with respect to the number of connections, we will use configurations with 5, 15 and 50 TCP connections on a single bottleneck link. For single hop LEO configurations, we use 15, 50 and 100 sources.

Buffer size: This is the most important parameter of this study. The goal is to estimate the smallest buffer size that results in good TCP performance, and is scalable to the number of TCP sources. The values chosen for the buffer size are approximately:

$$Buffer\_size = 2^{-k} \times RTT \times bottleneck\_link\_data\_rate, k = -1..6$$

i.e., we choose 2, 1, 0.5, 0.25, 0.125, 0.0625, 0.031 and 0.016 multiples of the round trip delay-bandwidth product of the TCP connections. The resulting buffer sizes (in cells) used in the earth stations are as follows:

- *Single LEO:* 375, 750, 1500, 3000, 6000, 12000 (=1 RTT), 24000 and 36000 cells.
- *Multiple LEO:* 780, 1560, 3125, 6250, 12500, 25000, 50000 (=1 RTT), and 100000 cells.
- *GEO:* 3125, 6250, 12500, 25000, 50000, 100000, 200000 (=1 RTT), and 400000 cells.

We plot the buffer size against the achieved TCP throughput for different delay-bandwidth products and number of sources. The asymptotic nature of this graph provides information about the optimal buffer size for the best performance.

Buffer allocation policy: We use a per-VC buffer allocation policy called *selective drop* [GOY97a][STAL98]to fairly allocate switch buffers to the competing TCP connections.

End system policies: We use an enhanced version of TCP called SACK TCP [RF2018], for this study. SACK TCP improves performance by using selective acknowledgements for retransmission. Further details about our SACK TCP implementation can be found in [GOY97a]. The maximum value of the TCP receiver window is 600000 bytes, 2500000 bytes and 8704000 bytes for single hop LEO, multiple hop LEO and GEO respectively. These window sizes are obtained using the TCP window scaling option, and are sufficient to achieve full utilization on the 155.52 Mbps links. The TCP maximum segment size is 9180 bytes. This conforms to the segment size recommended for TCP connections over long latency connections. The TCP timer granularity is set to 100 ms. This value limits the time taken for retransmissions to multiples of 100 ms. The value is chosen to balance the attainable throughput with the limitations of the TCP RTT measurement algorithm. With large granularity, TCP could wait a long time before detecting packet loss, resulting in poor throughput. Finer granularity of the retransmission timer leads to false timeouts even with a small variation in the measured RTT values.

6.2.2 Performance Metrics

The performance of TCP over UBR is measured by the *efficiency* and *fairness* which are defined as follows:

$$Efficiency = \frac{\sum_{i=1}^{N} x_i}{x_{\max}}$$

Where x_i is the throughput of the i^{th} TCP connection, x_{max} is the maximum TCP throughput achievable on the given network, and N is the number of TCP connections. The TCP throughputs are measured at the destination TCP layers. Throughput is defined as the total number of bytes delivered to the destination application, divided by the total simulation time. The results are reported in Mbps. The maximum possible TCP throughput (x_{max}) is the throughput attainable by the TCP layer running over UBR on a 155.52 Mbps link. For 9180 bytes of data (TCP maximum segment size), the ATM layer receives 9180 bytes of data + 20 bytes of TCP header + 20 bytes of IP header + 8 bytes of LLC header + 8 bytes of AAL5 trailer. These are padded to produce 193 ATM cells. Thus, each TCP segment results in 10229 bytes at the ATM layer. From this, the maximum possible throughput = 9180/10229 = 89.7% = 135 Mbps approximately on a 155.52 Mbps link.

$$Fairness = \frac{\sum_{i=1}^{N}\left(\frac{x_i}{e_i}\right)^2}{N \times \left(\sum_{i=1}^{N}\frac{x_i}{e_i}\right)^2}$$

Where e_i is the expected throughput of the i^{th} TCP connection. Both metrics lie between 0 and 1, and the desired values of efficiency and fairness are close to 1 [JAIN91]. In the symmetrical configuration presented above,

$$e_i = \frac{x_{\max}}{N}$$

and the fairness metric represents a equal share of the available data rate. For more complex configurations, the fairness metric specifies max-min fairness [JAIN91].

6.2.3 Simulation Results

Figures 4, 5, and 6 show the resulting TCP efficiencies for the 3 different latencies. Each point in the figure shows the efficiency (total achieved TCP throughput divided by maximum possible throughput) against the buffer size used. Each figure plots a different latency, and each set of points (connected by a line) in a figure represents a particular value of N (the number of sources).

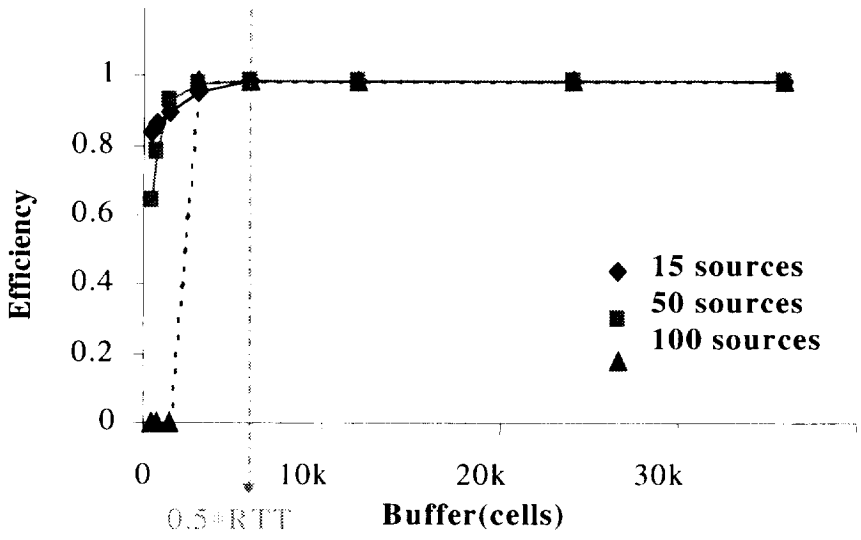


Figure 4: TCP/IP UBR buffer requirements for single hop LEO

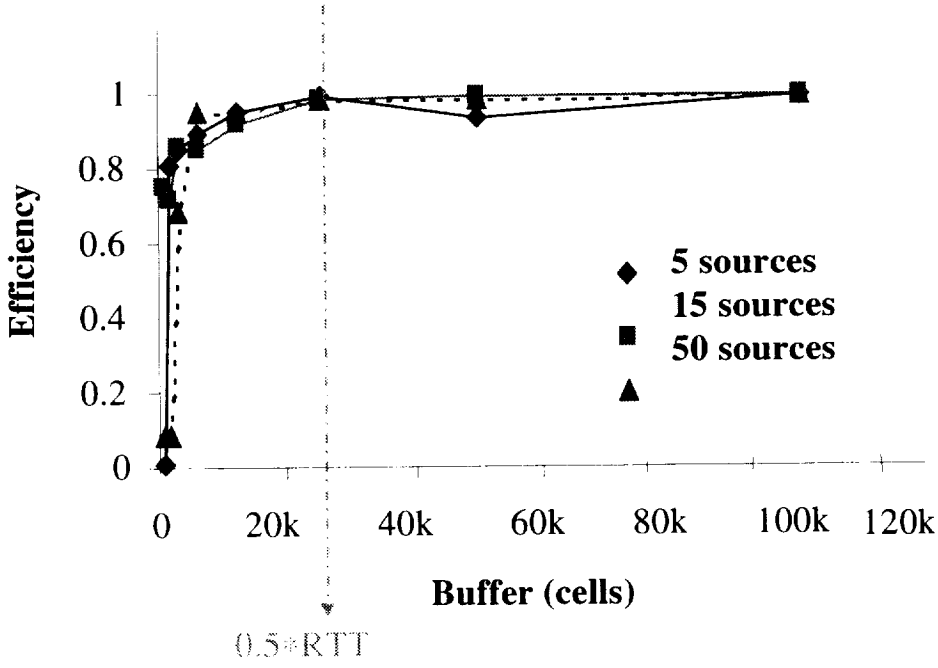


Figure 5: TCP/IP UBR buffer requirements for multiple hop LEO

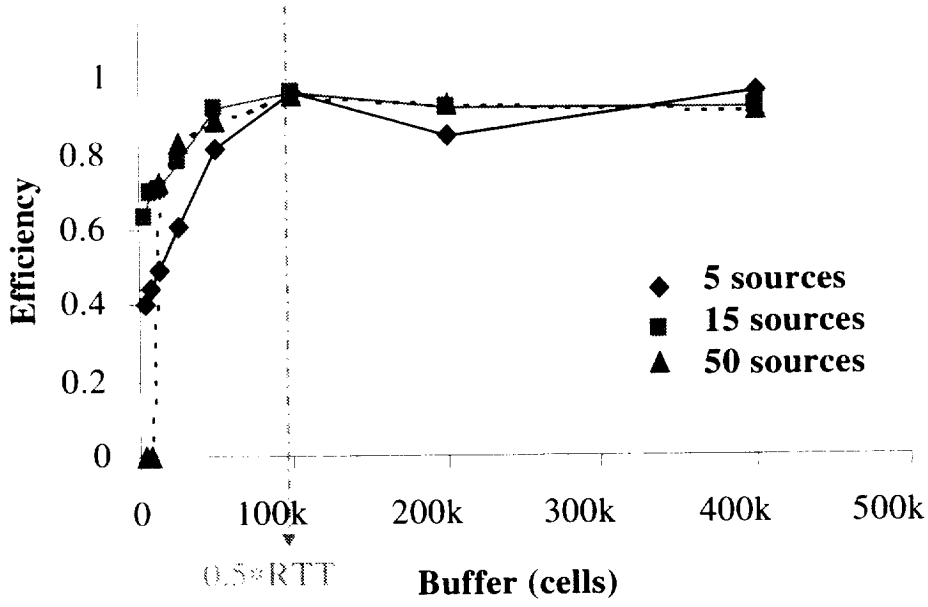


Figure 6: TCP/IP UBR buffer requirements for single hop GEO

The following conclusions can be drawn from the figures:

For very small buffer sizes, (0.016RTT, 0.031RTT, 0.0625RTT), the resulting TCP throughput is very low. In fact, for a large number of sources (N=50), the throughput is sometimes close to zero. For small

buffer sizes, the performance of TCP/IP deteriorates with increasing number of sources. This is because more TCP packets are dropped from each connection causing TCP timeout and retransmissions. This results in decreased throughput.

For moderate buffer sizes (less then 1 round trip delay times bandwidth), TCP throughput increases with increasing buffer sizes. TCP throughput asymptotically approaches the maximal value with further increase in buffer sizes.

TCP performance over UBR for sufficiently large buffer sizes is scalable with respect to the number of TCP sources. The throughput is never 100%, but for buffers greater than 0.5xRTT, the average TCP throughput is over 98% irrespective of the number of sources.

The knee of the buffer versus throughput graph is more pronounced for larger number of sources. For a large number of sources, TCP performance is very poor for small buffers, but jumps dramatically with sufficient buffering and then stays about the same. For smaller number of sources, the increase in throughput with increasing buffers is more gradual. With sufficient buffers, TCP dynamics enable the connections to share buffer space and link capacity. TCP windows are controlled by the rate at which ACKs are received by the source. The total amount of unacknowledged data is thus controlled by the connection's bandwidth delay product. All this data can be queued at a single queuing point within the network. As a result, each queuing point must have sufficient buffers to support one delay-bandwidth product worth of TCP data so that it can ensure minimal loss.

For large round trip delays, and a small number of sources, a buffer of 1 RTT or more can result in reduced throughput. This is because of the variability in the connection's measured RTT due to buffering delays. When the queuing delay is of the order of the round trip propagation delay, the retransmission timeout values become highly variable. During the initial phase (startup exponential increase), when the queuing delays are small, the timeout value corresponds to the propagation RTT. When the windows increase to fill the switch buffer, the queuing delay increases to about 1 RTT (for a buffer size of about 1 RTT), and packets at the tail of the queue get dropped. Retransmitted packets are sent out after 3 duplicate ACKS are received. However, these retransmitted packets are queued behind a whole RTT worth of queues at the bottleneck switch. As a result, before the sender gets an ACK for retransmitted packets, a timeout occurs, and slow start is incurred. At this point, the sender starts to retransmit from the last unacknowledged segment, but soon receives an ACK for that segment (because the segment was not really lost, but the delay was incorrectly estimated). The loss in throughput occurs during to the time lost in waiting for the retransmission timeout. With smaller buffers, the variability in the RTT is smaller, and false timeouts do not occur. Also, the negative effects of large buffers is not seen in the single hop LEO configuration, because the RTT in this case is much smaller than the timer granularity. As a result, even a high queuing delay is not enough to exceed the minimum timeout value.

The simulation results show that TCP sources with a good per-VC buffer allocation policy like selective drop, can effectively share the link bandwidth. A buffer size of about 0.5RTT to 1RTT is sufficient to provide over 98% throughput to infinite SACK TCP traffic for long latency networks and a large number of sources. This buffer requirement is independent of the number of sources. The fairness in the throughputs measured by the fairness index is high due to the selective drop policy [KOTA97].

7 Delay Analysis: Case Study

In this section, we evaluate the end-to-end delay performances of LEO and GEO systems presented in this paper. We consider end-to-end the delay of a connection from New York to Paris. The connection is serviced by a single GEO satellite. For the 6x11 LEO constellation, the connection passes through 5 satellites. For the 12x24 constellation, the connection passes through 10 satellites. The corresponding propagation delays (up-down plus inter-satellite link) can be found from tables 1, 4 and 6. The one way propagation delays from ground station to ground station are 60 ms, 77 ms and 250 ms for the 6x11 LEO, 12x24 LEO and GEO networks respectively. We assume a TCP/IP application over the UBR service category, similar to the one simulated in the previous section. We consider a single queuing point as before, as well as multiple queuing points possibly on-board the satellites. As discussed in the previous section, each queuing point including the ground terminal could have a buffer size of about 0.5RTT for the connection. The resulting buffer sizes are 60 ms for the 6x11 network, 77 ms for the 12x24 network, and 250 ms for the GEO network. This means that a TCP connection can suffer a delay of 60 ms, 77 ms or 250 ms at each queuing point in the respective networks.

Table 8: New York to Paris: Delay Analysis

Delay	**GEO (ms)** **1 satellite**	**6x11 LEO (ms)** **5 satellites**	**12x24 LEO (ms)** **10 satellites**
Transmission	Negligible	Negligible	Negligible
Propagation (up+down+ISL)	250	60	77
Switching and Processing	Negligible	Negligible	Negligible
Buffering (N queuing points)	0 to N*250	0 to N*60	0 to N*77
Total Delay	**250 to 500**	**60 to 420**	**77 to 924**

Table 8 lists the individual and total delays for the connection. The transmission and processing delays on a 155.52 Mbps link are small compared to the propagation delay and is thus ignored. From the table, it can be seen that the minimum delay for GEO systems is large in comparison to LEO systems. However, for TCP/IP over UBR traffic, the maximum delays for LEOs are comparable and even higher for the 12x24 system than the GEO system. Moreover, TCP/IP RTT measurement algorithms might experience large variations in delay in LEO systems.

8 Summary

In this paper we presented a model to analyze the delay performance of GEO and LEO systems. We first presented a satellite-ATM network architecture model for QoS guarantees over satellite systems. The architecture presents a trade-off between the on-board switching/processing features and the complexity of the satellite communication systems. The end-to-end delay of a connection passing through a satellite constellation consists of the transmission delay, the uplink and downlink

propagation delays, the inter-satellite link propagation delays, the satellite switching and processing delays, and the buffering delays. The uplink and downlink propagation delays are much larger in GEO systems than LEO systems because of the higher altitude of the GEO satellites. However, LEO systems can have high delay variations due to orbital dynamics, and connection handovers. The buffering delay for TCP/IP traffic depends on the buffers at each queuing point in the network. The per-hop buffering delay for TCP/IP over ATM-UBR can be about 0.5RTT of the TCP connection. We presented case studies and calculated end-to-end delays of a sample connection from New York to Paris, and concluded that while GEO systems have a large propagation delay, buffering delay can be significant in both GEO and LEO networks.

We have not presented a quantitave analyis of the delay variation experienced by LEO connections. This analysis will lead to greater insights into the feasibility of using satellite networks to support voice, video and data services. A robust technique is needed to mitigate the effect of long delay paths in TCP connections. Protocols like the TCP spoof protocol, or the ABR virtual source / virtual destination option (VS/VD) need to be studied for their feasibility. Optimal routing algorithms in LEOs are also a topic of further study.

9 References

[AKYL97] Ian F. Akyildiz, Seong-Ho Jeong, "Satellite ATM Networks: A Survey," IEEE Communications Magazine, July 1997, Vol 5.35. No. 7.

[ERIC97] S. Kalyanaraman, R. Jain, et. al, "The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks," Submitted to IEEE/ACM Transactions on Networking, November 1997, http://www.cis.ohio-state.edu/~jain/papers/erica.htm

[GOY97a] R. Goyal, R. Jain et. al., "Per-VC rate allocation techniques for ABR feedback in VS/VD networks," ATM Form/98-1086r1, February 1998.

[GOY97b] R. Goyal, R. Jain, Sastri Kota et.al., "Selective Acknowledgments and UBR+ Drop Policies to Improve TCP/UBR Performance over Terrestrial and Satellite Networks," Proceedings of IC3N'97, September 1997.

[I35696] ITU-T Recommendation I-356, "B-ISDN ATM Layer Cell Transfer Performance," Geneva, October, 1996.

[IQTC97] David Lucantoni, Patrick Reilly, "Supporting ATM on a Low-Earth Orbit Satellite System," http://www.isoquantic.com

[IT4B97] ITU-R 4B Preliminary Draft Recommendation, "Transmission of Asynchronous Transfer Mode (ATM) Traffic via Satellite," Geneva, January 1997.

[JAIN91] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling,* Wiley-Interscience, New York, NY, April 1991.

[KOT97b] Sastri Kota, Jerry Kallaus, Henry Huey, David Lucantoni, "Demand Assignment Multiple Access (DAMA) For Multimedia Services – Performance Results," Proceedings of Milcom'97, Monterey, CA, 1997.

[KOTA97] Sastri Kota, R. Goyal, Raj Jain, "Satellite ATM Network Architectural Considerations and TCP/IP Performance," Proceedings of the 3rd K-A Band Utilization Conference, 1997.

[PHAM97] C. Pham, S. Kota, R. Gobbi, "Performance of Concatenated Coding for Satellite ATM Networks," Document in preparation.

[PONZ97] C. Ponzoni, "High Data Rate On-board Switching," 3rd Ka-band Utilization Conference, September 13-18, 1997.

[RF2018] M. Mathis, J. Madhavi, S. Floyd, A. Romanov, "TCP Selective Acknowledgment Options," Internet RFC 2018, October 1996.

[SHIV97] S. Kalyanaraman, R. Jain, et. al., "Performance of TCP over ABR with self-similar VBR video background traffic over terrestrial and satellite ATM networks," ATM Forum 97-0177r2, April 1997.

[SHIV98] S. Kalyanaraman, R. Jain, et. al., "Performance and Buffering Requirements of Internet Protocols over ATM ABR and UBR Services," To appear, IEEE Computer Communications Magazine.

[STAL98] W. Stallings, *High-Speed Networks. TCP/IP and ATM Design Principles,* Prentice Hall, New Jersey, 1998.

[TCPS98] Mark Allman, Dan Glover, "Enhancing TCP Over Satellite Channels using Standard Mechanisms," IETF draft, February 1998, http://tcpsat.lerc.nasa.gov/tcpsat

[TM4096] ATM Forum, "ATM Traffic Management Specification Version 4.0," April 1996, ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0056.000.ps

[WUPI94] William Wu, Edward Miller, Wilbur Pritchard, Raymond Pickholtz, "Mobile Satellite Communications," Proceedings of the IEEE, Vol. 82, No. 9, September 1994.

****

ATM Forum Document Number: ATM_Forum/97-0616

****

Title: UBR Buffer Requirements for TCP/IP over Satellite Networks

****

Abstract: In this contribution, we present simulation results to assess buffer requirements for TCP/IP over satellite UBR networks. We perform experiments with both LEO and GEO satellite delays, for various buffer sizes and number of sources. We conclude that with sufficiently large buffers (0.5RTT or more) the performance of TCP-SACK over UBR with per-VC accounting is scalable with respect to the number of sources.

****

Source:
Rohit Goyal, Raj Jain, Sonia Fahmy, Bobby Vandalore, Shiv Kalyanaraman
Department of CIS, The Ohio State University (and NASA)
395 Dreese lab, 2015 Neil Ave,
Columbus, OH 43210-1277
Phone: 614-292-3989, Fax: 614-292-2911, Email: {goyal,jain}@cis.ohio-state.edu

Sastri Kota
Lockheed Martin Telecommunications/Astrolink
1272 Borregas Avenue,
Bldg B/551 O/GB - 70
Sunnyvale, CA 94089
Email: sastri.kota@lmco.com

Pradeep Samudra,
Broadband Network Lab
Samsung Electronics Co. Ltd.
Samsung Telecom America, Inc.
1130 E Arapaho, Richardson, TX 75081
email: psamudra@telecom.sna.samsung.com

****

Date: July 1997, Montreal

****

Distribution: ATM Forum Technical Working Group Members (AF-TM)

****

****

1 Introduction

Satellite communication systems play an important role in the integration of networks of various types and services. Satellite systems will be used for a wide range of applications and will play an important role in the future of the Global Information Infrastructure. The main advantages of satellite systems are the long range broadcasting ability, support of mobile systems, and potentially high available bandwidth. However, satellite systems have several inherent constraints. The resources of the satellite communication network, especially the satellite and the earth station have a high cost and must be used efficiently. *A crucial issue is that of the high end-to-end propagation delay of satellite connections.*

The ATM-UBR service category is relatively cheap to implement in switch hardware. As a result, switches can multiplex thousands of transport connections that use the UBR service for non-real time applications. On board satellite switches and switches at the earth stations fall into this category and are expected to multiplex a large number of transport connections over UBR virtual circuits.

Apart from interoperability issues, several performance issues need to be addressed before a transport layer protocol like TCP can satisfactorily work over UBR. Moreover, with an acknowledgment and timeout based congestion control mechanism (like TCP's), the performance is inherently related to the delay-bandwidth product of the connection. As a result, the congestion control issues for high bandwidth satellite networks can be somewhat different from those of LAN and WAN networks.

The performance optimization problem can be analyzed from two perspectives - network policies and end system policies. The network can implement a variety of mechanisms to optimize resource utilization, fairness and higher layer throughput. For UBR, these include enhancements like intelligent drop policies to improve utilization, some minimal per-VC accounting [1, 2] to improve fairness, and even minimum throughput guarantees to the higher layers.

At the end system, the transport layer can implement various congestion avoidance and control policies to improve its performance and to protect against congestion collapse. Several transport layer congestion control mechanisms have been proposed and implemented. The mechanisms implemented in TCP are slow start and congestion avoidance [5], fast retransmit and recovery [7], and selective acknowledgments [8]. Several others like forward acknowledgments [9] and negative acknowledgments [4] have been proposed as enhancements to timeout based schemes.

Studies have shown that small switch buffer sizes result in very low TCP throughput over UBR [2]. It is also clear, that the buffer requirements increase with increasing delay-bandwidth product of the connections (provided the TCP window can fill up the pipe). However, the studies have not quantitatively analyzed the effect of buffer sizes on performance. *As a result, it is not clear how the increase in buffers affects throughput, and what buffer sizes provide the best cost-performance benefits for TCP/IP over UBR.*

In this contribution, we present our simulation results to assess the buffer requirements for various delay-bandwidth products for TCP/IP over UBR.

2 Previous Work: TCP performance over UBR

In our previous work, we have studied TCP performance over the ATM-UBR service for LAN, WAN and satellite networks. In our studies, we have used an N-source symmetrical TCP configuration with unidirectional TCP sources. The performance of TCP over UBR is measured by the efficiency and fairness which are defined as follows:

$$\text{Efficiency} = (\text{Sum of TCP throughputs})/(\text{Maximum possible TCP throughput})$$

The TCP throughputs are measured at the destination TCP layers. Throughput is defined as the total number of bytes delivered to the destination application, divided by the total simulation time. The results are reported in Mbps.

The maximum possible TCP throughput is the throughput attainable by the TCP layer running over UBR on a 155.52 Mbps link. For 9180 bytes of data (TCP maximum segment size), the ATM layer receives 9180 bytes of data + 20 bytes of TCP header + 20 bytes of IP header + 8 bytes of LLC header + 8 bytes of AAL5 trailer. These are padded to produce 193 ATM cells. Thus, each TCP segment results in 10229 bytes at the ATM Layer. From this, the maximum possible throughput = 9180/10229 = 89.7% = 135 Mbps approximately on a 155.52 Mbps link (149.7 Mbps after SONET overhead).

$$\text{Fairness Index} = (\Sigma x_i)^2 / (\text{n} \times \Sigma x_i^2)$$

Where x_i = throughput of the ith TCP source, and n is the number of TCP sources. The fairness index metric applies well to our N-source symmetrical configuration.

In most cases, the performance of TCP over UBR has been poor. A summary of our previous results is presented below [2, 3]:

- TCP achieves maximum possible throughput when no segments are lost. To achieve zero loss for TCP over UBR, switches need buffers equal to the sum of the receiver windows of all the TCP connections.
- With limited buffer sizes, TCP performs poorly over vanilla UBR switches. TCP throughput is low, and there is unfairness among the connections. The coarse granularity TCP timer is an important reason for low TCP throughput.
- Efficiency typically increases with increasing buffer size.
- Fast retransmit and recovery improve performance for LAN configurations, but degrade performance in long latency configurations.
- SACK TCP improves performance especially for large latency networks.
- Early Packet Discard improves efficiency but not fairness.
- Per-VC buffer management improves both efficiency and fairness.

3 Buffer Requirements Study

In this contribution we present results of TCP throughput over satellite UBR for various delays, buffer sizes and number of sources.

1. **Latency.** Our primary aim is to study the performance of large latency connections. The typical latency from earth station to earth station for a single LEO (700 km altitude, 60 degree elevation angle) hop is about 5 ms [10]. The latencies for multiple LEO hops can easily be up to 50 ms from earth station to earth station. GEO latencies are typically 275 ms from earth station to earth station. We study these three latencies (5 ms, 50 ms, and 275 ms) with various number of sources and buffer sizes.

2. **Number of sources.** To ensure that the recommendations are scalable and general with respect to the number of connections, we will use configurations with 5, 15 and 50 TCP connections on a single bottleneck link. For single hop LEO configurations, we use 15, 50 and 100 sources.

3. **Buffer size.** This is the most important parameter of this study. The set of values chosen are $2^{-k} \times RTT, k = -1..6$, (i.e., 2, 1, 0.5, 0.25, 0.125, 0.0625, 0.031, 0.016 multiples of the round trip delay-bandwidth product of the TCP connections.) We plot the buffer size against the achieved TCP throughput for different delay-bandwidth products and number of sources. The asymptotic nature of this graph provides information about the optimal buffer size for the best cost-performance ratio.

4. **Switch drop policy.** We use a per-VC buffer allocation policy called selective drop (see [2]) to fairly allocate switch buffers to the competing connections.

5. **End system policies.** We use an enhanced version of TCP called SACK TCP for this study. SACK TCP improves performance by using selective acknowledgements for retransmission. Further details about our SACK TCP implementation can be found in [3].

4 Simulation Setup

Figure 1 shows the basic network configuration that was simulated. In the figure, the switches represent the earth stations that connect to the satellite constellation. The entire satellite network is assumed to be a 155 Mbps ATM link without any on board processing or queuing. All processing and queuing are performed at the earth stations.

- All simulations use the N source configuration shown in Figure 1. All sources are identical and infinite TCP sources. The TCP layer always sends a segment as long as it is permitted by the TCP window. Moreover, traffic is unidirectional so that only the sources send data. The destinations only send ACKs. The delayed acknowledgement timer is deactivated, and the receiver sends an ACK as soon as it receives a segment. As discussed before, SACK TCP is used in our simulations.

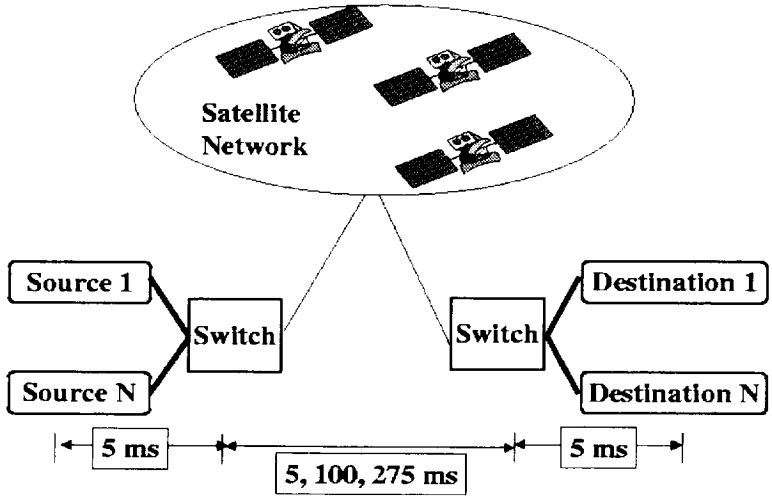


Figure 1: The N source TCP configuration

- Three different configurations are simulated that represent a single LEO hop, multiple LEO hops and a single GEO hop. The link delays between the switches and the end systems are 5 ms in all configurations. The inter-switch (earth station to earth station) propagation delays are 5 ms, 100 ms, and 275 ms for single hop LEO, multiple hop LEO and GEO configurations respectively. This results in a round trip propagation delays of 30 ms, 120 ms and 570 ms respectively.

- The number of sources (N) was 15, 50, and 100 for single hop LEO, and 5, 15 and 50 for GEO and multiple hop LEO configurations.

- The maximum value of the TCP receiver window is 600000 bytes, 2500000 bytes and 8704000 bytes for single hop LEO, multiple hop LEO and GEO respectively. These window sizes are sufficient to fill the 155.52 Mbps links.

- The TCP maximum segment size is 9180 bytes. A larger value is used because most TCP connections over ATM with satellite delays are expected to use larger segment sizes.

- The buffer sizes (in cells) used in the switch are the following:

  - Single LEO: 375, 750, 1500, 3000, 6000, 12000 (=1 RTT) , 24000 and 36000.
  - Multiple LEO: 780, 1560, 3125, 6250, 12500, 50000 (=1 RTT) , and 100000.
  - GEO: 3375, 6750, 12500, 25000, 50000, 100000, 200000 (=1 RTT) , and 400000.

- The duration of simulation is 100 seconds for multiple hop LEO and GEO and 20 secs for single hop LEO.

- All link bandwidths are 155.52 Mbps, and peak cell rate at the ATM layer is 149.7 Mbps after the SONET overhead.

5 Simulation Results

Figures 2, 3, and 4 show the resulting TCP efficiencies for the 3 different latencies. Each point in the figure shows the efficiency (total achieved TCP throughput divided by maximum possible throughput) against the buffer size used. Each figure plots a different latency, and each set of points (connected by a line) in a figure represents a particular value of N (the number of sources). The following conclusions can be drawn from the figures:

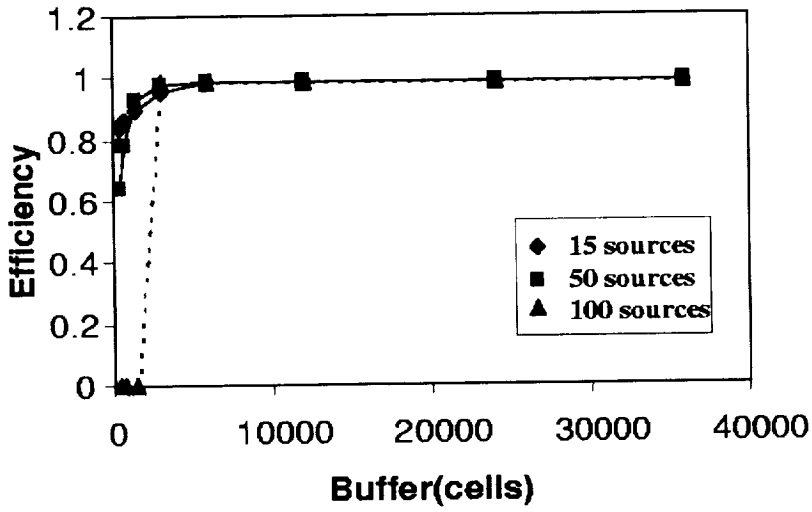


Figure 2: Buffer requirements for single hop LEO

1. For very small buffer sizes, (0.016×RTT, 0.031×RTT, 0.0625×RTT), the resulting TCP throughput is very low. In fact, for a large number of sources (N=50) , the throughput is sometimes close to zero.

2. For moderate buffer sizes (less then 1 round trip delay-bandwidth), TCP throughput increases with increasing buffer sizes.

3. TCP throughput asymptotically approaches the maximal value with further increase in buffer sizes.

4. **TCP performance over UBR for sufficiently large buffer sizes is scalable with respect to the number of TCP sources.** The throughput is never 100%, but for buffers greater than 0.5×RTT, the average TCP throughput is over 98% irrespective of the number of sources.

5. The knee of the buffer versus throughput graph is more pronounced for larger number of sources. For a large number of sources, TCP performance is very poor for small buffers, but jumps dramatically with sufficient buffering and then stays about the same. For smaller number of sources, the increase in throughput with increasing buffers is more gradual.

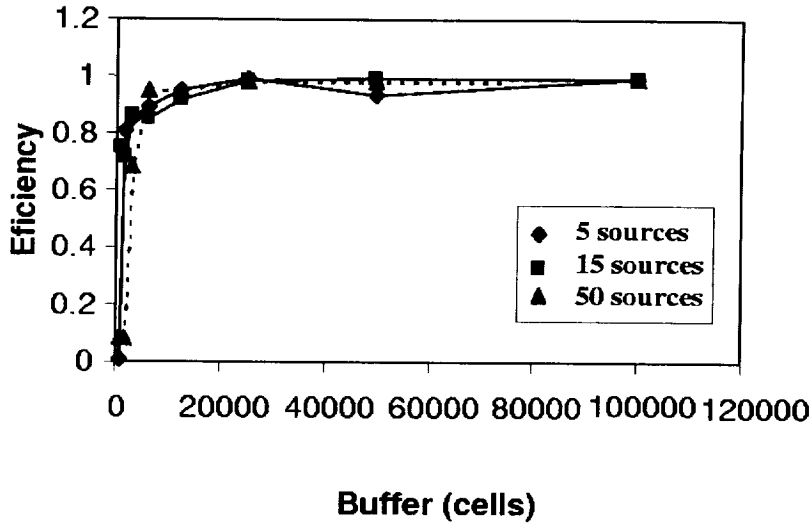


Figure 3: Buffer requirements for multiple hop LEO

6. **For large round trip delays, and a small number of sources, a buffer of 1 RTT or more can result in a slightly reduced throughput.** This is because of the variability in the TCP retransmission timer value. When the round trip is of the order of the TCP timer granularity (100 ms in this experiment), and the queuing delay is also of the order of the round trip time, the retransmission timeout values become very variable. During the initial phase (startup exponential increase), when the queueing delays are small, the timeout value corresponds to the propagation delay. When the windows increase to fill the switch buffer, the queuing delay increases to about 1 RTT and packets at the tail of the queue get dropped. Retransmitted packets are sent out after 3 duplicate ACKS are received. However, these retransmitted packets are queued behind a whole RTT worth of queues at the bottleneck switch. As a result, before the sender gets an ACK for retransmitted packets, a timeout occurs, and slow start is incurred. At this point, the sender starts to retransmit from the last unacked segment, but soon receives an ACK for that segment (because the segment was not really lost, but the delay was incorrectly estimated). The loss in throughput is due to the time lost in waiting for the retransmission timeout.

7. Fairness is high for a large number of sources. This shows that TCP sources with a good per-VC buffer allocation policy like selective drop, can effectively share the link bandwidth.

6 Summary

A buffer size of about 0.5×RTT to 1×RTT is sufficient to provide over 98% throughput to infinite TCP traffic for long latency networks and a large nubmer of sources. This buffer requirement is independent of the number of sources. The fairness is high for a large numbers of sources because of the nature of TCP traffic and the per-VC buffer management performed at the switches.

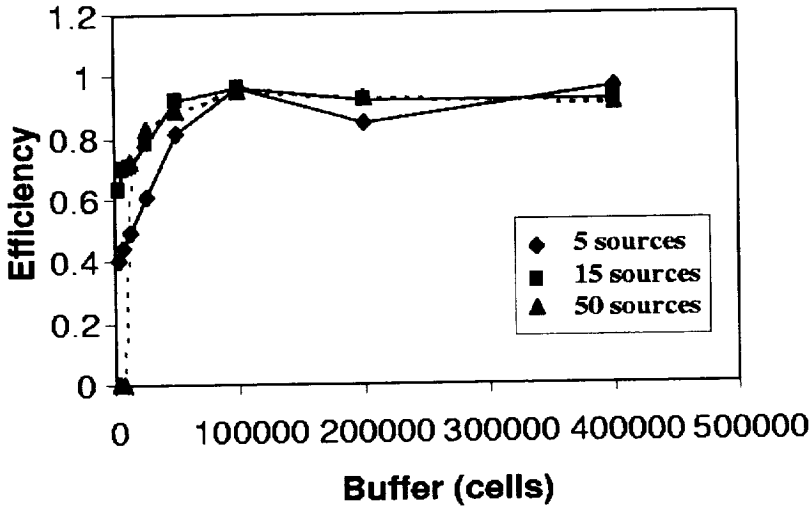


Figure 4: Buffer requirements for GEO

Throughput may slightly decrease for buffers larger than 1RTT because of variability in the RTT estimate approaches the timer granularity.

References

[1] Juha Heinanen, and Kalevi Kilkki, "A fair buffer allocation scheme," Unpublished Manuscript.

[2] R. Goyal, R. Jain, S. Kalyanaraman, S. Fahmy and Seong-Cheol Kim, "UBR+: Improving Performance of TCP over ATM-UBR Service," Proc. ICC'97, June 1997.

[3] R. Goyal, R. Jain et.al., "Selective Acknowledgements and UBR+ Drop Policies to Improve TCP/UBR Performance over Terrestrial and Satellite Networks," To appear: Proceedings of IC3N'97, September 1997. [1]

[4] Grace Yee, Sastri Kota, Gary Ogasawara, "TCP Performance over a Satellite ATM Network," Submitted to Globecom'97.

[5] V. Jacobson, "Congestion Avoidance and Control," Proceedings of the SIGCOMM'88 Symposium, pp. 314-32, August 1988.

[6] V. Jacobson, R. Braden, "TCP Extensions for Long-Delay Paths," Internet RFC 1072, October 1988.

[7] V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance," Internet RFC 1323, May 1992.

[1] All our papers and ATM Forum contributions are available from http://www.cis.ohio-state.edu/~jain

[8] M. Mathis, J. Madhavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgement Options," Internet RFC 2018, October 1996.

[9] M. Mathis, J. Madhavi, "Forward Acknowledgement: Refining TCP Congestion Control," Proc. SIGCOMM'96, August 1996.

[10] Satellite altitudes taken from "Lloyd's satellite constellation," http://www.ee.surrey.ac.uk/Personal/L.Wood/constellations/overview.html

IV. UBR bandwidth starvation by higher priority VBR traffic

IV.A

"Guaranteed Rate for Improving TCP Performance on UBR+ over Terrestrial and Satellite Networks"

****

ATM Forum Document Number: ATM_Forum/97-0424

****

Title: Guaranteed Rate for Improving TCP Performance on UBR+ over Terrestrial and Satellite Networks.

****

Abstract: We analyze the effect of providing a guaranteed rate to the UBR service class on the TCP performance over UBR. We describe an enhancement to UBR+ called the guaranteed rate service. Guaranteed rate service provides a minimum rate guarantee to the UBR traffic class. We examine effect of strict priority VBR over UBR traffic and show that the UBR performance can be improved by guaranteed rate. We also present simulation results for the effect of guaranteed rate on TCP efficiency and fairness metrics for LAN, WAN and satellite networks.

****

Source:
Rohit Goyal, Raj Jain, Shiv Kalyanaraman, Sonia Fahmy, Bobby Vandalore, Xiangrong Cai
Department of CIS, The Ohio State University (and NASA)
395 Dreese lab, 2015 Neil Ave,
Columbus, OH 43210-1277
Phone: 614-292-3989, Fax: 614-292-2911, Email: {goyal,jain}@cis.ohio-state.edu

Seong-Cheol Kim
Samsung Electronics Co. Ltd.
Chung-Ang Newspaper Bldg.
8-2, Karak-Dong, Songpa-Ku
Seoul, Korea 138-160
Email: kimsc@metro.telecom.samsung.co.kr

Sastri Kota
Lockheed Martin Telecommunications
1272 Borregas Avenue,
Bldg B/551 O/GB - 70
Sunnyvale, CA 94089
Email: sastri.kota@lmco.com

****

Date: April 1997

****

Distribution: ATM Forum Technical Working Group Members (AF-TM)

****

****

1 Introduction

TCP performance over UBR+ can be degraded when high priority VBR uses up 100% of the link. Providing a rate guarantee to the UBR class can ensure a continuous flow of TCP packets it the network. The Guaranteed Rate (GR) service provides such a guarantee to the UBR service category. The guarantees provided are for the entire UBR class, and per-VC guarantees are not provided. UBR+ with Guaranteed Rate requires no additional signalling requirements or standards changes, and can be implemented on current switches that support the UBR service. Another UBR+ service called Guaranteed Rate (GR) has been proposed in [11, 12]. The Guaranteed Rate service requires per-VC rate guarantees to UBR. This is more complex to implement and could significantly increase the cost of UBR switches. The Guaranteed Rate (GR) service is intended for applications that do not need any QoS guarantees, but whose performance depends on the availability of a continuous amount of bandwidth. GR guarantees a minimum rate to the UBR applications, while maintaining the simplicity of the basic UBR service. This guarantee is made for the entire UBR class for each link in the switch. The goal of GR is to protect the UBR class from total bandwidth starvation, and provide a continuous minimum bandwidth guarantee. In the presence of high load of higher priority CBR, VBR and ABR traffic, TCP congestion control mechanisms are expected to benefit from a guaranteed minimum rate.

In this paper, we discuss the performance of TCP with UBR in the presence of higher priority traffic. We present simulation results that show how the performance of TCP over UBR can degrade in the presence of VBR, and study the behavior of TCP over UBR with GR. Simulation results on the performance of TCP over UBR with and without GR are presented.

2 TCP over UBR+

In this section, we describe the basic TCP congestion control mechanisms and the UBR+ drop policies. We briefly discuss our implementaions of the UBR+ switch drop policies used in our simulations to optimize TCP performance over UBR.

2.1 TCP Congestion Control

TCP uses a window based protocol for congestion control. The sender TCP maintains a variable called congestion window (CWND) that limits the number of unacknowledged packets that can be sent. Current and proposed versions of the TCP protocol use the following three methods for congestion avoidance and control. For a detailed discussion on the TCP model and its performance over UBR+, refer to [10].

- **Slow start and congestion avoidance (Vanilla TCP).** The sender TCP detects congestion when a retransmission timeout expires. At this time, half the congestion window value is saved in SSTHRESH, and CWND is set to one segment. The sender now doubles CWND every round trip time until CWND reaches SSTHRESH, after which CWND is increased by one segment every round trip time. These two phases correspond to an exponential increase and a linear increase in CWND respectively. The retransmission timeout is maintained as a coarse granularity timer. As a result, even when a single packet is dropped, much time is lost waiting for the timeout to occur and then to increase the window back to SSTHRESH.

- **Fast retransmit and recovery (Reno TCP).** This mechanism was developed to optimize TCP performance for isolated segment losses due to errors. If a segment is lost, the data receiver sends duplicate ACKs for each out of sequence segment it receives. The sending TCP waits for three duplicate ACKs and retransmits the lost packet immediately. It then waits for half a window and sends a segment for each subsequent duplicate ACK. When the retransmitted packet is ACKed, the sending TCP sets CWND to half of its original value, and enters the congestion avoidance phase.

- **Selective acknowledgements (SACK TCP).** Fast retransmit and recovery cannot recover efficiently from multiple packet losses in the same window. Selective acknowledgements can be provided by the receiving TCP indicating to the sender the packets it has received. During the fast retransmission phase, the sender first retransmits the lost packets before sending out any new packets.

2.2 UBR+ Drop Policies

In [9], we examined TCP performance over UBR using various switch drop policies. These policies are:

- **Tail Drop.** This is the simplest possible drop policy, where the switch drops cells when the buffer becomes full. Tail drop typically results in poor performance of TCP over UBR.
- **Early Packet Discard (EPD).** Early Packet Discard maintains a threshold in its buffer. When the buffer occupancy exceeds the threshold, EPD drops complete new incoming packets to the buffer. EPD avoids the transmission of incomplete packets and increases TCP throughput over tail drop.
- **Selective Drop (SD) and Fair Buffer Allocation.** These schemes use per-VC accounting to maintain buffer utilizations of each active VC in the switch. When the buffer occupancy exceeds a preset threshold, complete packets are dropped from connections that are using more buffer than others. As a result greater fairness is achieved.

3 The UBR+ Guaranteed Rate Model

In this section we describe our implementation of the UBR+ GR model. Our ATM switch model is output buffered, where each output port has a separate buffer for each service category. Figure 1 shows our switch model. The switch supports multiple service categories as shown in the figure. Each service category is provided with a bandwidth guarantee. In our examples, we consider only two classes - VBR and UBR. VBR typically has strict priority over UBR, but with GR, UBR is guaranteed a fraction (=GR) of the total link capacity.

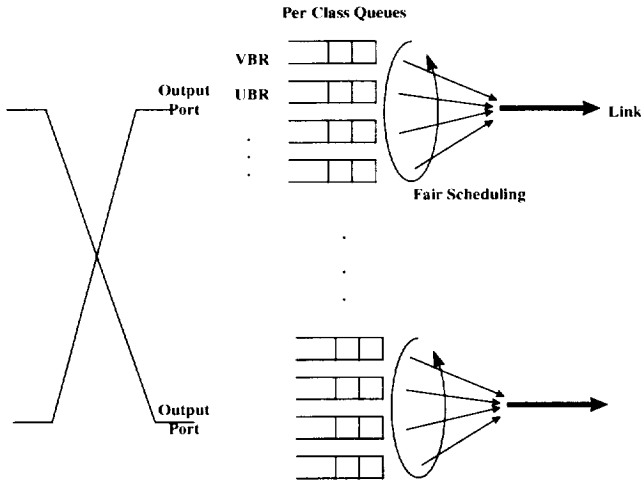


Figure 1: Switch model for UBR with GR

To enforce a GR (as a fraction of the total link capacity), we perform fair scheduling among the queues on each port. Our fair scheduling algorithm ensures that when GR ¿ 0.0, the UBR class is never starved, i.e., on the average, for every N cells transmitted on to the link, GR × N cells are from the UBR queue. This means that the VBR cells could be queued if the VBR connections are using more than (1-GR) of the link capacity. Any unused capacity by VBR is also allocated to UBR. The cell level minimum rate guarantee tranlates directly to a packet level guarantee for the TCP connections, because all TCP segment sizes in our simulations are the same. When transport packet sizes are different, per packet scheduling can be performed to provide frame level guarantees. The details of the scheduling algorithm will be presented in a future contribution.

Figure 2 shows the link capacity allocations for three values of GR. There is a single VBR source with an on/off burst pattern, which uses up 100% of the link capacity during the on period, and zero capacity during the off period. In the figure, the VBR on and off times are equal, so the average bandwidth requirements for VBR is 50% of the link capacity. When GR is 0, the VBR service is assigned strict priority over the UBR service. UBR is not guaranteed any rate, and must use whatever capacity is left over by the VBR source. The VBR bursts are scheduled just as they arrive and VBR cells are not queued. When GR = 0.1, 10% of the link capacity is guaranteed to the UBR service class. This 10% must be shared by all the UBR connections going through the link. In this case, if the VBR bursts may be queued in the VBR buffer to allow for UBR cells to be scheduled. The VBR bursts are thus flattened out with the VBR allocated Peak Cell Rate equal to 90% of the link capacity. Any link capacity unused by the VBR source is also available for UBR to use.

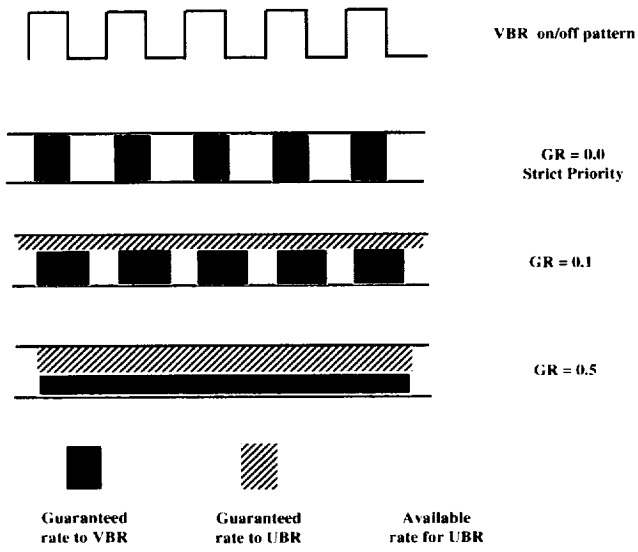


Figure 2: Link Capacity allocations for VBR and UBR with GR

When GR = 0.5, the VBR is further smoothed out so that it is now allocated a steady rate of 50% of the link capacity. On the average, the VBR queues are zero, but the on/off pattern results in temporary bursts until the burst can be cleared out.

In each of the three GR allocations, VBR uses up only 50% of the link capacity. As a result, UBR can use up to the remaining 50%. The difference between the three configurations is the way in which UBR is given the 50% capacity. With GR = 0, UBR is starved for the time VBR is using up 100guaranteed a continuous flow of bandwidth, and is never completely starved.

In this work, we experiment with a per-port bandwidth guarantee for UBR. The study of UBR with per-VC rate guarantees is a subject of future study.

4 Simulation of SACK TCP over UBR+

This section presents the simulation results of the various enhancements of TCP and UBR presented in the previous sections.

4.1 The Simulation Model

All simulations use the N source configuration shown in figure 3. Some simulations use an additional VBR source not shown in the figure. The VBR sources is also an end to end VBR source like the other TCP connections. All sources are identical and infinite TCP sources. The TCP layer always sends a segment as long as it is permitted by the TCP window. Moreover, traffic is unidirectional so that only the sources send data. The destinations only send ACKs. The performance of TCP over UBR with bidirectional traffic is a topic of further study. The delayed acknowledgement timer is deactivated, and the receiver sends an ACK as soon as it receives a segment.

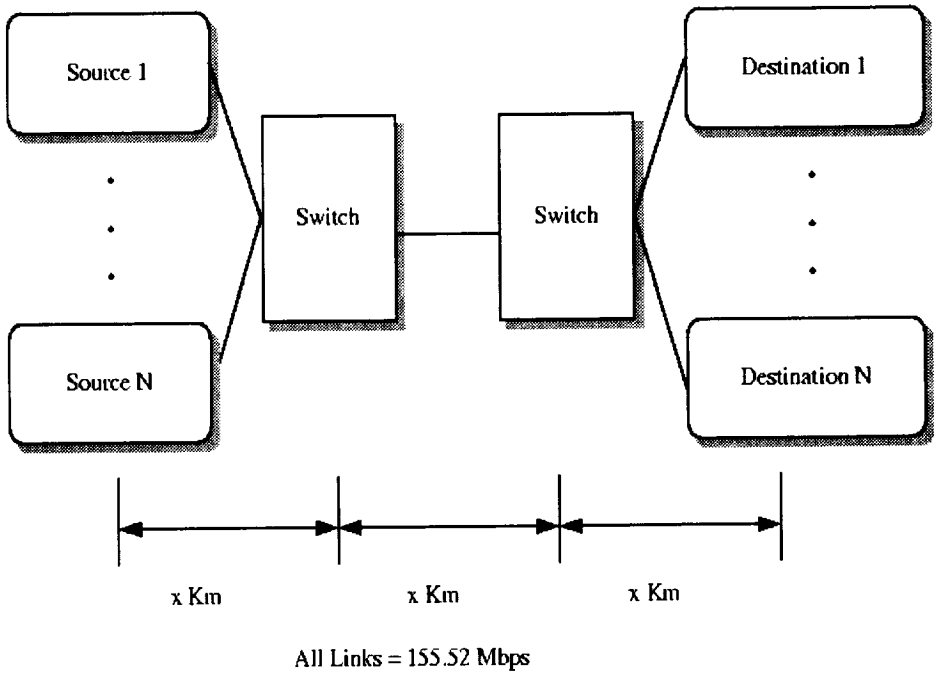


Figure 3: The N source TCP configuration

Link delays are 5 microseconds for LAN configurations and 5 milleseconds for WAN configurations. This results in a round trip propagation delay of 30 microseconds for LANs and 30 milliseconds for WANs respectively. For satellite configurations, the propagation delay between the two switches is 275 milliseconds and the distance between the TCPs and the switches is 1 km. The round trip propagation delay for satellite networks is about 550 milliseconds.

The TCP segment size is set to 512 bytes for LAN and WAN configurations. This is the common segment size used in most current TCP implementations. For satellite netowrks, larger segment sizes have been proposed, and we use a segment size of 9180 bytes. For the LAN configurations, the TCP maximum window size is limited by a receiver window of 64K bytes. This is the default value specified for TCP implementations. For WAN configurations, a window of 64K bytes is not sufficient to achieve 100% utilization. We thus use the window scaling option to specify a maximum window size of 600000 Bytes. For satellite configurations, this value is further scaled up to 8704000 Bytes.

All link bandwidths are 155.52 Mbps, and Peak Cell Rate at the ATM layer is 155.52 Mbps. The Duration of the simulation is 10 seconds for LANs, 20 seconds for WANs and 40 seconds for satellites. This allows for adequate round trips for the simulation to give stable results.

4.2 Performance Metrics

The performance of the simulation is measured at the TCP layer by the Efficiency and Fairness as defined below.

$$\text{Efficiency} = (\text{Sum of TCP throughputs})/(\text{Maximum possible TCP throughput})$$

TCP throughput is measured at the destination TCP layer as the total number of bytes delivered to the application divided by the simulation time. This is divided by the maximum possible throughput attainable by TCP. With 512

Table 1: SACK TCP with VBR (strict priority) : Efficiency

Config-uration	Number of Sources	Buffer (cells)	VBR period (ms)	UBR	EPD	Selective Drop
LAN	5	1000	300	0.71	0.88	0.98
LAN	5	3000	300	0.83	0.91	0.92
LAN	5	1000	100	0.89	0.97	0.95
LAN	5	3000	100	0.96	0.95	0.96
LAN	5	1000	50	0.97	0.93	0.93
LAN	5	3000	50	0.95	0.97	0.97
WAN	5	12000	300	0.42	0.43	0.61
WAN	5	36000	300	0.55	0.52	0.96
WAN	5	12000	100	0.72	0.58	0.70
WAN	5	36000	100	0.95	0.97	0.97
WAN	5	12000	50	0.97	0.65	0.73
WAN	5	36000	50	0.97	0.98	0.98

bytes of TCP data in each segment, 20 bytes of TCP header, 20 bytes of IP header, 8 bytes of LLC header, and 8 bytes of AAL5 trailer are added. This results in a net possible throughput of 80.5% of the ATM layer throughput for UBR. Without VBR, the the maximum possible throughput is 125.2 Mbps on a 155.52 Mbps link. When a VBR source uses up 50% of the capacity, then the maximum possible TCP throughput reduces to 80.5% of 50% of 155.52 Mbps. This evaluates to about 63 Mbps.

$$\text{Fairness Index} = (\Sigma x_i)^2 / (n \times \Sigma x_i^2)$$

Where x_i = throughput of the ith TCP source, and n is the number of TCP sources

5 Simulation Results

When higher priority VBR traffic is present in the network, TCP over UBR may get considerably lower link capacity than without VBR. Moreover, the presence of VBR traffic could result in the starvation of UBR traffic for periods of time for which VBR uses up the entire link capacity. When VBR has strict priority over UBR, TCP (over UBR) traffic is transmitted in bursts and the round trip time estimates for the TCP connection are highly variable. An underestimation of the RTT is likely to cause a false timeout in the TCP indicating congestion even though the TCP packet is queued behind a VBR burst. An overestimtion of the RTT may result in much time being wasted waiting for a timeout when a packet is dropped due to congestion.

5.1 SACK TCP over UBR+ with strict priority VBR background

The effect of UBR starvation is seen in tables 1 and 2. In this set of simulations, we used five source LAN and WAN configurations with SACK TCP. SACK TCP was chosen because it provides best performance for TCP over UBR+ [10]. Three different VBR on/off periods were simulated - 300ms, 100ms and 50ms. In each case, the on times were equal to the off times and, during the on periods, the VBR usage was 100% of the link capacity. VBR was given strict priority over UBR, i.e., GR for UBR was 0. From the tables we can see that longer VBR bursts (for the same average VBR usage of 50%) result in lower throughput for TCP over UBR+.

Figure 4 shows the efficiency versus fairness plots for tables 1 and 2. The desirable points are those on the upper right corners of the plots, i.e., those with high efficiency and fairness values. For the WAN configuration, the upper right corner points are those from the lower VBR on/off frequencies (50 and 100 ms). With 300 ms VBR, TCP performance for WANs is poor. This is because, the VBR burst time is of the order of the TCP timeout value (2 to 3 ticks of 100 ms each). As a result the TCP source is starved long enough that a retransmission timeout occurs.

Table 2: SACK TCP with VBR (strict priority) : Fairness

Config-uration	Number of Sources	Buffer (cells)	VBR period (ms)	UBR	EPD Drop	Selective
LAN	5	1000	300	0.21	0.20	0.20
LAN	5	3000	300	0.95	0.99	0.99
LAN	5	1000	100	0.21	0.20	0.99
LAN	5	3000	100	0.91	0.93	0.96
LAN	5	1000	50	0.20	0.21	0.96
LAN	5	3000	50	0.93	0.99	1.00
WAN	5	12000	300	0.99	0.97	0.82
WAN	5	36000	300	0.88	0.97	0.63
WAN	5	12000	100	0.99	0.96	0.93
WAN	5	36000	100	1.00	0.88	0.89
WAN	5	12000	50	0.92	0.98	0.97
WAN	5	36000	50	1.00	0.97	0.80

Much time (several roundtrips of at least 30 ms each) is then wasted in recovering from the timeout during the slow start phase. This causes poor utilization of the link and lower efficiency values. When VBR on/off times are smaller compared to the retransmission value, the UBR delay is not enough to result in a TCP timeout and higher throughput results.

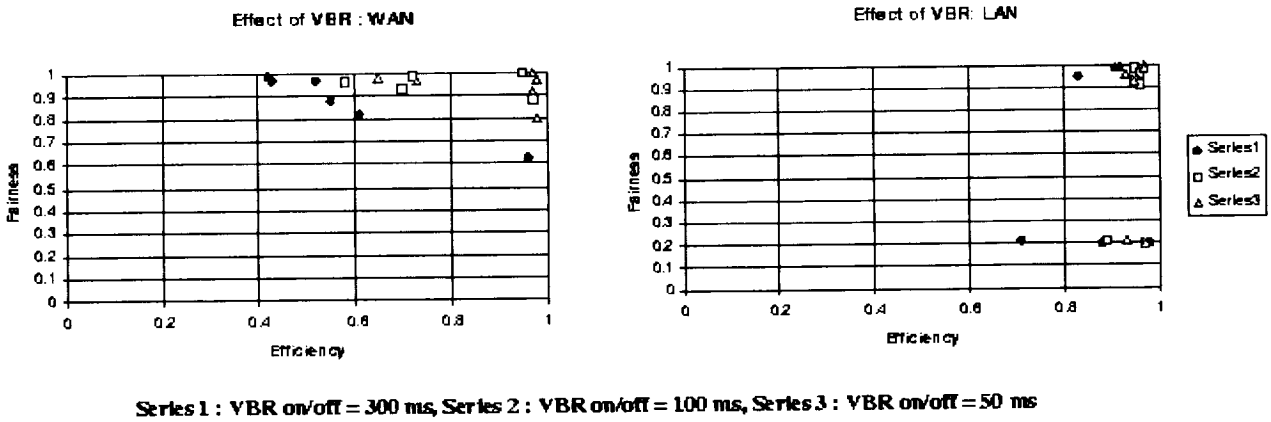


Figure 4: Variable VBR frequencies of UBR+ with strict priority

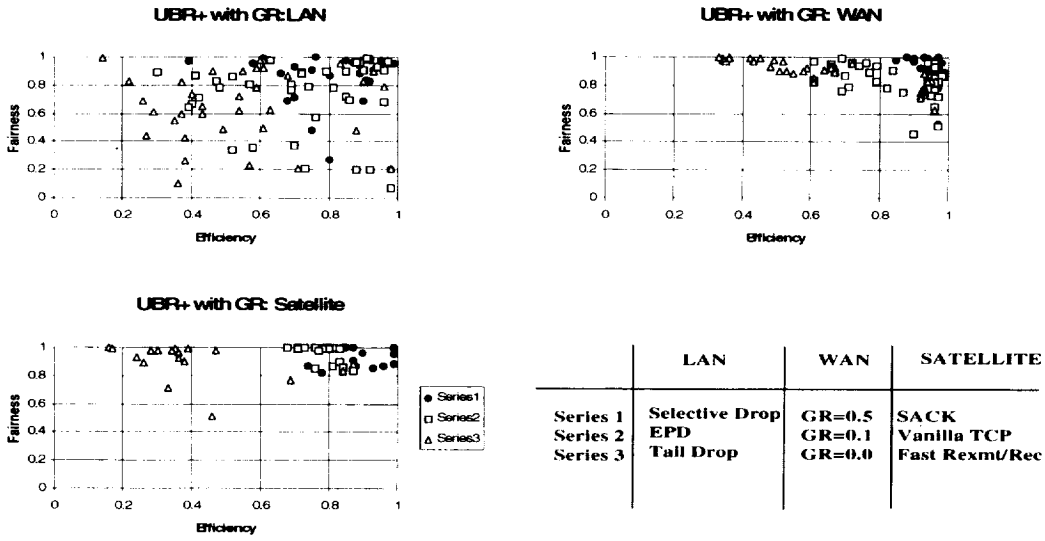
For LANs, the above argument also holds, but other factors are more dominant. The LAN plot in 4 shows that the effects of the switch drop policy and the buffer size are also important. The selective drop policy significantly improves the LAN performance of TCP over UBR+. This is because the round trip time is very small, and even during the congestion avoidance phase, the recovery is very fast. The TCP timeouts are often in phase with the VBR burst times. As a result, when TCP is waiting for the timer to expire, and not utilizing the link, VBR is using the link at 100% capacity. When TCP times out and starts to send segments, the congestion window increases very fast.

5.1.1 SACK TCP over Guaranteed Rate UBR+ with VBR background

We now present simulation results for TCP over UBR+ with various GR values. For LAN, WAN and satellite configurations, we ran simulations with the following parameters :

- Number of sources = 5, 15 for LAN and WAN. For satellite networks, we ran the same set but only for 5 sources.
- Buffer size = 1000 cells and 3000 cells for LANs, 12000 cells and 36000 cells for WANs; and 200000 cells and 600000 cells for satellites.
- Vanilla TCP (with only slow start and congestion avoidance), Reno TCP (with fast retransmit and recovery) and SACK TCP.
- Tail Drop UBR, EPD and Selective Drop.
- UBR GR = 0.5, 0.1, 0.0 of the link capacity.

The tables 3 - 9 in the list the results of the simulations. From the tables, we categorized the results in terms of the highest effeciency and fairness values. The plots in figure 5 summarize the results in the tables.



	LAN	WAN	SATELLITE
Series 1	Selective Drop	GR=0.5	SACK
Series 2	EPD	GR=0.1	Vanilla TCP
Series 3	Tail Drop	GR=0.0	Fast Rexmt/Rec

Figure 5: TCP performance over UBR+ with GR

The following observations can be made from the tables and the plots:

1. **For LANs, the dominating factor that effects the performance is the switch drop policy.** Series 1 in the figure represents the points for the selective drop policy. Clearly, selective drop improves the performance irrespective of most TCP and GR parameters. This result holds with or without the presence of background VBR traffic. In LANs, the switch buffer sizes are of the order of 1000 and 3000 cells. This is very small in comparison with the maximum TCP receiver window. As a result, TCP can easily overload the switch buffers. This makes buffer management very important for LANs.
2. **For WANs, the dominating factor is the GR, and a GR of 0 hurts the TCP performance.** GR values of 0.5 and 0.1 produce the highest throughput and effeciency values. A constant amount of bandwidth provided by GR ensures that TCP keeps receiving ACKs from the destination. This reduces the variation in the round trip times. Consequently, TCP is less likely to timeout. Buffer management policies do have an

impact on TCP performance over WANs, but the effect is less than in LANs. This is because the buffer sizes of WAN switches are comparable to the bandwidth × round trip delays of the network. The TCP maximum windows are also usually based on the round trip times. As a result, buffers are more easily available, and drop policies are less important.

3. **For satellite networks, the TCP congestion control mechanism makes the most difference; SACK TCP produces the best results, and Reno TCP results in the worst performance.** SACK TCP ensures quick recovery from multiple packet losses, whereas fast retransmit and recovery is unable to recover from multiple packet drops. The satellite buffer sizes are quite large, and so the drop policies do not make a significant difference. The GR fractions do not significantly affect the TCP performance over satellite networks because in our simulations, the VBR burst durations are smaller than the round trip propagation delays. The retransmission timeout values are typically close to 1 second, and so a variation of the RTT by 300 milliseconds can be tolerated by the TCP. GR may have more impact on satellite networks in cases where UBR is starved for times larger than the round trip time of the connection.

6 Summary

In this paper we examined the effect of higher priority VBR traffic on the performance of TCP over UBR+. Several factors can effect the performance of TCP over UBR in the presence of higher priority VBR traffic. These factors include:

- The propagation delay of the TCP connection.
- The TCP congestion control mechanisms.
- The UBR switch drop policies.
- The Guaranteed Rate provided to UBR.

For large propagation delays, end-to-end congestion control is the most important factor. For small propagation delays, the limited switch buffers makes buffer management very important. A minimun bandwidth guarantee improves TCP performance over UBR when the TCP connection may be starved for periods longer than the round trip propagation delay. The minimum bandwidth scheme explored here provides a minimum rate to the entire UBR class on the link. Per-VC GR mechanisms are an area of future study.

References

[1] Allyn Romanov, Sally Floyd, "Dynamics of TCP Traffic over ATM Networks," IEEE JSAC, May 1995.

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

[3] Chien Fang, Arthur Lin, "On TCP Performance of UBR with EPD and UB R-EPD with a Fair Buffer Allocation Scheme," ATM FORUM 95-1645, December 1995.

[4] Hongqing Li, Kai-Yeung Siu, and Hong-Ti Tzeng, "TCP over ATM with ABR service versus UBR+EPD service," ATM FORUM 95-0718, June 1995.

[5] H. Li, K.Y. Siu, H.T. Tzeng, C. Ikeda and H. Suzuki "TCP over ABR and UBR Services in ATM," Proc. IPCCC'96, March 1996.

[6] Hongqing Li, Kai-Yeung Siu, Hong-Yi Tzeng, Brian Hang, Wai Yang, " Issues in TCP over ATM," ATM FORUM 95-0503, April 1995.

[7] J. Jaffe, "Bottleneck Flow Control," IEEE Transactions on Communications, Vol. COM-29, No. 7, pp. 954-962.

[8] Juha Heinanen, and Kalevi Kilkki, "A fair buffer allocation scheme," Unpublished Manuscript.

Table 3: TCP with VBR (300ms on/off) over UBR+ with GR : Efficiency for LAN

Config-uration	Number of Sources	Buffer (cells)	TCP	GR	UBR	EPD	Selective Drop
LAN	5	1000	SACK	0.5	0.26	0.85	0.96
LAN	5	1000	SACK	0.1	0.98	0.57	0.75
LAN	5	1000	SACK	0.0	0.71	0.88	0.98
LAN	5	3000	SACK	0.5	0.96	0.97	0.95
LAN	5	3000	SACK	0.1	0.93	0.89	0.99
LAN	5	3000	SACK	0.0	0.83	0.91	0.92
LAN	15	1000	SACK	0.5	0.38	0.74	0.92
LAN	15	1000	SACK	0.1	0.49	0.76	0.91
LAN	15	1000	SACK	0.0	0.57	0.98	0.90
LAN	15	3000	SACK	0.5	0.90	0.96	0.92
LAN	15	3000	SACK	0.1	0.61	0.94	0.96
LAN	15	3000	SACK	0.0	0.43	0.86	0.95
LAN	5	1000	Reno	0.5	0.22	0.30	0.61
LAN	5	1000	Reno	0.1	0.37	0.41	0.66
LAN	5	1000	Reno	0.0	0.14	0.92	0.39
LAN	5	3000	Reno	0.5	0.60	0.69	0.76
LAN	5	3000	Reno	0.1	0.55	0.79	0.93
LAN	5	3000	Reno	0.0	0.59	0.72	0.92
LAN	15	1000	Reno	0.5	0.43	0.52	0.70
LAN	15	1000	Reno	0.1	0.35	0.48	0.68
LAN	15	1000	Reno	0.0	0.29	0.40	0.70
LAN	15	3000	Reno	0.5	0.68	0.88	0.95
LAN	15	3000	Reno	0.1	0.63	0.81	0.97
LAN	15	3000	Reno	0.0	0.54	0.69	0.89
LAN	5	1000	Vanilla	0.5	0.46	0.47	0.58
LAN	5	1000	Vanilla	0.1	0.40	0.58	0.70
LAN	5	1000	Vanilla	0.0	0.27	0.73	0.80
LAN	5	3000	Vanilla	0.5	0.88	0.72	0.87
LAN	5	3000	Vanilla	0.1	0.61	0.63	0.90
LAN	5	3000	Vanilla	0.0	0.61	0.88	0.85
LAN	15	1000	Vanilla	0.5	0.59	0.42	0.80
LAN	15	1000	Vanilla	0.1	0.38	0.52	0.70
LAN	15	1000	Vanilla	0.0	0.36	0.39	0.75
LAN	15	3000	Vanilla	0.5	0.68	0.90	0.97
LAN	15	3000	Vanilla	0.1	0.54	0.96	0.98
LAN	15	3000	Vanilla	0.0	0.37	0.85	0.89

Table 4: TCP with VBR (300ms on/off) over UBR+ with GR : Efficiency for WAN

WAN	5	12000	SACK	0.5	0.95	0.93	0.94
WAN	5	12000	SACK	0.1	0.87	0.66	0.69
WAN	5	12000	SACK	0.0	0.42	0.43	0.61
WAN	5	36000	SACK	0.5	0.97	0.99	0.99
WAN	5	36000	SACK	0.1	0.96	0.98	0.96
WAN	5	36000	SACK	0.0	0.55	0.52	0.96
WAN	15	12000	SACK	0.5	0.88	0.85	0.90
WAN	15	12000	SACK	0.1	0.72	0.61	0.76
WAN	15	12000	SACK	0.0	0.64	0.48	0.58
WAN	15	36000	SACK	0.5	0.96	0.95	0.97
WAN	15	36000	SACK	0.1	0.95	0.94	0.97
WAN	15	36000	SACK	0.0	0.93	0.72	0.95
WAN	5	12000	Reno	0.5	0.93	0.96	0.94
WAN	5	12000	Reno	0.1	0.61	0.79	0.71
WAN	5	12000	Reno	0.0	0.34	0.45	0.33
WAN	5	36000	Reno	0.5	0.97	0.97	0.93
WAN	5	36000	Reno	0.1	0.90	0.96	0.75
WAN	5	36000	Reno	0.0	0.33	0.92	0.33
WAN	15	12000	Reno	0.5	0.97	0.94	0.97
WAN	15	12000	Reno	0.1	0.84	0.66	0.79
WAN	15	12000	Reno	0.0	0.67	0.53	0.51
WAN	15	36000	Reno	0.5	0.97	0.97	0.98
WAN	15	36000	Reno	0.1	0.96	0.96	0.97
WAN	15	36000	Reno	0.0	0.67	0.66	0.59
WAN	5	12000	Vanilla	0.5	0.94	0.97	0.96
WAN	5	12000	Vanilla	0.1	0.82	0.70	0.69
WAN	5	12000	Vanilla	0.0	0.49	0.36	0.42
WAN	5	36000	Vanilla	0.5	0.97	0.97	0.97
WAN	5	36000	Vanilla	0.1	0.96	0.90	0.94
WAN	5	36000	Vanilla	0.0	0.92	0.33	0.92
WAN	15	12000	Vanilla	0.5	0.90	0.92	0.96
WAN	15	12000	Vanilla	0.1	0.77	0.66	0.74
WAN	15	12000	Vanilla	0.0	0.67	0.61	0.67
WAN	15	36000	Vanilla	0.5	0.98	0.97	0.97
WAN	15	36000	Vanilla	0.1	0.96	0.96	0.97
WAN	15	36000	Vanilla	0.0	0.94	0.93	0.93

Table 5: SACK TCP with VBR (300ms on/off) over UBR+ with GR : Fairness for LAN

Config-uration	Number of Sources	Buffer (cells)	TCP	GR	UBR	EPD	Selective Drop
LAN	5	1000	SACK	0.5	0.69	0.90	0.97
LAN	5	1000	SACK	0.1	0.21	0.81	0.91
LAN	5	1000	SACK	0.0	0.21	0.20	0.20
LAN	5	3000	SACK	0.5	0.79	0.97	0.94
LAN	5	3000	SACK	0.1	0.90	0.96	0.95
LAN	5	3000	SACK	0.0	0.95	0.99	0.99
LAN	15	1000	SACK	0.5	0.43	0.79	0.83
LAN	15	1000	SACK	0.1	0.49	0.57	0.84
LAN	15	1000	SACK	0.0	0.23	0.07	0.69
LAN	15	3000	SACK	0.5	0.83	0.91	0.98
LAN	15	3000	SACK	0.1	0.50	0.93	0.91
LAN	15	3000	SACK	0.0	0.65	0.70	0.96
LAN	5	1000	Reno	0.5	0.83	0.89	0.99
LAN	5	1000	Reno	0.1	0.60	0.87	0.88
LAN	5	1000	Reno	0.0	0.99	0.20	0.97
LAN	5	3000	Reno	0.5	0.98	0.81	1.00
LAN	5	3000	Reno	0.1	0.90	0.90	0.91
LAN	5	3000	Reno	0.0	0.92	0.89	0.98
LAN	15	1000	Reno	0.5	0.60	0.86	0.93
LAN	15	1000	Reno	0.1	0.55	0.78	0.69
LAN	15	1000	Reno	0.0	0.61	0.67	0.37
LAN	15	3000	Reno	0.5	0.87	0.96	0.98
LAN	15	3000	Reno	0.1	0.63	0.78	0.95
LAN	15	3000	Reno	0.0	0.72	0.77	0.94
LAN	5	1000	Vanilla	0.5	0.90	0.83	0.95
LAN	5	1000	Vanilla	0.1	0.74	0.36	0.93
LAN	5	1000	Vanilla	0.0	0.44	0.21	0.27
LAN	5	3000	Vanilla	0.5	0.48	0.88	0.96
LAN	5	3000	Vanilla	0.1	0.92	0.98	0.98
LAN	5	3000	Vanilla	0.0	0.98	0.96	0.98
LAN	15	1000	Vanilla	0.5	0.78	0.71	0.87
LAN	15	1000	Vanilla	0.1	0.26	0.34	0.71
LAN	15	1000	Vanilla	0.0	0.10	0.64	0.48
LAN	15	3000	Vanilla	0.5	0.87	0.91	0.96
LAN	15	3000	Vanilla	0.1	0.62	0.68	0.95
LAN	15	3000	Vanilla	0.0	0.82	0.72	0.88

Table 6: SACK TCP with VBR (300ms on/off) over UBR+ with GR : Fairness for WAN

WAN	5	12000	SACK	0.5	0.95	1.00	0.99
WAN	5	12000	SACK	0.1	0.75	0.92	0.99
WAN	5	12000	SACK	0.0	0.99	0.97	0.82
WAN	5	36000	SACK	0.5	0.95	0.86	0.89
WAN	5	36000	SACK	0.1	0.96	0.87	0.77
WAN	5	36000	SACK	0.0	0.88	0.97	0.63
WAN	15	12000	SACK	0.5	1.00	0.98	0.99
WAN	15	12000	SACK	0.1	0.96	0.97	0.96
WAN	15	12000	SACK	0.0	0.91	0.93	0.90
WAN	15	36000	SACK	0.5	0.92	0.98	0.96
WAN	15	36000	SACK	0.1	0.73	0.96	0.83
WAN	15	36000	SACK	0.0	0.74	0.95	0.84
WAN	5	12000	Reno	0.5	0.77	0.93	0.96
WAN	5	12000	Reno	0.1	0.84	0.94	0.79
WAN	5	12000	Reno	0.0	0.99	0.99	1.00
WAN	5	36000	Reno	0.5	0.87	1.00	0.97
WAN	5	36000	Reno	0.1	0.46	0.82	0.97
WAN	5	36000	Reno	0.0	1.00	0.71	1.00
WAN	15	12000	Reno	0.5	0.53	0.90	0.91
WAN	15	12000	Reno	0.1	0.91	0.95	0.83
WAN	15	12000	Reno	0.0	0.91	0.90	0.90
WAN	15	36000	Reno	0.5	0.90	0.79	0.96
WAN	15	36000	Reno	0.1	0.65	0.73	0.51
WAN	15	36000	Reno	0.0	0.89	0.92	0.92
WAN	5	12000	Vanilla	0.5	0.99	0.78	0.89
WAN	5	12000	Vanilla	0.1	0.78	0.87	0.76
WAN	5	12000	Vanilla	0.0	0.98	0.99	0.99
WAN	5	36000	Vanilla	0.5	1.00	0.78	0.98
WAN	5	36000	Vanilla	0.1	0.93	0.46	0.83
WAN	5	36000	Vanilla	0.0	0.75	1.00	0.73
WAN	15	12000	Vanilla	0.5	0.97	0.92	0.95
WAN	15	12000	Vanilla	0.1	0.89	0.94	0.94
WAN	15	12000	Vanilla	0.0	0.93	0.85	0.92
WAN	15	36000	Vanilla	0.5	0.89	0.88	0.92
WAN	15	36000	Vanilla	0.1	0.97	0.85	0.72
WAN	15	36000	Vanilla	0.0	0.83	0.77	0.88

Table 7: TCP with VBR (300ms on/off) over UBR+ with GR : Satellite

Drop Policy	TCP	Buffer	GR	Efficiency	Fairness
Selective Drop	SACK	200000	0.5	0.87	0.91
Selective Drop	SACK	200000	0.1	0.78	0.82
Selective Drop	SACK	200000	0.0	0.74	0.87
Selective Drop	SACK	600000	0.5	0.99	1.00
Selective Drop	SACK	600000	0.1	0.99	0.99
Selective Drop	SACK	600000	0.0	0.99	1.00
Selective Drop	Reno	200000	0.5	0.33	0.71
Selective Drop	Reno	200000	0.1	0.24	0.93
Selective Drop	Reno	200000	0.0	0.16	1.00
Selective Drop	Reno	600000	0.5	0.35	0.99
Selective Drop	Reno	600000	0.1	0.39	0.99
Selective Drop	Reno	600000	0.0	0.30	0.98
Selective Drop	Vanilla	200000	0.5	0.83	0.90
Selective Drop	Vanilla	200000	0.1	0.71	0.99
Selective Drop	Vanilla	200000	0.0	0.81	0.87
Selective Drop	Vanilla	600000	0.5	0.79	1.00
Selective Drop	Vanilla	600000	0.1	0.80	0.99
Selective Drop	Vanilla	600000	0.0	0.76	1.00

Table 8: TCP with VBR (300ms on/off) over UBR+ with GR : Satellite

Drop Policy	TCP	Buffer	GR	Efficiency	Fairness
Early Packet Discard	SACK	200000	0.5	0.84	1.00
Early Packet Discard	SACK	200000	0.1	0.88	0.87
Early Packet Discard	SACK	200000	0.0	0.82	0.99
Early Packet Discard	SACK	600000	0.5	0.99	0.95
Early Packet Discard	SACK	600000	0.1	0.99	0.88
Early Packet Discard	SACK	600000	0.0	0.99	1.00
Early Packet Discard	Reno	200000	0.5	0.46	0.51
Early Packet Discard	Reno	200000	0.1	0.26	0.89
Early Packet Discard	Reno	200000	0.0	0.17	0.99
Early Packet Discard	Reno	600000	0.5	0.36	0.96
Early Packet Discard	Reno	600000	0.1	0.34	0.98
Early Packet Discard	Reno	600000	0.0	0.28	0.98
Early Packet Discard	Vanilla	200000	0.5	0.71	1.00
Early Packet Discard	Vanilla	200000	0.1	0.76	0.85
Early Packet Discard	Vanilla	200000	0.0	0.68	1.00
Early Packet Discard	Vanilla	600000	0.5	0.78	0.99
Early Packet Discard	Vanilla	600000	0.1	0.80	0.99
Early Packet Discard	Vanilla	600000	0.0	0.77	0.98

Table 9: TCP with VBR (300ms on/off) over UBR+ with GR : Satellite

Drop Policy	TCP	Buffer	GR	Efficiency	Fairness
UBR	SACK	200000	0.5	0.87	0.91
UBR	SACK	200000	0.1	0.87	1.00
UBR	SACK	200000	0.0	0.85	1.00
UBR	SACK	600000	0.5	0.93	0.85
UBR	SACK	600000	0.1	0.96	0.87
UBR	SACK	600000	0.0	0.90	0.96
UBR	Reno	200000	0.5	0.87	0.88
UBR	Reno	200000	0.1	0.36	0.92
UBR	Reno	200000	0.0	0.38	0.9
UBR	Reno	600000	0.5	0.84	0.84
UBR	Reno	600000	0.1	0.69	0.77
UBR	Reno	600000	0.0	0.47	0.98
UBR	Vanilla	200000	0.5	0.87	0.84
UBR	Vanilla	200000	0.1	0.73	1.00
UBR	Vanilla	200000	0.0	0.84	0.86
UBR	Vanilla	600000	0.5	0.83	0.99
UBR	Vanilla	600000	0.1	0.83	0.99
UBR	Vanilla	600000	0.0	0.81	1.00

[9] R. Goyal, R. Jain, S. Kalyanaraman, S. Fahmy and Seong-Cheol Kim, "UBR+: Improving Performance of TCP over ATM-UBR Service," Proc. ICC'97, June 1997.

[10] R. Goyal, R. Jain et.al., "Selective Acknowledgements and UBR+ Drop Policies to Improve TCP/UBR Performance over Terrestrial and Satellite Networks," ATM FORUM 97-0423, April 1997.
[1]

[11] Roch Guerin, and Juha Heinanen, "UBR+ Service Category Definition," ATM FORUM 96-1598, December 1996.

[12] Roch Guerin, and Juha Heinanen, "UBR+ Enhancements," ATM FORUM 97-0015, February 1997.

[13] Shiv Kalyanaraman, Raj Jain, Sonia Fahmy, Rohit Goyal, Fang Lu and Saragur Srinidhi, "Performance of TCP/IP over ABR," Proc. IEEE Globecom'96, November 1996.

[14] Shivkumar Kalyanaraman, Raj Jain, Rohit Goyal, Sonia Fahmy and Seong-Cheol Kim, "Performance of TCP over ABR on ATM backbone and with various VBR background traffic patterns," Proc. ICC'97, June 1997.

[15] Stephen Keung, Kai-Yeung Siu, "Degradation in TCP Performance under Cell Loss," ATM FORUM 94-0490, April 1994.

[16] Tim Dwight, "Guidelines for the Simulation of TCP/IP over ATM," ATM FORUM 95-0077r1, March 1995.

[17] V. Jacobson, "Congestion Avoidance and Control," Proceedings of the SIGCOMM'88 Symposium, pp. 314-32, August 1988.

[18] V. Jacobson, R. Braden, "TCP Extensions for Long-Delay Paths," Internet RFC 1072, October 1988.

[19] V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance," Internet RFC 1323, May 1992.

[20] Kevin Fall, Sally Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP,"

[21] Sally Floyd, "Issues of TCP with SACK,"

[1] All our papers and ATM Forum contributions are available from http://www.cis.ohio-state.edu/~jain

[22] M. Mathis, J. Madhavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgement Options," Internet RFC 2018, October 1996.

[23] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," Internet RFC 2001, January 1997.

[24] Janey C. Hoe, "Start-up Dynamics of TCP's Congestion Control and Avoidance Schemes," MS Thesis, Massachusetts Institute of Technology, June 1995.

[25] Mark Allman, Chris Hayes, hans Kruse, Shawn Ostermann, " TCP Performance over Satellite Links," Proc. 5th International Conference on Telecommunications Systems, 1997.

V. Bursty Sources

V.A

"Performance Analysis TCP Enhancements for WWW Traffic using UBR+ with Limited Buffers over Satellite Links"

****

ATM Forum Document Number: ATM_Forum/98-0876R1

****

TITLE: Performance Analysis of TCP Enhancements for WWW Traffic using UBR+ with Limited Buffers over Satellite Links

****

SOURCE: Mukul Goyal, Rohit Goyal, Raj Jain, Bobby Vandalore, Sonia Fahmy
The Ohio State University,
Department of Computer and Information Science,
2015 Neil Ave, DL 395, Columbus, OH 43210
Phone: 614-688-4482
{mukul,goyal,jain}@cis.ohio-state.edu

Thomas vonDeak and Kul Bhasin
NASA Glenn Research Center, MS: 54-6
21000 Brookpark Road
Cleveland, OH 44135
Email: {tvondeak, bhasin}@lerc.nasa.gov

Norm Butts and Sastri Kota
Lockheed Martin Telecommunications
1272 Borregas Ave
Sunnyvale, CA 94089
Email: {Norm.Butts, Sastri.Kota}@lmco.com

This work is sponsored partially by NASA Glenn Research Center.

****

DISTRIBUTION: ATM Forum Technical Committee
Traffic Management Working Group

****

DATE: December, 1998 (Nashville)

****

ABSTRACT: This contribution presents the simulation results on the performance of TCP enhancements over ATM-UBR+ for satellite latencies with limited switch buffers for WWW traffic. We analyze the impact of Fast Retransmit and Recovery (FRR or Reno), New Reno, Selective Acknowledgement (SACK), Early packet drop (EPD), and Selective Drop on LEO, MEO, and GEO satellite configurations for several buffer sizes.

****

****

1 Introduction

In recent years there has been a tremendous growth in the amount of WWW traffic over the Internet. WWW traffic is essentially bursty in nature with periods of activity. The traffic pattern generated by a large number of WWW connections is expected to be different from that generated by persistent TCP traffic. In this contribution, we study the performance of WWW traffic over long delay networks for the UBR+ service. In our previous work [GOYAL97a, GOYAL97b, KOTA97], we have assessed the performance of persistent TCP traffic over UBR+ for various TCP options, UBR+ drop policies, buffer sizes and link delays. In this contribution, we extend the previous studies to WWW traffic. We also introduce another TCP model, NewReno [HOE96][FALL96], into our suite of TCP enhancements.

In this study, using our WWW traffic model, we perform full factorial simulations [JAIN91] involving

- **TCP flavors**: Vanilla, Fast Retransmit Recovery (Reno), NewReno and SACK
- **UBR+ drop policies**: Early Packet Drop (EPD) and Selective Drop (SD)
- **Propagation delays**: Satellite (Single-hop GEO, multiple-hop LEO/single-hop MEO) and WAN delays
- **Buffer sizes**: We use three buffer sizes approximately corresponding to 0.5, 1, and 2 times the round trip delay-bandwidth products

The simulation results are analyzed using ANOVA techniques presented in [JAIN91], and briefly described in Section 8.

Section 2 briefly discusses the key results of our previous work in TCP over UBR+. Section 3 presents an overview of the TCP enhancements, especially NewReno and SACK. Section 4 briefly overviews two UBR+ drop policies, EPD and Selective Drop, used in our simulations. We then describe our WWW model in Section 5. Finally, we present our simulation experiments, techniques, performance metrics, results and analysis.

2 Previous Work

In our past work, we have studied the performance of TCP over UBR+ and GFR for persistent TCP traffic. Studies have shown that TCP results in poor performance over UBR. Performance was measured in terms of efficiency and fairness. TCP performance over UBR can be improved by enhanced end-system policies as well as switch buffer management policies (or drop policies). The results for **persistent TCP** over UBR+ can be summarized as follows [GOYAL97a][GOYAL97b][GOYAL98]:

- TCP performance over UBR can be improved by TCP enhancements, intelligent drop policies, guaranteed rate and sufficient buffer sizing.
- For low delay networks, intelligent drop policies provide the most improvement in performance.
- For long delay networks, end system policies have a more significant effect than drop policies. TCP SACK results in the best performance[1].
- Providing a guaranteed rate to the UBR service categories significantly improves the performance of TCP over UBR+ especially for long delay networks.

[1] In our previous work, we did not assess the performance of NewReno by simulation.

- For satellite networks with Selective Drop and TCP SACK, a buffer size of 0.5 RTT*bandwidth is sufficient for high performance even for a large number of TCP sources. Although higher buffer sizes improve the performance, the improvement is small. Lower buffer sizes decrease the performance significantly (both efficiency and fairness).

3 TCP Enhancements

TCP uses a window-based flow control and uses it also to limit the number of segments in the network. "Vanilla TCP" consists of the slow start and congestion avoidance phases for congestion control. It detects segment losses by the retransmission timeout. Coarse granularity of timeouts are the primary cause of low TCP throughput over the UBR service. TCP Reno implements the fast retransmit and recovery algorithms that enable the connection to quickly recover from isolated segment losses [STEV97]. TCP Reno can efficiently recover from isolated segment losses, but not from bursty losses. As a result, TCP Reno results in poor efficiency for long latency configurations especially for low buffer sizes and persistent traffic.

3.1 TCP New Reno: A Modification to Fast Retransmit and Recovery

As indicated above, TCP Reno can not recover effectively from multiple packet drops. A modification to Reno, popularly known as NewReno was proposed by Jenny Hoe [HOE96] to overcome this shortcoming. She introduced a "fast-retransmit phase", in which the sender remembers the highest sequence number sent (RECOVER) when the fast retransmit is first triggered. After the first unacknowledged packet is retransmitted (when three duplicate ACKs are received), the sender follows the usual fast recovery algorithm and increases the CWND by one segment for each duplicate ACK it receives. When the sender receives an acknowledgment for the retransmitted packet, it checks if the ACK acknowledges all segments including RECOVER. If so, the sender exits the fast retransmit phase, sets its CWND to SSTHRESH and starts a linear increase (congestion avoidance phase). On the other hand, if the ACK is a partial ACK, i.e., it acknowledges the retransmitted segment and only some of the segments before RECOVER, then the sender immediately retransmits the next expected segment as indicated by the ACK. A partial ACK also resets the CWND to SSTHRESH. This continues until all segments including RECOVER are acknowledged. The NewReno retransmits one segment every round trip time until an ACK is received for RECOVER. This mechanism ensures that the sender will recover from N segment losses in N round trips.

Very recently, a description of the NewReno algorithm has appeared in [FLOY98]. This description recommends a modification in which on receiving a partial ACK the congestion window is reduced by amount of new data acknowledged and then incremented by 1 MSS. Another modification is suggested to avoid multiple fast retransmits. Our implementation of NewReno reflects the behavior as implemented in version ns-2.1b3 of *ns* simulator [NS] and does not have these modifications.

Another issue raised in [FLOY98] is whether the retransmit timer should be reset after each partial ACK or only after the first partial ACK. For satellite links, where retransmission timeout value is not much larger than the round trip time (RTT), the first option is better. If the retransmit timer is reset only after the first partial ACK, a retransmission timeout will be caused even for a small number of packets lost in a window. For satellite links with their long delays, a timeout is very costly. However, for WAN links, the retransmission timeout value is much larger than the RTT. For WAN links, if there are a number of packets lost in a window, it is better to timeout and retransmit all the packets using slow-start than to retransmit just 1 packet every RTT. In such a case, the second option is better. In our implementation, we reset the retransmit timer after each partial ACK.

Further, in our implementation of NewReno, we have incorporated two changes suggested by [HOE96]. The first suggestion is to set the initial SSTHRESH value to RTT-bandwidth product. The second is to send one new packet beyond RECOVER upon receiving 2 duplicate ACKs while in the fast-retransmit

phase (to keep the "flywheel" going). In our implementation, on receiving a partial ACK, a single packet is retransmitted. Since the TCP delay ACK timer is NOT set, all segments are ACKed as soon as they are received.

3.2 SACK TCP: Selective Acknowledgements

TCP with Selective Acknowledgments (SACK TCP) has been proposed to efficiently recover from multiple segment losses [MATH96]. In SACK TCP, acknowledgments contain additional information about the segments that have been received by the destination. When the destination receives out-of-order segments, it sends duplicate ACKs (SACKs) acknowledging the out-of-order segments it has received. From these SACKs, the sending TCP can reconstruct information about the segments not received at the destination. On receiving three duplicate ACKs, the sender retransmits the first lost segment and enters "fast-retransmit" phase as in NewReno. The CWND is set to half its current value. SSTHRESH is set to the new value of CWND and the highest sequence number sent so far is recorded in RECOVER. As in NewReno, the sender does not come out of the fast-retransmit phase until it has received the ACK for RECOVER. However, in the fast-retransmit phase if allowed by the window, the sending TCP uses the SACK information to retransmit lost segments before sending any new data. A sender implementing NewReno can retransmit only one lost segment every RTT. Thus it recovers from N segment losses in N RTTs. However, a sender implementing SACK can recover from N segment losses much faster.

Our implementation of SACK is based on the description in [FALL96][FLOY95][MATH96]. The SACK is sent whenever out-of-sequence data is received. All duplicate ACKs contain the SACK option. The receiver keeps track of all the out-of-sequence data blocks received. When the receiver generates a SACK, the first SACK block specifies the block of data formed by the most recently received data segment. This ensures that the receiver provides the most up to date information to the sender. After the first SACK block, the remaining blocks can be filled in any order.

The sender keeps a table of all the segments sent but not ACKed. When a segment is sent, it is entered into the table. When the sender receives an ACK with the SACK option, it marks all the segments specified in the SACK option blocks as SACKed. The entries for each segment remain in the table until the segment is ACKed. When the sender receives three duplicate ACKs, it retransmits the first unacknowledged packet and enters fast-retransmit phase. During the fast retransmit phase, when the sender is allowed to send, it first tries to retransmit the holes in the SACK blocks before sending any new segments. When the sender retransmits a segment, it marks the segment as retransmitted in the table. If a retransmitted segment is lost, the sender times out and performs slow start. When a timeout occurs, the sender resets the SACK table.

During the fast retransmit phase, the sender maintains a variable called "PIPE" that indicates the number of bytes currently in the network. When the third duplicate ACK is received, PIPE is set to the value of CWND - 3 segments and CWND is reduced by half. For every subsequent duplicate ACK received, PIPE is decremented by one segment because the ACK denotes a packet leaving the network. The sender sends data (new or retransmitted) only when PIPE is less than CWND value. This implementation is equivalent to inflating the CWND by one segment for every duplicate ACK and sending segments if the number of unacknowledged bytes is less than the congestion window value.

When a segment is sent, PIPE is incremented by one segment. When a partial ACK is received, PIPE is decremented by two. The first decrement is because the partial ACK represents a retransmitted segment leaving the pipe. The second decrement is done because the original segment that was lost, and had not been accounted for, is now actually considered to be lost.

4 UBR+ Drop Policies

The basic UBR service can be enhanced by implementing intelligent drop policies at the switches. In this study, we have used EPD and Selective Drop as drop policies.

4.1 Early Packet Discard (EPD)

The Early Packet Discard policy [ROMA95] maintains a threshold R, in the switch buffer. When the buffer occupancy exceeds R, then all new incoming packets are dropped. Partially received packets are accepted if possible.

The drop threshold R should be chosen so that on crossing the threshold, the switch has enough buffer left to accept cells of all incomplete packets currently in the buffer. However, if the number of VCs passing through the switch is large, it is possible that the entire buffer may not be enough to store one complete packet of each VC.

4.2 Selective Drop (SD)

The Selective Drop policy [GOYAL98] uses per-VC accounting, i.e., keeps track of current buffer utilization of each active UBR VC. A UBR VC is called "active" if it has at least one cell currently buffered in the switch. The total buffer occupancy, X, is allowed to grow until it reaches a threshold R, maintained as a fraction of the buffer capacity K. A fair allocation is calculated for each active VC, and if the VC's buffer occupancy X_i exceeds its fair allocation, its subsequent incoming packet is dropped. Mathematically, in the Selective Drop scheme, an active VC's entire packet is dropped if

$$(X > R) \text{ AND } (X_i > Z \ X/N_a)$$

where N_a is the number of active VCs and Z is another threshold parameter ($0 < Z <= 1$) used to scale the effective drop threshold.

5 WWW Traffic Model

The WWW uses Hypertext Transfer Protocol (HTTP). HTTP uses TCP/IP for communication between WWW clients and WWW servers [LEE96]. Modeling of the WWW traffic is difficult because of the changing nature of web traffic. In this section, we outline our model and the inherent assumptions.

5.1 Implications of the HTTP/1.1 standard

The main difference between version 1.1 of the Hypertext Transfer Protocol, HTTP/1.1 [FIEL97], and earlier versions is the use of persistent TCP connections as the default behavior for all HTTP connections. In other words, a new TCP connection is not set up for each HTTP/1.1 request. The HTTP client and the HTTP server assume that the TCP connection is persistent until a *Close* request is sent in the HTTP Connection header field.

Another important difference between HTTP/1.1 and earlier versions is that the HTTP client can make multiple requests without waiting for responses from the server (called *pipelining*). Earlier models were *closed-loop* in the sense that each request needed a response before the next request could be sent.

5.2 WWW Server Model

Our WWW traffic arrival model is an extension of that specified in SPECweb96 benchmark [SPEC96]. In our model, a WWW server, on receiving a request from a WWW client sends some data back. The amount of data to be sent (the requested file size) is determined by classifying the client request into one of five classes (Class 0 through Class 4), shown in Table 1. As shown in the table, 20% of the requests are classified as Class 0 requests, i.e., less than 1 KB of data is sent in the response. Similarly 28% of file requests are classified as Class 1 requests and so on. We model our WWW servers as infinitely fast servers, i.e., the response is generated as soon as the request is received.

Table 1 WWW Server File Size Classes

Class	*File Size Range*	*Frequency Of Access*
Class 0	0 - 1 KB	20 %
Class 1	1 KB - 10 KB	28 %
Class 2	10 KB – 100 KB	40 %
Class 3	100 KB - 1 MB	11.2 %
Class 4	1 MB - 10 MB	0.8 %

There are nine discrete sizes in each class (e.g. Class 1 has 1 KB, 2 KB, up to 9 KB and Class 2 has 10 KB, 20 KB, through 90 KB and so on.). Within a class, one of these nine file sizes is selected according to a Poisson Distribution with mean 5. The model of discrete sizes in each class is based on the SPECweb96 benchmark [SPEC96]. The key differences from the SPEC model are

(i) the assumptions of an infinite server, i.e. no processing time taken by server for a client request, and

(ii) a new class of file sizes (1 MB - 10 MB), which allows us to model file sizes larger than those in the SPEC benchmark and the corresponding change in the percentage distribution of client requests into server file size classes.

The reason for the new class of file sizes is to model downloading of large software and offline browsing of search results. The percentages of requests falling into each of file size classes have been changed so that average requested file size is around 120 KB, as opposed to 15 KB in SPECweb96 model. We believe the new figure better represents the current WWW traffic scenario. The reason for having 20% of the requests classified as Class 0 requests is explained in next sub-section.

5.3 WWW Client Model

The HTTP-model in [MAH97] describes an empirical model of WWW clients based on observations in a LAN environment. Specifically, a typical client is observed to make, on the average, four HTTP GET requests for a single document. Multiple requests are needed to fetch inline images, if any. With the introduction of JAVA scripts in web pages, additional accesses maybe required to fetch the scripts. Therefore, we use five as the average number of HTTP GET requests. In our model, a WWW client makes 1 to 9 requests for a single document, The number is Poisson distributed around a mean of 5.

These requests are separated by a random time interval between 100 ms to 500 ms. Caching effects at the clients are ignored.

Typically, the first request from an HTTP client accesses the index page (plain text), which is of size 1 KB or less. Since every fifth request is expected to be an index page access, WWW server classifies 20% (= 1/5) of the client requests as Class 0 requests and sends 1 KB or less data in the response.

We also model a time lag between batches of requests (presumably for the same document), that corresponds to the time taken by the user to request a new document, as a constant, 10 seconds. While this may be too short a time for a human user to make decisions, it also weights the possibility of offline browsing where the inter-batch time is much shorter.

We do not attempt to model user behavior across different servers. The main purpose of using this simplistic model is to approximate the small loads offered by individual web connections, and to study the effects of aggregation of such small loads on the network.

6 Simulation Experiments

Figure 1 shows the configuration used in all our simulations. The configuration consists of 100 WWW clients being served by 100 WWW servers, one server for each client. Both WWW clients and servers use underlying TCP connections for data transfer. The switches implement the UBR+ service with optional drop policies described earlier. The following subsections describe various configuration parameters, TCP parameters, and switch parameters used in the simulations.

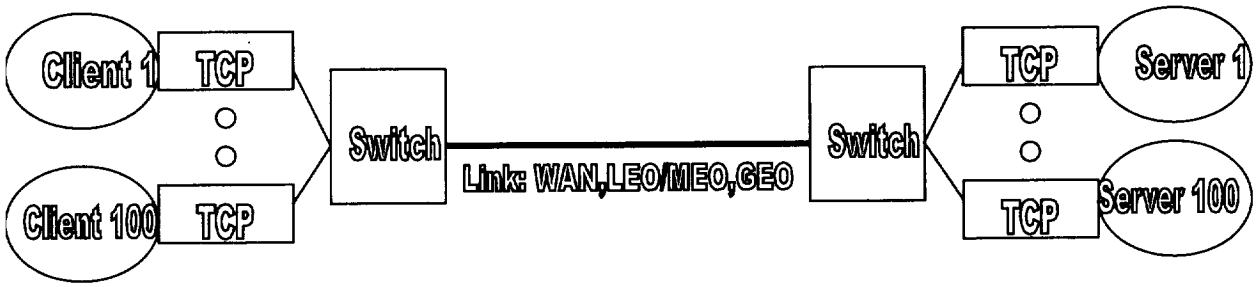


Figure 1 Simulation Configuration with 100 WWW Client-Server Connections

6.1 Configuration Parameters

- Links connecting server/client TCPs to switches have a bandwidth of 155.52 Mbps (149.76 Mbps after SONET overhead), and a one way delay of 5 microseconds.

- The link connecting the two switches has a bandwidth of 45Mbps (T3). It simulates one of three scenarios: a WAN, a multiple hop LEO/single hop MEO or a GEO link. The corresponding one-way link delays are 5 ms, 100 ms and 275 ms, respectively.

- Since the propagation delays on the links connecting client/server TCPs to switches are negligible compared to the delay on the inter-switch link, the round trip times (RTTs) due to propagation delay are 10 ms, 200 ms and 550 ms for WAN, LEO/MEO and GEO, respectively.

- All simulations run for 100 secs. Since every client makes a new set of requests every 10 secs, the simulations run for 10 cycles of client requests.

6.2 TCP Parameters

- Underlying TCP connections send data as specified by the client/server applications. A WWW client asks its TCP to send a 128 byte packet as a request to the WWW server TCP.

- TCP maximum segment size (MSS) is set to 1024 for WAN links and 9180 for LEO/MEO and GEO links.

- TCP timer granularity is set to 100 ms.

- TCP maximum receiver window size is chosen so that it is always greater than RTT-bandwidth product of the path. Such a value of receiver window ensures that receiver window does not prevent sending TCPs from filling up the network pipe. For WAN links (10 ms RTT due to propagation delay), the default receiver window size of 64K is sufficient. For MEO links (200 ms RTT), RTT-bandwidth product is 1,125,000 bytes. By using the TCP window scaling option and having a window scale factor of 5, we achieve an effective receiver window of 2,097,120 bytes. Similarly, for GEO links (550 ms RTT), the RTT-bandwidth product is 3,093,750 bytes. We use a window scale factor of 6 to achieve an effective receiver window of 4,194,240 bytes.

- TCP "Silly Window Syndrome Avoidance" is disabled because in WWW traffic many small segments (due to small request sizes, small file sizes or last segment of a file) have to be sent immediately.

- It has been proposed in [HOE96] that instead of having a fixed initial SSTHRESH of 64 KB, the RTT-bandwidth product of the path should be used as initial SSTHRESH. In our simulations, we have implemented this. Hence, the initial SSTHRESH values for WAN, MEO and GEO links are 56,250, 1,125,000 and 3,093,750 bytes respectively.

- The TCP delay ACK timer is NOT set. Segments are ACKed as soon as they are received.

6.3 Switch Parameters

- The drop threshold R is 0.8 for both drop policies - EPD and SD. For SD simulations, threshold Z also has a value 0.8.

- We use three different values of buffer sizes in our experiments. These buffer sizes approximately correspond to 0.5, 1, and 2 RTT - bandwidth products. For WAN delays, the largest buffer size is 2300 cells. This is a little more than the 2 RTT - bandwidth product. The reason for selecting 2300 is that this is the smallest buffer size that can hold one complete packet (MSS=1024 bytes) for each of the 100 TCP connections. For WAN, 0.5 RTT and 1 RTT buffers are not sufficient to hold one packet from each of the 100 TCPs. This problem will also occur in MEO and GEO TCPs if the number of TCPs is increased. Some preliminary analysis has shown that the buffer size required for good performance may be related to the number of active TCP connections as well as the RTT-bandwidth product. Further research needs to be performed to provide conclusive results of this effect. Table 2 shows the switch buffer sizes used in the simulations.

Table 2 Switch Buffer Sizes used for Simulations

Link Type (RTT)	RTT-bandwidth product (cells)	Switch Buffer Sizes (cells)
WAN (10 ms)	1062	531, 1062, 2300
Multiple-Hop LEO/Single-Hop MEO (200 ms)	21230	10615, 21230, 42460
Single-Hop GEO (550 ms)	58380	29190, 58380, 116760

7 Performance Metrics

The performance of TCP is measured by the efficiency and fairness index which are defined as follows. Let x_i be the throughput of the ith TCP connection ($1 \le i \le 100$). Let C be the maximum TCP throughput achievable on the link. Let E be the efficiency of the network. Then, E is defined as

$$E = \frac{\sum_{i=1}^{i=N} x_i}{C}$$

where N = 100 and Σx_i is sum of all 100 server throughputs.

The TCP throughput values are measured at the client TCP layers. Throughput is defined as the highest sequence number in bytes received at the client from the server divided by the total simulation time. The results are reported in Mbps.

Due to overheads imposed by TCP, IP, LLC and AAL5 layers, the maximum possible TCP over UBR throughput depends on the TCP maximum segment size (MSS). For MSS = 1024 bytes (on WAN links), the ATM layer receives 1024 bytes of data + 20 bytes of TCP header + 20 bytes of IP header + 8 bytes of LLC header + 8 bytes of AAL5 trailer. These are padded to produce 23 ATM cells. Thus, each TCP segment of 1024 bytes results in 1219 bytes at the ATM layer. Thus, the maximum possible TCP throughput C is 1024/1219 = 84%. This results in 37.80 Mbps approximately on a 45 Mbps link. Similarly, for MSS = 9180 bytes (on MEO, GEO links), C is 40.39 Mbps approximately. Since, the "Silly Window Syndrom Avoidance" is disabled (because of WWW traffic), some of the packets have less than 1 MSS of data. This decreases the value of C a little. However, the resulting decrease in the value of C has an insignificant effect on the overall efficiency metric.

In all simulations, the 45 Mbps(T3) link between the two switches is the bottleneck. The average total load generated by 100 WWW servers is 48 Mbps[2].

---

[2] A WWW server gets on average 5 client requests every 10 s and sends on average 120 kB of data for each request. This means that on average a WWW server schedules 60 kBps i.e. 480 kbps of data. Hence average total load generated by 100 WWW servers is 48 Mbps.

We measure fairness by calculating the Fairness Index *F* defined by:

$$F = \frac{\left(\sum_{i=1}^{i=N} x_i/e_i\right)^2}{N \times \sum_{i=1}^{i=N} (x_i/e_i)^2}$$

where N = 100 and e_i is the expected throughput for connection i. In our simulations, e_i is the max-min fair share that should be allocated to server i. On a link with maximum possible throughput *C*, the fair share of each of the 100 servers is *C*/100. Let S_i be the maximum possible throughput that a server can achieve, calculated as the total data scheduled by the server for the client divided by simulation time.

For all i's for which $S_i < C/100$, $e_i = S_i$, i.e., servers that schedule less than their fair share are allocated their scheduled rates. This determines the first iteration of the max-min fairness calculation. These e_i's are subtracted from C, and the remaining capacity is again divided in a max-min manner among the remaining connections. This process is continued until all remaining servers schedule more than the fair share in that iteration, for those servers e_i = the fairshare.

8 Simulation Analysis and Results

In this section, we present a statistical analysis of simulation results for WAN, multiple hop LEO/single hop MEO and GEO links and draw conclusions about optimal choices for TCP flavor, switch buffer sizes and drop policy for these links. The analysis techniques we have used here are described in detail in [JAIN91]. The next subsection gives a brief description of these techniques. The following subsections present simulation results for WAN, LEO/MEO, and GEO links, respectively.

8.1 Analysis Technique

The purpose of analyzing results of a number of experiments is to calculate the individual effects of contributing factors and their interactions. These effects can also help us in drawing meaningful conclusions about the optimum values for different factors. In our case, we have to analyze the effects of the TCP flavors, buffer sizes and drop policies in determining the efficiency and fairness for WAN, MEO and GEO links. Thus, we have 3 factors: TCP flavor, switch buffer size and drop policy. The values a factor can take are called 'levels' of the factor. For example, EPD and SD are two levels of the factor 'Drop Policy'. Table 3 lists the factors and their levels used in our simulations.

Table 3 Factors and Levels in simulations

Factor	Levels			
TCP flavor	Vanilla	Reno	NewReno	SACK
Switch drop policy	EPD		SD	
Switch buffer size	0.5 RTT[3]	1 RTT	2 RTT	

[3] Here onwards, when we say 1 RTT worth of buffer, we mean a buffer size equal to the product of RTT and link bandwidth in terms of cells.

The analysis is done separately for efficiency and fairness, and consists of the calculating the following terms:

1. **Overall mean**: This consists of the calculation of the overall mean 'Y' of the result (efficiency or fairness).

2. **Total variation**: This represents the variation in the result values (efficiency or fairness) around the overall mean 'Y'..

3. **Main effects**: These are the individual contributions of a level of a factor to the overall result. A particular main effect is associated with a level of a factor, and indicates how much variation around the overall mean is caused by the level. We calculate the main effects of 4 TCP flavors, 3 buffer sizes, and 2 drop policies.

4. **First order interactions**: These are the interaction between levels of two factors. In our experiments, there are first order interactions between each TCP flavor and buffer size, between each drop policy and TCP flavor, and between each buffer size and drop policy.

5. **Allocation of variation**: This is used to explain how much each effect contributes to the total variation. An effect (a factor or interaction), which explains a large fraction of the total variation, is said to be important.

6. **Overall standard error**: This represents the experimental error associated with each result value. The overall standard error is also used in the calculation of the confidence intervals for each effect.

7. **Confidence intervals for main effects**: The 90% confidence intervals for each main effect are calculated. If a confidence interval contains 0, then the corresponding level of the factor is not statistically significant. If confidence intervals of two levels overlap, then the effects of both levels are assumed to be similar.

The first step of the analysis is the calculation of the overall mean 'Y' of all the values. The next step is the calculation of the individual contributions of each level 'a' of factor 'A', called the 'Main Effect'. The 'Main Effect' of 'A' at level 'a' is calculated by subtracting the overall mean 'Y' from the mean of all results with 'a' as the value for factor 'A'. The 'Main Effects' are calculated in this way for each level of each factor.

We then calculate the interactions between two factors. The interaction between levels of two factors is called 'First-order interaction'. For calculating the interaction between level 'a' of factor 'A' and level 'b' of factor 'B', an estimate is calculated for all results with 'a' and 'b' as values for factors 'A' and 'B'. This estimate is the sum of the overall mean 'Y' and the 'Main Effects' of levels 'a' and 'b'. This estimate is subtracted from the mean of all results with 'a' and 'b' as values for factors 'A' and 'B' to get the 'Interaction' between levels 'a' and 'b'. Although one could continue computing second and higher order interactions, we limit our analysis to 'First-order interactions' only. Higher order interactions are assumed to small or negligible.

We then perform the calculation of the 'Total Variation' and 'Allocation of Variation'. First, the value of the square of the overall mean 'Y' is multiplied by the total number of results. This value is subtracted from the sum of squares of individual results to get the 'Total Variation' among the results. The next step is the 'Allocation of Total Variation' to individual 'Main Effects' and 'First-order interactions'. To calculate the variation caused by a factor 'A', we take the sum of squares of the main effects of all levels of 'A' and multiply this sum with the number of experiments conducted with each level of 'A'. For example, to calculate the variation caused by TCP flavor, we take the sum of squares of the main effects of all its levels (Vanilla, Reno, NewReno and SACK) and multiply this sum by 6 (with each TCP flavor

we conduct 6 different simulations involving 3 buffer sizes and 2 drop policies). In this way, the variation caused by all factors is calculated. To calculate the variation caused by first-order interaction between two factors 'A' and 'B', we take the sum of squares of all the first-order interactions between levels of 'A' and 'B' and multiply this sum with the number of experiments conducted with each combination of levels of 'A' and 'B'.

The next step of the analysis is to calculate the overall standard error for the results. This value requires calculation of individual errors in results and the degrees of freedom for the errors. For each result value, an estimate is calculated by summing up the overall mean 'Y', main effects of the parameter levels for the result and their interactions. This estimate is subtracted from the actual result to get the error 'e_i' for the result.

If a factor 'A' has 'N_A' levels, then the total number of degrees of freedom is $\Pi(N_A)$. Thus, for our analysis, the total number of degrees of freedom is $4 \times 2 \times 3 = 24$. The degrees of freedom associated with the overall mean 'Y' is 1. The degrees of freedom associated with 'main effects' of a factor 'A' are '$N_A - 1$'. Thus, degrees of freedom associated with all 'main effects' are $\Sigma(N_A - 1)$. Similarly, the degrees of freedom associated with the first-order interaction between 2 factors 'A' and 'B' are $(N_A - 1) \times (N_B - 1)$. Thus, degrees of freedom associated with all first-order interactions are $\Sigma(N_A - 1) \times (N_B - 1)$, with the summation extending over all factors. In our analysis, the degrees of freedom associated with all 'main effects' are $3 + 1 + 2 = 6$ and the degrees of freedom associated with all first-order interactions are $(3 \times 1) + (3 \times 2) + (1 \times 2) = 11$.

Since we use the overall mean 'Y', the main effects of individual levels and their first-order interactions to calculate the estimate, the value of the degrees of freedom for errors 'd_e' is calculated as follows:

$$d_e = \prod(N_A) - 1 - \sum(N_A - 1) - \sum(N_A - 1) \times (N_B - 1)$$

In our case, $d_e = 24 - 1 - 6 - 11 = 6$.

To calculate the overall standard error 's_e', the sum of squares of all individual errors 'e_i' is divided by the number of degrees of freedom for errors 'd_e' (6 in our case). The square root of the resulting value is the overall standard error.

$$s_e = \sqrt{(\sum e_i^2)/d_e}$$

Finally, based on the overall standard error, we calculate the 90% confidence intervals for all 'main effects' of each factor. For this purpose, we calculate the standard deviation 's_A' associated with each factor 'A' as follows:

$$s_A = s_e \times \sqrt{(N_A - 1)/\prod(N_A)}$$

Here, 'N_A' is the number of levels for factor 'A' and $\Pi(N_A)$ is the total number of degrees of freedom.

The variation around the 'main effect' of all levels of a factor 'A' to get a 90% confidence level is given by the standard deviation 's_A' multiplied by $t[0.95,d_e]$, where $t[0.95,d_e]$ values are quantiles of the t distribution [JAIN91].

Hence, if 'ME_a' is the value of the main effect of level 'a' of factor 'A', then the 90% confidence interval for 'ME_a' is $\{ME_a \pm s_A \times t[0.95,d_e]\}$. The main effect value is statistically significant only if the confidence interval does not include 0.

This was only a brief description of the techniques used to analyze simulation results. The detailed description of the analysis method can be found in [JAIN91].

8.2 Simulation Results for WAN links

Table 4 presents the individual efficiency and fairness results for WAN links. Table 5 shows the calculation of 'Total Variation' in WAN results and 'Allocation of Variation' to main effects and first-order interactions. Table 6 shows the 90% confidence intervals for the main effects. A negative value of main effect implies that the corresponding level of the factor decreases the overall efficiency and vice versa. If a confidence interval encloses 0, the corresponding level of the factor is assumed to be not significant in determining performance.

Table 4 Simulation Results for WAN links

Drop Policy	TCP Flavor	Buffer = 0.5 RTT		Buffer = 1 RTT		Buffer = 2 RTT	
		Efficiency	Fairness	Efficiency	Fairness	Efficiency	Fairness
EPD	Vanilla	0.4245	0.5993	0.5741	0.9171	0.7234	0.9516
	Reno	0.6056	0.8031	0.7337	0.9373	0.8373	0.9666
	NewReno	0.8488	0.8928	0.8866	0.9323	0.8932	0.9720
	SACK	0.8144	0.7937	0.8948	0.8760	0.9080	0.8238
SD	Vanilla	0.4719	0.6996	0.6380	0.9296	0.8125	0.9688
	Reno	0.6474	0.8230	0.8043	0.9462	0.8674	0.9698
	NewReno	0.8101	0.9089	0.8645	0.9181	0.8808	0.9709
	SACK	0.7384	0.6536	0.8951	0.8508	0.9075	0.8989

Table 5 Allocation of Variation for WAN Efficiency and Fairness Values

Component	Sum of Squares		%age of Variation	
	Efficiency	Fairness	Efficiency	Fairness
Individual Values	14.6897	18.6266		
Overall Mean	14.2331	18.3816		
Total Variation	0.4565	0.2450	100	100
Main Effects:				
TCP Flavor	0.2625	0.0526	57.50	21.49
Buffer Size	0.1381	0.1312	30.24	53.55
Drop Policy	0.0016	0.0002	0.34	0.09
First-order Interactions:				
TCP Flavor-Buffer Size	0.0411	0.0424	8.99	17.32
TCP Flavor-Drop Policy	0.0104	0.0041	2.27	1.68
Buffer Size-Drop Policy	0.0015	0.0009	0.33	0.38
Standard Error, s_e = 0.0156(For Efficiency), 0.0472(For Fairness)				

Table 6 Main Effects and their Confidence Intervals for WAN

Factor	Main Effect		Confidence Interval	
	Efficiency	Fairness	Efficiency	Fairness
TCP Flavor:				
Vanilla	-0.1627	-0.0308	(-0.1734,-0.1520)	(-0.0632,0.0016)
Reno	-0.0208	0.0325	(-0.0315,-0.0101)	(0.0000, 0.0649)
NewReno	0.0939	0.0573	(0.0832,0.1046)	(0.0248, 0.0898)
SACK	0.0896	-0.0590	(0.0789,0.1003)	(-0.0914, -0.0265)
Buffer Size:				
0.5 RTT	-0.1000	-0.1034	(-0.1087,-0.0912)	(-0.1299,-0.0769)
1 RTT	0.0163	0.0382	(0.0076,0.0250)	(0.0117, 0.0647)
2 RTT cells	0.0837	0.0651	(0.0749,0.0924)	(0.0386, 0.0916)
Drop Policy:				
EPD	-0.0081	-0.0030	(-0.0142, -0.0019)	(-0.0217,0.0157)
SD	0.0081	0.0030	(0.0019,0.0142)	(-0.0157, 0.0217)

8.2.1 Analysis of Efficiency values: Results and Observations

The following conclusions can be drawn from the above tables:

1. TCP flavor explains 57.5% of the variation and hence is the major factor in determining efficiency. It can be established from confidence intervals of effects of different TCP flavors that NewReno and SACK have better efficiency performance than Vanilla and Reno. Since the confidence intervals of effects of SACK and NewReno overlap, we cannot say that one performs better than the other. Confidence intervals for the effects of Vanilla and Reno suggest that Reno performs better than Vanilla.

2. Buffer size explains 30.24% of the variation and hence is the next major determinant of efficiency. Confidence intervals for effects of different buffer sizes clearly indicate that efficiency increases substantially as buffer size is increased. However, if we look at individual efficiency values, it can be noticed that only Vanilla and Reno get substantial increase in efficiency as buffer size is increased from 1 RTT to 2 RTT.

3. The interaction between buffer size and TCP flavor explains 8.99% of the variation. The large interaction is because of the fact that only Vanilla and Reno show substantial gains in efficiency as the buffer size is increased from 1 RTT to 2 RTT. For SACK and NewReno, increasing buffer sizes from 1 RTT to 2 RTT does not bring much increase in efficiency. This indicates that SACK and NewReno can tolerate the level of packet loss caused by a buffer size of 1 RTT.

4. Though the variation explained by drop policy is negligible, it can be seen that for Vanilla and Reno, SD results in better efficiency than EPD for the same buffer size. This is because for EPD, after crossing the threshold R, all new packets are dropped and buffer occupancy does not increase much beyond R. However for SD, packets of VCs with low buffer occupancy are still accepted. This allows the buffer to be utilized more efficiently and fairly and to better efficiency as well as fairness.

5. For NewReno and SACK, the efficiency values are similar for EPD and SD for same buffer size. This is because NewReno and SACK are much more tolerant of packet loss than Vanilla and Reno. Thus the small decrease in number of packets dropped due to increased buffer utilization does not cause a significant increase in efficiency.

6. It can be noticed from individual efficiency values that SACK generally performs a little better than NewReno except when buffer size is very low (0.5 RTT). Better performance of NewReno for very low buffer size can be explained as follows. Low buffer size means that a large number of packets are dropped. When in fast retransmit phase, NewReno retransmits a packet for every partial ACK received. However, SACK does not retransmit any packet till *PIPE* goes below *CWND* value. A large number of dropped packets mean that not many duplicate or partial ACKs are forthcoming. Hence *PIPE* may not reduce sufficiently to allow SACK to retransmit all the lost packets quickly. Thus, SACK's performance may perform worse than NewReno under extreme congestion.

We conclude that SACK and NewReno give best performance in terms of efficiency for WAN links. For NewReno and SACK, a buffer size of 1 RTT is sufficient for getting close to best efficiency with either EPD or SD as the switch drop policy.

8.2.2 Analysis of Fairness values: Results and Observations

1. Buffer size largely determines fairness as 53.55 % of the variation is explained by the buffer size. Confidence intervals for effects of buffer sizes suggest that the fairness increases substantially as buffer size is increased from 0.5 RTT to 1 RTT. Since confidence intervals for buffers of 1 RTT and 2 RTTs overlap, it cannot be concluded that 2 RTT buffers result in better performance than 1 RTT buffers.

2. TCP flavor is the next major factor in determining fairness as it explains 21.49 % of the variation. Confidence intervals for effects of TCP flavor on fairness, clearly suggest that NewReno results in the best fairness and SACK results in the worst fairness.

3. SD only increases fairness for low buffer sizes. Overall, both the allocation of variation to drop policy, and confidence intervals for effects of SD and EPD suggest that SD does not result in higher fairness when compared to EPD for bursty traffic in WAN links unless buffer sizes are small.

8.3 Simulation Results for MEO links

Table 7 presents the individual efficiency and fairness results for MEO links. Table 8 shows the calculation of 'Total Variation' in MEO results and 'Allocation of Variation' to main effects and first-order interactions. Table 9 shows the 90% confidence intervals for main effects.

Table 7 Simulation Results for MEO Links

Drop Policy	TCP Flavor	Buffer = 0.5 RTT		Buffer = 1 RTT		Buffer = 2 RTT	
		Efficiency	Fairness	Efficiency	Fairness	Efficiency	Fairness
EPD	Vanilla	0.8476	0.9656	0.8788	0.9646	0.8995	0.9594
	Reno	0.8937	0.9659	0.9032	0.9518	0.9091	0.9634
	NewReno	0.9028	0.9658	0.9105	0.9625	0.9122	0.9616
	SACK	0.9080	0.9517	0.9123	0.9429	0.9165	0.9487
SD	Vanilla	0.8358	0.9649	0.8719	0.9684	0.9009	0.9615
	Reno	0.8760	0.9688	0.8979	0.9686	0.9020	0.9580
	NewReno	0.8923	0.9665	0.8923	0.9504	0.8976	0.9560
	SACK	0.9167	0.9552	0.9258	0.9674	0.9373	0.9594

Table 8 Allocation of Variation for MEO Efficiency and Fairness Values

Component	Sum of Squares		%age of Variation	
	Efficiency	Fairness	Efficiency	Fairness
Individual Values	19.3453	22.1369		
Overall Mean	19.3334	22.1357		
Total Variation	0.0119	0.0012	100	100
Main Effects:				
TCP Flavor	0.0067	0.0003	56.75	29.20
Buffer Size	0.0026	0.0001	21.73	7.70
Drop Policy	0.0001	0.0001	0.80	6.02
First-order Interactions:				
TCP Flavor-Buffer Size	0.0016	0.0001	13.42	10.16
TCP Flavor-Drop Policy	0.0007	0.0003	6.11	22.60
Buffer Size-Drop Policy	0.0001	0.0001	0.53	6.03
Standard Error, s_e = 0.0036(For Efficiency), 0.0060(For Fairness)				

Table 9 Main Effects and Their Confidence Intervals for MEO

Factor	Mean Effect		Confidence Interval	
	Efficiency	Fairness	Efficiency	Fairness
TCP Flavor:				
Vanilla	-0.0251	0.0037	(-0.0276,-0.0226)	(-0.0004,0.0078)
Reno	-0.0005	0.0024	(-0.0030,0.0019)	(-0.0017,0.0065)
NewReno	0.0038	0.0001	(0.0013,0.0062)	(-0.0040,0.0042)
SACK	0.0219	-0.0062	(0.0194,0.0244)	(-0.0103,-0.0020)
Buffer Size:				
0.5 RTT	-0.0134	0.0027	(-0.0154,-0.0114)	(-0.0007,0.0060)
1 RTT	0.0016	-0.0008	(-0.0005,0.0036)	(-0.0042,0.0026)
2 RTT	0.0119	-0.0019	(0.0098,0.0139)	(-0.0052,0.0015)
Drop Policy:				
EPD	0.0020	-0.0017	(0.0006,0.0034)	(-0.0041,0.0007)
SD	-0.0020	0.0017	(-0.0034,-0.0006)	(-0.0007,0.0041)

8.3.1 Analysis of Efficiency values: Results and Observations

1. TCP flavor explains 56.75% of the variation and hence is the major factor in deciding efficiency value. Non overlapping confidence intervals for effects of TCP flavors clearly indicate that SACK results in best efficiency followed by NewReno, Reno and Vanilla. However, it should be noticed that difference in performance for different TCP flavors is not very large.

2. Buffer size explains 21.73% of the variation and hence is the next major determinant of efficiency. Confidence intervals for effects of different buffer sizes indicate that efficiency does increase but only slightly as buffer size is increased. However, Vanilla's efficiency increases by about 5% with increase in buffer size from 0.5 RTT to 2 RTT. The corresponding increase in efficiency for other TCP flavors is around 2% or less. This also explains the large interaction between buffer sizes and TCP flavors (explaining 13.42% of the total variation).

3. Drop policy does not cause any significant difference in efficiency values.

Thus, SACK gives best performance in terms of efficiency for MEO links. However, difference in performance for SACK and other TCP flavors is not substantial. For SACK, NewReno and FRR, the increase in efficiency with increasing buffer size is very small. For MEO links, 0.5 RTT is the optimal buffer size for all non-Vanilla TCP flavors with either EPD or SD as drop policy.

8.3.2 Analysis of Fairness values: Results and Observations

As we can see from individual fairness values, there is not much difference between fairness values for different TCP flaovrs, buffer sizes or drop policies. This claim is also supported by the fact that all 9 main effects have very small values, and for 8 of them, their confidence interval encloses 0. Thus, for MEO delays, 0.5 RTT buffer is sufficient for good fairness with any drop policy for all flavors of TCPs.

8.3.3 Simulation Results for GEO links

Table 10 presents the individual efficiency and fairness results for GEO links. Table 11 shows the calculation of 'Total Variation' in GEO results and 'Allocation of Variation' to main effects and first-order interactions. Table 12 shows the 90% confidence intervals for main effects.

Table 10 Simulation Results for GEO Links

Drop Policy	TCP Flavor	Buffer = 0.5 RTT		Buffer = 1 RTT		Buffer = 2 RTT	
		Efficiency	Fairness	Efficiency	Fairness	Efficiency	Fairness
EPD	Vanilla	0.7908	0.9518	0.7924	0.9365	0.8478	0.9496
	Reno	0.8050	0.9581	0.8172	0.9495	0.8736	0.9305
	NewReno	0.8663	0.9613	0.8587	0.9566	0.8455	0.9598
	SACK	0.9021	0.9192	0.9086	0.9514	0.9210	0.9032
SD	Vanilla	0.8080	0.9593	0.8161	0.9542	0.8685	0.9484
	Reno	0.8104	0.9671	0.7806	0.9488	0.8626	0.9398
	NewReno	0.7902	0.9257	0.8325	0.9477	0.8506	0.9464
	SACK	0.9177	0.9670	0.9161	0.9411	0.9207	0.9365

Table 11 Allocation of Variation for GEO Efficiency and Fairness Values

Component	Sum of Squares		%age of Variation	
	Efficiency	Fairness	Efficiency	Fairness
Individual Values	17.3948	21.4938		
Overall Mean	17.3451	21.4884		
Total Variation	0.0497	0.0054	100	100
Main Effects:				
TCP Flavor	0.0344	0.0008	69.16	14.47
Buffer Size	0.0068	0.0006	13.65	11.48
Drop Policy	0.0001	0.0001	0.25	2.31
First-order Interactions:				
TCP Flavor-Buffer Size	0.0037	0.0012	7.54	22.16
TCP Flavor-Drop Policy	0.0025	0.0014	4.96	26.44
Buffer Size-Drop Policy	0.0002	0.0001	0.41	1.45
Standard Error, s_e = 0.0182(For Efficiency), 0.0139(For Fairness)				

Table 12 Main Effects and Their Confidence Intervals for GEO

Factor	Mean Effect		Confidence Interval	
	Efficiency	Fairness	Efficiency	Fairness
TCP Flavor:				
Vanilla	-0.0295	0.0037	(-0.0420,-0.0170)	(-0.0058,0.0133)
Reno	-0.0252	0.0027	(-0.0377,-0.0127)	(-0.0068,0.0123)
NewReno	-0.0095	0.0034	(-0.0220,0.0030)	(-0.0062,0.0129)
SACK	0.0642	-0.0098	(0.0517,0.0768)	(-0.0194,-0.0003)
Buffer Size:				
0.5 RTT	-0.0138	0.0050	(-0.0240,-0.0036)	(-0.0029,0.0128)
1 RTT	-0.0099	0.0020	(-0.0201,0.0004)	(-0.0058,0.0098)
2 RTT	0.0237	-0.0070	(0.0134,0.0339)	(-0.0148,0.0009)
Drop Policy:				
EPD	0.0023	-0.0023	(-0.0049,0.0095)	(-0.0078,0.0033)
SD	-0.0023	0.0023	(-0.0095,0.0049)	(-0.0033,0.0078)

In the following 2 subsections, we present the results of analyzing efficiency and fairness values for GEO links.

8.3.4 Analysis of Efficiency values: Results and Observations

1. TCP flavor explains 69.16% of the variation and hence is the major factor in deciding efficiency value. Confidence intervals for effects of TCP flavors clearly indicate that SACK results in substantially better efficiency than other TCP flavors. Since confidence intervals overlap for NewReno, Reno and Vanilla, one can not be said to be better than other in terms of efficiency.

2. Buffer size explains 13.65% of the variation and interaction between buffer size and TCP flavors explains 7.54% of the variation. Confidence intervals for 0.5 RTT and 1 RTT buffer overlap, thus indicating similar performance. There is a marginal improvement in performance as buffer size is increased to 2 RTT. Vanilla and Reno show substantial efficiency gains as buffer size is increased from 1 RTT to 2 RTT. There is not much improvement for Vanilla and FRR when buffer is increased from 0.5 RTT to 1 RTT. Hence, in this case, 1 RTT buffer does not sufficiently reduce number of packets dropped to cause an increase in efficiency. However, for a buffer of 2 RTT, the reduction in number of dropped packets is enough to improve Vanilla and Reno's performance.

3. Drop policy does not have an impact in terms of efficiency as indicated by negligible allocation of variation to drop policy.

From the observations above, it can be concluded that SACK with 0.5 RTT buffer is the optimal choice for GEO links with either of EPD and SD as switch drop policy.

8.3.5 Analysis of Fairness values: Results and Observations

The conclusion here is similar to MEO delays. As we can see from individual fairness values, there is not much difference between fairness values for different TCP flavors, buffer sizes or drop policies. All 9 main effects have very small values, and for 8 of them, their confidence intervals enclose 0. Thus, for GEO delays, 0.5 RTT buffer is sufficient for good fairness with any drop policy for all types of TCPs.

8.4 Overall Analysis

It is interesting to notice how the relative behavior of different TCP flavors change as link delay increases.

As link delay increases, SACK clearly comes out to be superior than NewReno in terms of efficiency. For WAN, SACK and NewReno have similar efficiency values. For MEO, SACK performs a little better than NewReno and for GEO, SACK clearly outperforms NewReno. The reason for this behavior is that NewReno needs N RTTs to recover from N packet losses in a window whereas SACK can recover faster, and start increasing CWND again. This effect becomes more and more pronounced as RTT increases.

SD does not always lead to increase in fairness as compared to EPD. This result can again be attributed to nature of WWW traffic. SD accepts packets of only under-represented VCs after crossing the threshold R. For sufficient buffer size, many of these VCs are under represented in switch buffer because they do not have a lot of data to send. Thus, SD fails to cause significant increase in fairness.

It has been already concluded that for long delay links, end system policies are more important than switch drop policies in terms of efficiency and fairness [GOYAL98]. Results presented in this contribution confirm this conclusion for WWW traffic.

9 Summary

In this contribution we studied the effects of TCP mechanisms, UBR+ drop policies and buffer sizes on the performance of WWW traffic over satellite networks. The following overall conclusions can be made about the efficiency and fairness of WWW TCP traffic over ATM-UBR+ for long delay networks:

Efficiency

1. *End system policies*: SACK generally results in the best efficiency, especially as the delay increases. For lower delay and small buffer sizes, NewReno can perform better than SACK.

2. *Drop policies*: For lower delays (WAN), selective drop improves performance over EPD. As the delay increases, buffer sizes used in our experiments become larger, and selective drop does not have much effect.

3. *Buffer size*: Increasing buffer size increases performance, but the effect of buffer size in much more significant for lower delay.

Fairness

1. *End system policies*: SACK hurts fairness in lower delay (WAN) compared to NewReno. SACK and NewReno have similar fairness for higher delay.

2. *Drop policies*: Drop policies do not have much effect on long delay networks.

3. *Buffer size*: Increasing buffer sizes increases fairness, but for sufficiently large buffers this effect is negligible.

In summary, as delay increases, the marginal gains of end system policies become more important compared to the marginal gains of drop policies and larger buffers.

10 References

[FALL96] K. Fall, S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," Computer Communications Review, July 1996

[FIEL97] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1", RFC 2068, January 1997.

[FLOY95] S. Floyd, "Issues of TCP with SACK," Lawrence Berkeley Labs, Technical report, December 1995

[FLOY98] S. Floyd, T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," Internet Draft, November 1998, Available from ftp://ftp.ietf.org/internet-drafts/drafts-ietf-tcpimpl-newreno-00.txt

[GOYAL97a] Rohit Goyal, Raj Jain, Shiv Kalyanaraman, Sonia Fahmy, Bobby Vandalore, Xiangrong Cai, Seong-Cheol Kim, Sastri Kota, "Guaranteed Rate for Improving TCP Performance on UBR+ over Terrestrial and Satellite Networks," ATM Forum/97-0424, April 1997, http://www.cis.ohio-state.edu/~jain/atmf/a97-0424.htm

[GOYAL97b] Rohit Goyal, Raj Jain, Shivkumar Kalyanaraman, Sonia Fahmy, Bobby Vandalore, Sastri Kota, "TCP Selective Acknowledgments and UBR Drop Policies to Improve ATM-UBR Performance over Terrestrial and Satellite Networks", Proc. ICCCN97, Las Vegas, September 1997, pp17-27. http://www.cis.ohio-state.edu/~jain/papers/ic3n97.htm

[GOYAL98] Rohit Goyal, Raj Jain, Shivkumar Kalyanaraman, Sonia Fahmy, Bobby Vandalore, "Improving the Performance of TCP over the ATM-UBR Service," *Computer Communications*, Vol 21/10, 1998, http://www.cis.ohio-state.edu/~jain/papers/cc.htm

[HOE96] J.C. Hoe, "Improving the Start-up Behavior of a Congestion Control Scheme for TCP," Proceedings of SIGCOMM96, August 1996.

[KOTA97] Sastri Kota, R. Goyal, Raj Jain, "Satellite ATM Network Architectural Considerations and TCP/IP Performance," Proceedings of the 3rd K-A Band Utilization Conference, 1997, http://www.cis.ohio-state.edu/~jain/papers/kaband.htm

[JAIN91] R. Jain, The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Simulation, and Modeling, John Wiley & Sons Inc., 1991.

[LEE96] T. Berners-Lee, R. Fielding, H. Frystyk, "Hypertext Transfer Protocol - HTTP/1.0", RFC 1945, May 1996.

[MAH97] B.A. Mah, "An Empirical Model of HTTP Network Traffic," IEEE INFOCOM'97, April 1997.

[MATH96] M. Mathis, J. Madhavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgment Options," RFC 2018, October 1996.

[NS] ns Simulator, Available from http://www-mash.cs.berkeley.edu/ns

[ROMA95] A. Romanow, S. Floyd, "Dymnamics of TCP Traffic over ATM Networks", IEEE JSAC, May 1995.

[SPEC96] SPEC, "An Explanation of the SPECweb96 Benchmark," Available from http://www.specbench.org/osg/web96/webpaper.html

[STEV97] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," RFC 2001, January 1997.

VI. Large congestion window and the congestion avoidance phase

VI.A

See ICCN '97 paper under deliverable 2 above

VII. Optimizing the performance of SACK TCP

VII.A

We analyzed the performance of SACK TCP using delayed retransmit. It was found not to have any significant effect on the performance. No papers were published on this topic.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1999	Final Contractor Report

4. TITLE AND SUBTITLE

Design Issues for Traffic Management for the ATM UBR+ Service for TCP Over Satellite Networks

5. FUNDING NUMBERS

WU–650–32–5A–00
NAS3–97198

6. AUTHOR(S)

Raj Jain

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Ohio State University
2015 Heil Avenue
DL395
Columbus, Ohio 43210

8. PERFORMING ORGANIZATION REPORT NUMBER

E–11718

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
John H. Glenn Research Center at Lewis Field
Cleveland, Ohio 44135–3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR—1999-209158

11. SUPPLEMENTARY NOTES

Project Manager, Thomas vonDeak, Communications Technology Division, NASA Glenn Research Center, organization code 5610, (216) 433–3277.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Categories: 17 and 32 Distribution: Standard

This publication is available from the NASA Center for AeroSpace Information, (301) 621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

This project was a comprehensive research program for developing techniques for improving the performance of internet protocols over Asynchronous Transfer Mode (ATM) based satellite networks. Among the service categories provided by ATM networks, the most commonly used category for data traffic is the unspecified bit rate (UBR) service. UBR allows sources to send data into the network without any feedback control. The project resulted in the numerous ATM Forum contributions and papers.

14. SUBJECT TERMS

Communication satellite; Communication theory; Communication networks–packets

15. NUMBER OF PAGES
219

16. PRICE CODE
A09

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	