INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

A Study of TCP Behavior over UBR and ABR

Tarlok Birdi

A Major Report
In
The Department
Of
Computer Science

Presented In Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montreal, Quebec, Canada

September 1999

Abstract

A Study of TCP Behavior over UBR and ABR

Tarlok Birdi

The ability of TCP to provide its own congestion control mechanism allows it to effectively overcome segment loss and avoid congestion collapse. When running TCP over UBR with EPD, we obtain better throughput than with UBR alone. A UBR+EPD based sub-network delivers just as efficient throughput and fair allocation as an ABR+EPD based sub-network. In fact, the Cost-Benefit Ratio, which is an indicator for the packet loss to throughput efficiency, is similar in both scenarios. However, this is assuming that all TCP sources have the same Round Trip Time (RTT). When different RTT sources are implemented, the ABR+EPD sub-network provides better fairness. The final decision deals with enduring the cost of controlling the buffering scheme at the ATM network edge devices, or relying on whatever policy is in place on the ATM switch, albeit at a lower implementation cost.

Contents

List of Figures

List of Tables

List of Graphs

List of Abbreviations

AAL	ATM Adaptation Layer
ABR	Available Bit Rate
ACK	Acknowledgement
ATM	Asynchronous Transfer Mode
Basize	Buffer Allocation Size
Btag	Beginning Tag
CAC	Call Admission Control
CB Ratio	Cost-Benefit Ration
CBR	Constant Bit Rate
CDVT	Cell Delay Variation Tolerance
CIF	Cells in Frames
CLP	Cell Loss Priority
CLR	Cell Los Ratio
COM	Continuation of Message

CPI	Common Part Indicator
CRC	Cyclic Redundancy Code
CS-PDU	Convergence SubLayer Protocol Data Unit
CTI	Computer Telephony Integration
cwnd	congestion window
EFCI	Explicit Flow Congestion Indicator
EPD	Early Packet Discard
Etag	End Tag
FBA	Fair Buffer Allocation
FDDI	Fiber Distributed Data Interface
FIFO	First In First Out
FRR	Fast Retransmit and Fast Recovery
GFC	Generic Flow Control
HEC	Header Error Check
IP	Internet Protocol
LAN	Local Area Network

LANE	LAN Emulation
maxCTD	Maximum Cell Transfer Delay
MBS	Maximum Burst Size
MCR	Minimum Cell Rate
MID	Multiplexing Identifier
MPOA	Multiprotocol over ATM
mss	Maximum Segment Size
NNI	Network-Network Interface
noc	Number of Connections
nrt-VBR	NonRealtime VBR
PCR	Peak Cell Rate
PDU	Protocol Data Unit
pkts	packets
PVC	Permanent Virtual Connection
QoS	Quality of Service

RM	Resource Management
RR	Round Robin
RSVP	Resource Reservation Protocol
RTT	Round Trip Time
rt-VBR	Realtime VBR
rxt	retransmit
SCR	Sustained Cell Rate
SD	Selective Drop
SEQ	Sequence
SMDS	Switched Multi-Megabit Data Services
ssthresh	Slow Start Threshold
SVC	Switched Virtual Connection
TCP	Transmission Control Protocol
UBR	Unspecified Bit Rate
UNI	User-Network Interface
UPC	Usage Parameter Control

VBR	Variable Bit Rate
VC	Virtual Channel
VCI	Virtual Circuit Identifier
VPI	Virtual Path Identifier
WAN	Wide Area Network
WRR	Weighted Round Robin
X	Buffer Size Varying Factor

1 Introduction

Most computer communication applications today are running over TCP/IP and since ATM has gained wide implementation and is continuing to grow in the industry, it is very useful to study TCP performance over ATM. ATM is a good infrastructure for supporting TCP applications. It can allow TCP to fill the bandwidth pipe unused by higher-priority traffic, since TCP can dynamically react to the end users demand for more bandwidth, based upon availability. The two service categories that TCP can utilize are UBR and ABR.

UBR is a best-effort service category, which provides no specific quality of service or throughput guarantee. UBR uses a very basic cell discard policy that implements a buffer threshold. Once this threshold is exceeded, all lower priority cells are discarded. A cell priority bit is set in the ATM cell header. Switches will usually have large buffers when supporting TCP over UBR. An order of the product of the Round-Trip-Time (RTT) and bandwidth is usually used. During congestion conditions, TCP must take on the task of noticing cell loss via timeouts and retransmit missing packets, because ATM network devices do not notify the sender that loss has occurred. When a cell loss occurs it sets of a chain reaction of retransmissions of the current packet and all of the remaining packets, up to the end of the transmit window to be retransmitted. Considerable performance degradation is then experienced due to slow recovery process (TCP's Slow Start mechanism) and causes host time-outs. However, an intelligent drop policy like Early Packet Discard (EPD) may increase throughput and fairness. EPD is based on the observation that it is better to drop all cells from one packet then to randomly drop cells belonging to different packets, which is important when many TCP sources contend for bandwidth. Typical UBR applications include LAN Emulation, IP over ATM and non-mission-critical traffic.[21]

ABR works in cooperation with sources that can change their transmission rate in response to rate-based network feedback. This is performed in the context of closed-loop congestion control. The goal is to dynamically provide access to bandwidth currently not in use by other service categories to those users who can adjust their

transmission rate. The resulting reward to the source for cooperation is very low loss. ABR uses a Minimum Cell Rate to set the lowest acceptable bandwidth value for each end to end virtual connection. A virtual connection is established for each source to destination transmission, utilizing the same physical network path. In the ATM network, the input rate of each source is controlled with Resource Management cells, which monitor the available bandwidth for each connection and switch congestion levels. With ABR a network traffic contract is created and enforced so that connections do not affect the other connection services. ABR is not good for real-time applications, such as telephony, videoconferencing and video distribution, because it does not provide bounded delay variation. Suitable applications include LAN interconnections, WAN transport, high performance file transfers, database archival, non-time sensitive traffic, web browsing and to a certain degree it can be used for compressed audio and interactive multimedia.[21]

We will study the behavior of TCP over UBR and ABR, utilizing EPD. We would like to understand the issues under which UBR+EPD may actually be a less complex, and in terms of industry goals, less expensive mechanism than ABR+EPD for attaining high performance, efficiency and fairness. We will try to determine if the cost of implementing ABR+EPD and having control over the buffering scheme of the TCP sources is justified in terms of bandwidth efficiency, packet loss and fair bandwidth allocation. Implementing lower cost UBR+EPD mechanism subjects the source to the buffering schemes of the ATM switch. We look at the relationship between various TCP parameters and the ATM switch buffer sizes and try to analyze if the Fast Retransmit and Recovery mechanism has a significant impact on performance. The Fast Retransmit allows a source to retransmit a dropped packet sooner than the regular timeout mechanism. Fast Recovery uses the acknowledgement packets that are still in the TCP pipe to clock the sending of packets, rather than drop the congestion window all the way back to one packet and run slow start.

The rest of the document is organized into three sections. Section 2 provides a background into ATM concepts and ideas, as well as TCP congestion control mechanisms. Section 3 discusses the simulation environment and issues of concern

and a detailed description of the experiments carried out. The findings and analysis of the experiments are also given in this section. Section 4 states a conclusion and talks about some of the further research and experiments that would prove useful, as well as a look at current and future ATM technology. A list of references follows.

This paper is intended to present an account of the study of the fundamental behavior of ATM Quality of Service (QoS) over basic TCP connections. It is not intended to present the latest cell drop policy, routing or switch algorithms for performance enhancement, but rather a summary of the authors personal research into the understanding of basic traffic and congestion control mechanisms. These findings are fundamental to any further analysis or development of ATM QoS and therefore are relevant in the discussion of current uses and service guarantees versus other technologies.

2 Background

2.1 ATM Primer

Asynchronous Transfer Mode (ATM) has become a tremendously important technology in recent years for a variety of reasons, but the emergence of ATM at the right time at the right place is perhaps the most important factor for its acceptance. ATM is a high speed switching technology that appeared on the scene just when shared media like Ethernet and FDDI were starting to look a bit too slow for many users of computer networks,

ATM refers to a high-bandwidth, low-delay switching and multiplexing technology that is available for both public and private networks. ATM principles and ATM based platforms form the foundations for the delivery of a variety of high speed digital communication services aimed at corporate users of high-speed data, LAN interconnection, imaging, and multimedia applications. Residential applications, such as video distribution, video-telephony, Computer Telephone Integration (CTI) and other information-based services are also planned. ATM is the technology of choice for evolving broadband integrated services digital network, public networks, for most next-generation LANs and WANs. ATM supports transmission speeds of 155 Mbps and 622 Mbps, and will be able to support speeds as high as 10 Gbps in the future.

The many forms of ATM technology allow it to provide a complete set of functions all included in one package. It functions as an interface between the user and the network, boundary between different types of hardware, interconnecting between nodes and connections between networks. It is defined as a protocol designed to switch constant, variable, unspecified and available bit rate traffic over a common transmission medium. It can use existing protocols and networks more cost-effectively and efficiently than separate networks and it can work with legacy protocols. Within the private network specifications, ATM can support automatic reconfiguration, optimized routing, bandwidth allocation, hierarchical scalability and multiple qualities of services. As a technology, it comes in the form of a network interface card, router, cross-connect or intelligent switch. Hardware and associated software together provide an efficient backbone technology, offering the best future integrated platform for multiple services. Greater bandwidth granularity and flexibility in designing network topologies is provided due to scalable aspects such as interface speed, port density, switch size, network size and addressing.

An essential aspect of ATM is its ability to provide integrated access and transport of voice, data and video over a single infrastructure. An important element of service integration is the provision of a range of services using a limited number of connection types and multipurpose user-network interfaces. ATM supports both, non-switched permanent virtual connections (PVC) and switched virtual connections (SVC). In a PVC service, virtual connections between endpoints in a customer's network are established at service subscription time through a provisioning process. These connections or paths can be changed via a subsequent provisioning process or via a customer network management application. In SVCs, the virtual connections are established as needed, in real-time, through signaling capability. ATM supports services requiring both circuit-mode and packet-mode information transfer capabilities. ATM can be used to support both connection-oriented and connectionless services.

2.1.1 Cells

Cells refer to the packets switched in an ATM network. These packets are

fixed length at 53 bytes. 5 bytes of header followed by 48 bytes of payload. One of the main reasons for implementing a fixed size packet was to facilitate the implementation of hardware switches. When ATM was being created in the mid and late 80's, 10 Mbps Ethernet was the innovative technology in terms of speed. To go much faster, most people thought in terms of hardware. In addition, for the telecommunications industry, fixed-length packets turn out to be easy to use if you want to build fast, highly scalable switches. It is easier to build hardware to do simple jobs and the job of processing packets is simple if the packet size is predetermined. Also, with equal sized packets, switching elements can operate in parallel and complete the tasks at the same time. Enabling parallelism also greatly improves the scalability of switch design.

Another property of cells relates to the behavior of queues. Queues build up in a switch when traffic from several inputs may be heading for a single output. In general, once a packet is de-queued and transmission has begun, it is necessary to continue until the packet is transmitted. Preempting the transmission of a packet is not practical. The maximum length of time a queue output can be busy is equal to the time taken to transmit a maximum sized packet. Fixed length cells mean that a queue output is never tied up for more than one cell transmission time. Thus, if tight control over the latency experienced by cells when they pass through a queue is important, cells provide some advantages. Cells provide potentially finer control over behavior of queues.

Consider the setting of the fixed length cell size. If a cell is made too small, then the amount of header information that must be transported relative to the amount of data that fits in one cell gets larger. Therefore, the percentage of link bandwidth that is actually used to carry data goes down. Even more seriously, if a device that processes cells at some maximum number of cells per second is used then as the cells get shorter, the total data rate drops in direct proportion to cell size. If a cell is made too large, then there is a problem of wasted bandwidth caused by the need to pad transmitted data to fill a complete cell.

Efficient link utilization is not the only factor that influences cell size. Cell size has a particular effect on voice traffic. The standard digital encoding of voice is

done at 64kpbs (8-bit samples taken at 8khz). To maximize efficiency, a full cell's worth of voice samples should be collected before transmitting a cell. A sample rate of 8khz means one byte is sampled every 125 seconds. Therefore, the time it takes to fill an n-byte cell with samples is n x 125 seconds. If cells are 1000 bytes long, it would take 125 ms to collect a full cell of samples before transmission to the receiver could start. The latency introduced is quite noticeable to a human listener.

The fixing of the cell length at 48 bytes was a compromise that pleased almost no one, except that it is actually an average of the 64-byte cell size proposal pushed by US telephone companies and the 32-byte cell size advocated by European companies.

2.1.2 Cell format

The format of a cell depends on the source-destination path taken by a cell. The User Network Interface (UNI) format is used when transmitting cells between a host and a switch. The Network-Network Interface (NNI) format is used when transmitting cells between switches. The only difference is that the NNI format replaces the GFC field with four extra bits of VPI. Figure 1 shows the cell format.

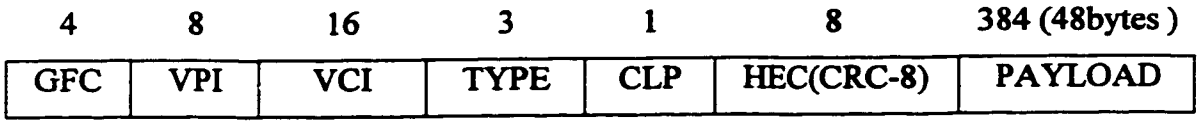


Figure 1: ATM Cell format

The GFC Generic Flow Control bits provides a means to arbitrate access to the link if the local site uses some shared medium to connect to ATM. The Virtual Circuit Identifier and Virtual Path Identifier can be considered as a single 24 bit identifier that is used to identify a virtual connection. The Type field has eight possible values. Setting the first bit of the Type field activates Management related functions. Clearing the first bit indicates that the cell contains data. In this case, the second bit is the Explicit Forward Congestion Indication (EFCI) bit and the third is the User Signaling bit. The former can be set by a switch to tell an end node it is congested. Cell Loss Priority (CLP) bit is set by a user or network element to indicate

that cells should be dropped, preferentially in case of an overload. For example, a video coding application could set this bit for cells that if dropped would not dramatically degrade the quality of the video. A network element might set this bit for cells that have been transmitted by a user more than the amount that was negotiated. The Header Error Check (HEC) uses a Cyclic Redundancy Code (CRC) polynomial to provide error detection and single bit error correction capability on the cell header only. Protecting the cell header is particularly important because an error in VCI will cause the cell to be misdelivered.

2.1.3 Segmentation and Reassembly

Packets handed down to the ATM layer from higher level protocols are often larger than 48 bytes. ATM being a data link layer protocol must somehow fragment the message into low level packets on the source, transmit the individual low level packets over the network, and reassemble the fragments back together at the destination. In ATM, this technique is called Segmentation and Reassembly.

Segmentation is not unique to ATM, but it is much more of a problem than in a network with a maximum packet size. To address the issue, a protocol layer sits between ATM and the variable length packet protocols that use ATM, such as IP. This layer is called the ATM Adaptation Layer (AAL). The AAL header simply contains the information needed by the destination to reassemble the individual cells back into the original message.

2.1.4 ATM Adaptation Layers

ATM was designed to support such services as video, voice and data that would have different AAL needs. Four adaptation layers were originally defined. Layers 1 and 2 were designed to support application like voice that required guaranteed bit rates, while layers 3 and 4 were intended to provide support for packet data running over ATM. The idea was that AAL 3 would be used by connection oriented services such as X.25 and AAL4 would be used by connectionless services like IP. Eventually, the reasons for having different AALs for these two types of service were found insufficient, and the AALs merged into one known as AAL3/4. Meanwhile, some people perceived shortcomings in AAL3/4 and proposed AAL5.

The main function of AAL 3/4 is to provide enough information to allow variable length packets to be transported across the ATM network as a series of fixed length cells, by supporting the segmentation and reassembly process. Protocol Data Units (PDU) is the name given to packets operating at this layer of the network hierarchy. The task of segmenting and reassembly involves two different packets formats. The first is the Convergence Sublayer Protocol Data Unit (CS-PDU), which defines a way of encapsulating variable PDUs prior to segmenting them into cells. The PDU passed down to the AAL layer is encapsulated by adding a header and trailer. The resultant CS-PDU is segmented into ATM cells.

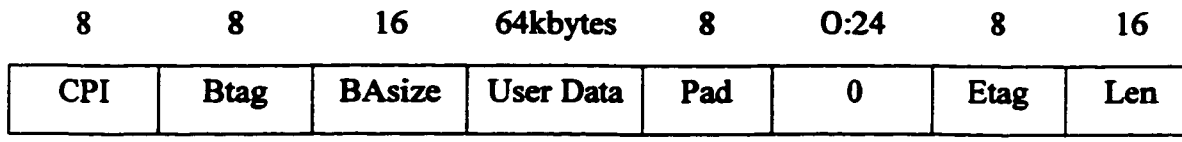


Figure 2: ATM AAL 3/4 packet format

Figure 2 shows the CS-PDU format. It begins with an 8-bit Common Part Indicator (CPI) indicating which version of the CS-PDU format is in use. Only the value zero is currently defined. The next eight bits contain the Beginning Tag (Btag) and is used to match the End Tag (Etag) for a given PDU. This protects against the situation where the loss of the last cell of one PDU and the first cell of another cause two PDUs to inadvertently join into a single PDU and be passed up to the next layer in the protocol stack. The Buffer Allocation Size (BAsize) field indicates to the reassembly process how much buffer space to allocate for reassembly. The Etag, real length of the PDU and a padding byte of zeros, make up the CS-PDU trailer.

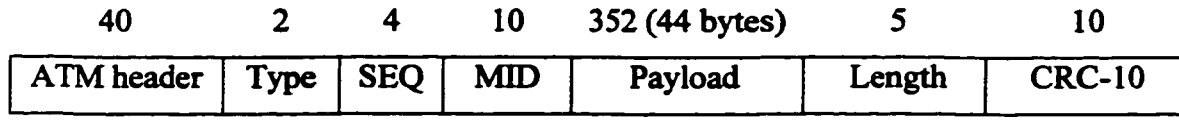


Figure 3: ATM cell format for AAL 3/4

The second part of AAL 3/4 is the header and trailer that is carried in each cell. The CS-PDU is actually segmented into 44-byte chunks. An AAL 3/4 header and trailer is attached to each one, bringing it up to 48 bytes, which is then carried as

the payload of an ATM cell. The type field indicates if this is the first cell of a CS-PDU, the last cell of the CS-PDU, a cell in the middle or a single PDU. The Sequence Number (SEQ) is intended to detect cell loss or misordering. Multiplexing Identifier (MID) is used to multiplex several PDUs onto a single connection. The Length field contains the number of bytes of PDU that are contained in the cell. The 10 bit CRC is used to detect errors anywhere in the 48-byte cell payload.

The AAL 3/4 uses a lot of overhead to perform the conceptually simple function of segmentation and reassembly. This was observed by many people in the field that led to the standardization of AAL5

AAL5 replaces the 2-bit TYPE field of AAL 3/4 with one bit of framing information in the ATM cell header. By setting that one bit, we can identify the last cell of a PDU. The next cell is assumed the first cell of the next PDU and subsequent cells are assumed Continuation of Message (COM) cells until another cell is received with the user signaling bit set. All pieces of AAL3/4 that provide protection against lost, corrupt or misordered cells, including the loss of an End of Message (EOM) cell, are provided by the AAL5 CS-PDU packet format, shown in figure 4.

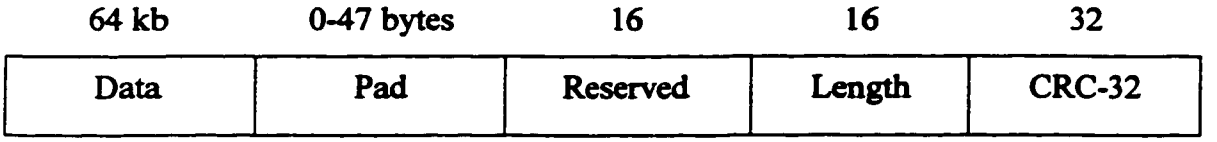


Figure 4: ATM Adaptation Layer 5 packet format

The AAL5 CS-PDU consists simply of the data portion (the PDU handed down by the higher layer protocol) and an 8-byte trailer. To make sure that the trailer always falls at the tail end of an ATM cell, there may be up to 47 bytes of padding between the data and trailer. The first two bytes of the trailer are currently reserved and must be zero. The length field is the number of bytes carried in the PDU, not including the trailer or any padding before the trailer. Finally, there is a 32-bit CRC.

2.1.5 Service Categories

To provide a guaranteed Quality of Service (QoS), a traffic contract is established during connection setup. This contains a connection traffic descriptor and a conformance definition. However, it is not necessary for every ATM virtual connection to have a specified QoS. If only specified QoS connections are supported by ATM, then a large percentage of the network resources will be wasted, This can happen when one or more connections are not utilizing the full capacity of their QoS contracts. Unspecified QoS contracts can be supported by an ATM network on a "best effort" basis. Such best-effort services are sufficient for supporting most of the existing data applications. In general, a traffic contract specifies one of the following classes of traffic. CBR, rt-VBR, nrt-VBR, UBR, ABR.

These service categories relate traffic characteristics and QoS requirements to network behavior and are distinguished as being either real-time or non-real-time. CBR and rt-VBR deal with real-time traffic and are distinguished by whether the traffic descriptor contains only the Peak Cell Rate (PCR) or both the PCR and Sustained Cell Rate (SCR) parameters. nrt-VBR, UBR and ABR deal with non-real-time traffic.

The Constant Bit Rate (CBR) service category is used by connections that request a static amount of bandwidth that is continuously available during the lifetime of the connection. This amount is characterized by the PCR value. The basic commitment made by the network to a user who reserves resources via CBR is that once the connection is established, the negotiated ATM layer QoS is assured to all cells when they are conforming to the relevant test. Conformance for a CBR connection is characterized by PCR and the corresponding Cell Delay Variation Tolerance (CDVT) for the CLP=0+1 traffic flow. The source can transmit at the PCR at any time and for any duration and the QoS will be maintained. CBR service is intended to support real-time applications requiring tightly constrained delay variation, for example video, voice and circuit switching emulation.

The real-time Variable Bit Rate (rt-VBR) service category is intended for those applications requiring tightly constrained delay and delay variation. These types of connections are characterized in terms of PCR, SCR and Maximum Burst Size

(MBS). Sources are expected to transmit at a rate which varies with time; i.e. bursty traffic. Cells that are delayed beyond the value specified by Maximum Cell Transfer Delay (maxCTD) are assumed to be of insignificant value to the application. rt-VBR may support statistical multiplexing of real-time sources.

The non real-time Variable Bit Rate (nrt-VBR) service category is intended for non real-time applications which have bursty traffic characteristics and which are characterized in terms of PCR, SCR and MBS. For those cells that are transferred within the traffic contract, the application expects a low cell loss ratio. nrt-VBR may support statistical multiplexing of connections. Delay bounds are not associated with this service category. A typical application would be multimedia email, while an application for rt-VBR would be interactive video.

Applications not requiring tightly constrained delay and delay variation, such as traditional computer communications, i.e. FTP, TELNET and email are supported by Unspecified Bit Rate (UBR). UBR does not specify traffic related service guarantees. Numerical commitments are not made with respect to Cell Loss Ratio (CLR) experienced by a UBR connection or as to the CTD experienced by cells on the connection. A network may or may not apply PCR to the Call Admission Control (CAC) or Usage Parameter Control (UPC) functions. This is unlike nrt-VBR where a connection may be rejected and at least a mean delay must be specified. Congestion control may be performed at a higher layer on an end-to-end basis. The Best Effort Indicator in the ATM User Cell Rate Information Element is used to indicate the UBR service.

The Available Bit Rate (ABR) service category employs transfer characteristics provided by the network that may change after connection establishment. A flow control mechanism supporting several types of feedback to control the source rate in response to changing ATM layer transfer characteristics is used. This feedback is conveyed to the source through specific control cells called Resource Management (RM) cells. It is expected that an end-system that adapts its traffic in accordance with the feedback will experience a low cell-loss ratio and obtain fair share of the available bandwidth according to the network specific allocation policy. The ABR service does not require bounded delay or the delay

variation experienced by a given connection. It is also not intended to support real-time applications. On establishment of an ABR connection, the end-system shall specify to the network both a maximum required bandwidth and a minimum usable bandwidth. These shall be distinguished as PCR and Minimum Cell Rate (MCR). Depending on the congestion state of the network, the source is required to control its rate, whereas for UBR, cells are lost but the sources are not expected to reduce their cell rate.

2.1.6 Nature of Service Guarantees and Mechanism

Since in this paper we will only be talking about the UBR and ABR services, we will discuss the service guarantees and mechanisms for these two traffic classes only.

The UBR service offers no traffic related service commitments. Numerical commitments are not made as to the cell loss ratio experienced by a connection or as to the cell transfer delay experienced by cells on the connection. Fairness among connections can not be assumed, although a local policy in some network elements may have this effect.

The ABR service provides a low cell-loss ratio for those connections whose end-stations comply with a specific reference behavior. Numerical commitments are not made about cell transfer delay. If the endpoints fail to observe the reference behavior then the cell loss ration is not guaranteed. The MCR and local policy in network elements is assumed to modify fairness among connections. A sufficient degree of isolation between connections is necessary so that connections that fail to observe the reference behavior do not cause quality degradation on connections that do observe the reference behavior.

The UBR service is inherently open-loop, meaning that no end-to-end feedback is employed. UBR is not subject to a specific contract but may be subject to a local policy in individual switches and end-systems.

The ABR service uses a closed-loop approach. The source performs dynamic traffic shaping based on feedback received from the network. This behavior may be enforced by the network using UPC. A non-zero MCR assumes that resources are

reserved in network nodes to ensure that the feedback never causes the available cell rate to fall below the MCR. Therefore, CAC is provided in the network. Local policy in network elements contributes to fairness and isolation, with the observance of achieving low CLR for those connections that obey the source behavior.

2.2 Aspects of TCP Congestion Control

2.2.1 Introduction

The idea of TCP congestion control is for each source to determine how much capacity is available in the network, in order to know how many packets can be safely transmitted. This is useful in avoiding congestion collapse. The sender is prevented from excessively retransmitting unacknowledged packets up to a point where the network is continually flooded and the source effectively experiences very little or no throughput. TCP employs mechanisms that act as indicators and mediators of congestion. Once a given source's packets are in transit, it uses the arrival of an ACK as a signal that one of its packets has left the network and that it is now time to sent a new packet through the network without adding to the level of congestion. By using ACKs to pace the transmission of packets, TCP is termed as self-clocking. The Sliding Window protocol of TCP guarantees the reliable delivery of data, ensures that data is delivered in order and enforces flow control between the sender and the receiver. The receiver advertises a window size to the sender, which is limited to the amount of unacknowledged data at any given time. The receiver selects a suitable value based on the amount of memory allocated to the connection for buffering data. The idea is to keep the sender from overrunning the receiver's buffer.

The three algorithms used by TCP to deal with congestion are Additive Increase/ Multiplicative Decrease (also known as congestion avoidance), Slow Start and Fast Transmit and Fast Recovery. These algorithms working with each other to provide the congestion control mechanism of TCP.

2.2.2 Additive Increase / Multiplicative Decrease

A state variable, congestion window (*cwnd*), is maintained for each connection. This variable is used by the source to limit how much data it is allowed to

have in transit at a given time. The congestion window is like the opposite of the flow control advertised window. TCP will have no more than the minimum of the congestion window and the advertised window of unacknowledged data. This gives the TCP maximum window. The effective window, determined by *maximum window – (last byte sent – last byte acked)*, indicates to the source that it is allowed to send no faster than the slowest component can accommodate. The congestion window value is set based on the level of congestion the TCP source perceives to exist in the network. This involves decreasing the congestion window when the level of congestion goes up and increasing the congestion window when the level of congestion goes down. This technique is called additive increase/multiplicative decrease.

To determine when to raise or lower the congestion window is based on the observation that the main reason packets are not delivered and a timeout results, is that a packet was dropped due to congestion. TCP interprets timeouts as a sign of congestion and reduces the rate at which it is transmitting. Specifically, each time a timeout occurs, the source sets congestion window to half its value. This halving corresponds to the multiplicative decrease part. This mechanism will not decrease the congestion value below a value equal to the maximum segment size (MSS). Therefore, a TCP source will always be sending at least one packet onto the network. In order to take advantage of newly available capacity in the network the additive increase mechanism increases the congestion window. Every time the source successfully sends a congestion window worth of packets, it adds the equivalent of one packet to the congestion window. However, in practice TCP actually increments congestion window by a little for each ACK that arrives according to the following:

congestion window += (MSS x MSS)/congestion window.

2.2.3 Slow Start

The additive increase mechanism described above takes too much time to ramp up a connection when it is starting from the beginning. TCP therefore provides a second mechanism called slow start, which increases the congestion window exponentially rather than linearly.

The source starts by setting congestion window to one packet. When the ACK for this packet arrives, TCP adds one packet to congestion window and then sends two packets. When the two corresponding ACKs are received, the congestion window is increased by two and hence sends four packets. TCP effectively doubles the number of packets it has in transit every Round Trip Time (RTT).

The two situations in which slow start is employed are at the very beginning of a connection and when the connection goes idle for a timeout to occur. At the beginning of a connection, the source is not aware of how many packets it is able to have in transit at a given time. In this situation, slow start continues to double the congestion window each RTT until there is a loss, at which time a timeout causes multiplicative decrease to divide the congestion window by two. In the second case, the source is using slow start again, but this time it has more knowledge of the network traffic. It has the current value of the congestion window, which because of the timeout, has already been divided into half. Slow start is used to rapidly increase the sending rate up to this value and then additive increase is used beyond this point. A temporary variable called the Slow Start Threshold (*ssthresh*) is set to the value of the congestion window after multiplicative decrease. The congestion window is then reset to one packet and is incremented by one packet for every ACK that is received until it reaches the Slow Start Threshold, at which point it is incremented by one packet every RTT.

2.2.4 RENO with Fast Transmit and Fast Recovery

The simulator used for the experiments described in this paper used the TCP RENO implementation. This implementation has the Fast Retransmit and Fast Recovery algorithm that triggers the retransmission of a dropped packet sooner than the regular timeout period.

The Fast Retransmit mechanism does not replace regular timeouts, rather it enhances that facility. Each time a data packet arrives at the receiving side, the receiver responds with an acknowledgement even if this sequence number has already been acknowledged. Whenever a packet arrives out of order TCP resends the same acknowledgement it sent last time. This second transmission of the same

acknowledgement is called a duplicate ACK. When the sender sees a duplicate ACK, it knows that the other side must have received a packet out of order, which suggests that an earlier packet might have been lost. Since it is also possible that the earlier packet has only been delayed rather than lost, the sender waits until it sees some number of duplicate ACKs and then retransmits the missing packet. In practice, TCP waits until it has seen three duplicate ACKs before retransmitting the packet. At this point, the Slow Start threshold is set to one-half the current congestion window size. The missing segment is retransmitted and the current congestion window is then set to the Slow Start threshold plus 3 times the segment size, which correspond to the three duplicate ACKs. Each time another duplicate ACK arrives, the current congestion window is incremented by the segment size and another packet is transmitted, assuming the new congestion window value permits it. When the next ACK that acknowledges new data arrives, the current congestion window is set to the Slow Start threshold value. This should be the ACK of the retransmitted packet and should acknowledge all intermediate segments sent between the lost packet and the receipt of the first duplicate ACK.

When the Fast Retransmit mechanism signals congestion, rather than drop the congestion window all the way back to one packet and enter the slow start phase, it is possible to use ACKs that are still in the TCP pipe to clock the sending of packets. This mechanism is called Fast Recovery and effectively removes the slow start phase. Slow start occurs after Fast Retransmit detects a lost packet but before additive increase begins. In sense, it cuts the congestion window in half and resumes additive increase.

2.2.5 Synchronization Effect

TCP connections are synchronized when their sources experience network congestion and all are chosen for packet loss. These connections timeout at the same time then the network channels become idle and no packets are transmitted, resulting in a complete waste of bandwidth. The connections that were not dropped send their next window and keep filling the buffer. All other connections timeout and retransmit at the same time. This results in their segments being dropped again and the

synchronization effect is seen. The sources that escape the synchronization get the most of the bandwidth.

3 Simulation Environment

3.1 Setup and Issues

This section describes the environment in which the experiments were performed. We will discuss the simulator used and its capabilities. Also detailed, are the various issues explored in the setup. These issues include TCP parameters like packet processing rate, maximum congestion window size, Round Trip Time, Slow Start Threshold, maximum packet size and retransmit strategies. The various traffic models available will be listed. Other network component related issues such as link delays and bandwidth, switch buffer sizes, cell processing rate and drop policies are discussed. We will touch on the expected results from LAN and WAN simulations in terms of total throughput and packet loss and present a description of how the simulations were done in the final subsections.

3.1.1 The Simulator

The C language based simulator used for the experiments described in this paper was developed at AT&T Bell Labs to be used as an internal generic tool for simulating various ATM based environments. Many different modules were placed as add-ons in a very ad-hoc manner to suit the requirements of the particular experiments performed over the years. Since its creation in 1992, the simulator has undergone several transformations to become the extremely complex and unintuitive tool that it is today. The coding is quite difficult to follow. There are no comments describing the flow and purpose of each module, as well as there is no documentation describing the use of the simulator.

Approximately four months is required to fully understand the capabilities of the software, however once this hurdle is overcome, a very powerful and flexible tool is revealed. Any type of ATM network environment can be simulated, because it is possible to customize exactly what is required. It then becomes easier to enhance or add code to obtain the outcome desired, although this trait was far from evident when

we first started using this software. The simulator allows us to select and configure parameters for all of the issues that will be mentioned later in this section. All simulations were performed on SUN© Sparc 2 stations running SOLARIS© v2.5.

This simulator is event driven, meaning that an event queue is maintained and contains events sorted by time. Every event has a start time and length. These simulation times are maintained by the event manager in units of ticks. One clock tick is equivalent to 10^{-7} seconds. To fire an event, the first event in the queue is removed, the global time is set to the time of the event and an action routine is called which is referred to by a pointer in the event structure. When the simulation starts, each component in the network configuration is sent a reset command followed by a start command. The simulator then enters a loop. The loop processes any number of events, updates the display (if it is active), then fires one event at a time from the head of the queue, going back to the loop to fire the next one. TCP sources, hosts, links and switches are some of the simulator components.

3.1.2 TCP Parameters

Values of TCP parameters like maximum congestion window, packet rate and maximum packet size affect the results observed from simulations.

If the maximum congestion window size is chosen to be too small, then the receiver buffers become the bottleneck. The congestion window is set to the delay-bandwidth product to ensure that a source can fully utilize the network and still have a small transient period. The delay is taken to be the Round Trip Time of a connection and the bandwidth is that of the bottleneck link. This bottleneck link will become more evident when the experimental network topology is illustrated in a later section.

Round Trip Time values effect the congestion window, the effective throughput, the definition of network topology, whether it's a LAN or WAN, as well as TCP timeout, retransmit and recovery times. Setting the link delays effects the RTT value and hence effects all other parameters. For this paper, most experiments use link delays of 2 milliseconds, giving an RTT value of 20 milliseconds, representing a WAN scenario.

The Slow Start Threshold maximum value has implications on how quickly a connection will reach steady state when it initially begins packet transport and also on how quickly a TCP source can recover from congestion following packet loss caused by a timeout.

The maximum size affects the calculation of the congestion window. The Maximum Segment Size is set at the TCP standard value of 512 bytes.

The simulator allows the use of several different retransmit strategies. TCP traditionally utilizes the SLOW START strategy and hence this is what we use in our experiments. This retransmit strategy should not be confused with the Fast Retransmit mechanism.

The TCP parameters values that we have chosen are interdependent on the delay-bandwidth product. We attempt to ensure that no other component except the bottleneck switch account for the results obtained. In essence, we do not want to create artificial bottlenecks by setting a slow TCP source or slow host-to-switch links. The slow-start threshold is set to 341 packets. This value makes the initial transient period small.

3.1.3 Traffic Models

Several different traffic patterns are available to model various applications. Some of these patterns include Constant Flow, Poisson Distribution, Uniform Distribution, Two State, Trace, Train, Infinite and Burst.

The simulator allows us to choose the traffic model, protocol and the length of the pattern. Since all our simulations will be performed with either the UBR or ABR service class, we will be employing the Infinite traffic pattern for TCP. As previously described in the ATM Service Categories section, UBR and ABR are designed to handle non-bounded delay and delay variable traffic. Therefore, we use a traffic pattern that will continue to generate packets until the end of the simulation.

3.1.4 Links

As previously mentioned, link delays are chosen to set the overall connection RTT value. The RTT value directly affects the congestion window which in turn

affects the switch buffer size which in turn affects the throughput efficiency of the system simulated.

Link bandwidths should be chosen just as carefully to reflect current existing real network situations. As will be described in the coming sections, each experiment setup will be using some number of TCP sources and for each case a bottleneck link is desired. Link bandwidths are chosen so that the bottleneck link is the main component tested. All measurements will be based on how this link and the two switches at each end handle traffic congestion.

For each experiment, there are three types of links. One type of link connects TCP hosts to edge routers, another connects routers to ATM switches and the other interconnects switches. The experiments will be the same in one aspect, in that we will be testing one bottleneck link between two switches. The number of sending and receiving hosts will vary. The router-to-switch bandwidths will always to be set to a value more than twice that of the bottleneck switch.

3.1.5 Switches

ATM switches have various ways of receiving, queuing and sending out cells along the virtual channels that may exist. Some of these schemes are Round Robin (RR), Weighted Round Robin (WRR), and First In First Out (FIFO).

RR selects a cell from each virtual channel in a predictable equal manner, processes it and queues it in the respective outgoing channel. WRR assigns a weight to each virtual channel. Cells are taken from a channel that has a larger weight more often than from channels with lesser weights. Factors like bandwidth allocation, channel queue size and cell loss influence the dynamically determined weights. FIFO simply takes the cells as they arrive at the switch regardless of which channel they came from. There are two ways of selecting and processing the cells as they arrive at the switch. Per VC-queuing is used with the RR and WRR schemes where each channel has a buffer into which cells are queued. The switch then selects the cells from these buffers. A single buffer is maintained in the FIFO scheme where cells are queued and processed in a first-come first-serve basis.

For this paper we are performing the experiments with a FIFO scheme and have set the rate at which the switch selects and processes the cells to be sent out, equal to the bottleneck bandwidth. This conceptually allows the switch to load the link with cells at the same rate at which the link can transport them.

The simulator switch component has a cell processing rate parameter. This sets the rate at which cells are retrieved from the incoming channels and then sent into the outgoing channel queue. We have set this value to the bottleneck link bandwidth.

3.1.6 Drop Policies

The simplest form of a cell drop policy would be to drop incoming cells when the switch buffer becomes full. This is known as Tail Drop and results in excessive waste in bandwidth. If cells from multiple TCP packets were dropped, then all TCP sources would have to retransmit these partially lost packets. If all cells from a particular packet have been accepted by the switch buffer except the very last one, then because of this one cell loss, the entire packet must be resent by the TCP source. There is no guarantee that the same situation will not occur again. An intelligent drop policy like Early Packet Discard (EPD) can improve throughput of TCP connections.

EPD introduces a threshold value set at the switch. When the buffer occupancy level exceeds this threshold, any subsequent cells from newly arriving packets are discarded. In other words, a threshold value is selected such that cells arriving from a packet before the threshold value was exceeded are continued to be accepted even after the limit is reached. We set this threshold value based on the number of TCP connections (noc) and the cell per packet ratio (cell/pkt):

$$threshold = noc * cell/pkt$$

EPD may increase the efficiency of bandwidth usage, however it makes no attempt at ensuring fairness among connections. EPD indiscriminately discards complete packets from all connections without taking into account their current rates or buffer utilization. The increase in fairness in LAN environments is because EPD

can sometimes break TCP synchronization and in such cases, a few connections are dropped during connection.

Two other drop policies are Selective Drop (SD) and Fair Buffer Allocation (FBA). SD keeps track of activity of each VC by counting the number of cells from each VC in the buffer. The deciding value, called the cutoff parameter, is used to determine how much each VC is overloading the buffer. The one with highest overloading factor is chosen for cell drop. SD with per-VC accounting improves the fairness of TCP over UBR+EPD because cells from overloading connections are dropped in preference to underloading ones. FBA uses a smooth form of the cutoff parameter used in SD. More detailed discussions on these two schemes can be found in [9,11].

3.1.7 Efficiency in LAN and WAN simulations

Efficiency of bandwidth usage is lower in LANs than it is in WAN simulations. The larger RTT values in WANs allow more cells to be cleared out before the next window is seen, as compared to LANs. Consequently, WANs have a higher throughput than LANs.

The default TCP maximum window size of 65535 bytes corresponds to more than 1500 cells at a switch for each source in a LAN. For 5 sources and a buffer size of 1000 cells the sum of the window size is almost 8 times the buffer size. For WANs, with 5 sources and a buffer size of 12000 cells, the sum of the window size is less than 6 times the buffer size. Higher efficiency values result when the window size of LAN simulations is set to less than the default value.

Increase in efficiency in both types of topologies are noticed when the switch buffer size is also increased. Larger buffers accept more cells before loss occurs and result in better performance. Of course, the buffer size is directly influenced by the window size.

3.1.8 Simulation Process

This section describes the procedure used to create the different simulation environments and generate the data from which results were analyzed and presented in this paper.

As previously mentioned the main tool used was an ATM software simulator. The tool was used to create different network environments and allowed us to define various network devices and their operating parameters. These parameters were discussed in the previous subsections of Section 3. The application was launched as a command line executable with various input parameters and flags, such as the name of a configuration file, the duration of the simulation and the output filenames.

All experiments were created by the simulator, based on the contents of the input configuration file specified at the command line at execution time. These configuration files define all objects and operating parameters. The entire network environment is specified in these files from which the simulator obtains its information. The contents define which network devices are to be used, what values they are to have, where they are to be placed in relation to each other and how they should interact. The simulator reads the input file and parses each line looking for key words. For example, when it finds an occurrence of the word HOST, it will execute that part of the code that defines a host computer. It will then expect the input file to provide the name of the host and the size of its packet buffer. The key word LINK will define a connection between devices. The properties of a link are name, bandwidth, delay and the network devices that the link will connect. The SWITCH key word will alert the simulator to the properties of an ATM switch device, whose parameters include name, cell processing rate, cell buffer size as well as input and output paths. The TCP keyword will define a TCP traffic source with such properties as source and destination hosts, packet size, packet processing rate, packet window size and transmission model. The last parameter specifies the traffic pattern to be used and was discussed in subsection 3.1.3. The final parameter we are able to specify is which type of data we wished to record and view. For all of our experiments, we specified that the TCP source congestion window and throughput values be recorded. The output file to where this data is dumped is specified as a parameter at the command line, during execution of the simulations

The values that were chosen for the properties defining all the network objects reflect real network scenarios. Component values such as Round Trip Time and buffer sizes of the switches and TCP host are chosen so that the results obtained can

be compared to real network environments. The previous subsections discussed issues that concerned the reasoning on how the network properties should be set. These decisions or designs were translated to actual values and applied to the input configuration files.

Our experiments deal with the ABR and UBR services classes, therefore we had to create two different instances of the simulator. One executable was compiled with the ABR code and another executable was compiled with the UBR code. A Makefile was used to choose which code would be compiled. The Early Packet Discard algorithm was invoked in the same manner, with a flag specified in the Makefile.

Once a simulation run was completed, we manually extracted the information from one of the output files and organized it into tables. This particular output file provided a summary of the statistics that we wished to gather. Information such as the number of packets sent, received, lost and retransmitted are given in a simple line by line form. The tables presented in each experiment's subsection contain information from these output files. The larger output files that contained the congestion window and throughput values were parsed and separated into smaller data files using PERL extraction scripts. These data files were then plotted and converted to GIF format using GNUPLOT and then later imported into this document.

The above procedures were followed for every instance of every experiment. The next section will describe the design model used in simulating the different network environments. It will also describe the experiments performed and the results obtained.

3.2 Experiments and Results

This section outlines the experiments performed with descriptions of the testing conditions and network topology. The data collected is presented in tabular form and full plots of the graphs obtained from each experiment are given at the end of each respective sub-section. Time slices of the plots are used to enhance the explanation of TCP behavior subjected to the experimental conditions at a particular instance of time.

Before outlining the experiments, we give a small description of the model used for testing. We use an IP/ATM/IP model as described in [12]. The ATM Net is comprised of two ATM switches interconnected by a single link. The interswitch link will be referred to as the bottleneck link and will effect the performance of our experiments. IP routers connect the TCP hosts to the ATM switches. Thus, we have the following diagram.

Figure 5: IP/ATM/IP model

A host-to-router delay (HR delay) is introduced to simulate the time taken for one packet to traverse the IP cloud originating from the TCP host. In order to reflect a WAN configuration, we have set this delay to 2ms and use it in all of our RTT values. A delay and bandwidth setting is also placed on the router-to-switch links. The bandwidths of the links are set to large enough values as to not artificially influence the outcome of the experiments.

We will be looking at two different scenarios with this model. In the first case we use infinite buffering on the routers and assume that every host will be transmitting over unique routers and links. The first three experiments will cover this case. The second case will implement finite buffering and use both separate and shared routers. The latter will be useful in determining how well the router is able to buffer multiple sources. Experiments 4 to 6 are based on the second case. The purpose of having two scenarios is to show the environment in which UBR and ABR are most useful. This will be exhibited in full detail in the subsequent sections.

These experiments were performed in order to understand the behavior of TCP over UBR and ABR. The results of the experiments will not reveal a design for optimal efficiency or fairness, but rather a thorough understanding of the situations may lead towards the development of an optimal design. The experiments are designed and performed in a logical order of increasing complexity and strictness.

This means that the first experiment will look at a simple configuration with one source and study the TCP behavior. Next, we will add another source to the configuration and add to the strain applied on the switch buffering and interswitch link. The following logical step is to see the behavior if sources have different Round Trip times. If we performed the above experiments with UBR, then we should also see what results are given when ABR is used on the same configuration as above, except for the single TCP source case. All these experiments are configured so that each source has a unique path to the ATM network; i.e. separate routing devices are used. In the last experiment, we look at the TCP behavior when sources must share the same router.

An in-depth explanation and analysis of the experiments is given in each following sub-sections. The basic TCP behavior over a single fixed bandwidth connection is investigated in sub-section 3.2.1. A situation where more than one TCP source must share the single bottleneck link is discussed in sub-section 3.2.2. Sub-section 3.2.3 analyzes the TCP performance for multiple connections with different RTTs. The next three sub-sections deal with ABR experiments performed over finite buffered routers. Sub-section 3.2.4 examines the case of two sources transmitting over separate routers. Sub-section 3.2.5 examines the case of two sources transmitting over separate routers, with each source having a different RTT. The sub-section 3.2.6 describes the situation of two sources sharing the same router.

Each experiment's results are illustrated as graphical representations. These graphs are given at the end of each respective experiment's subsection and represent the evolution of the congestion window and throughput over time. We are interested in the steady state phase of the experiments, therefore most graphs are plotted for the period starting from 10 seconds into the simulation and ending at 30 seconds into the simulation. All experiments were run for 40 seconds, but we are neglecting the first 10 seconds, representing the transient period leading to steady state.

3.2.1 Experiment 1: TCP Behavior

This experiment is used to measure and detail the behavior of TCP over a fixed bandwidth channel. The results will serve as a basis of comparison for TCP throughput values.

This experiment will only use the UBR service class. In this experiment one source will be transmitting to one receiver, hence there will be no competition for bandwidth and the single connection will be allocated the entire amount. Use of the ABR service class will not serve any purpose in this set of tests.

We will perform several simulation runs based on some changing parameter values. The switch buffer size will be varied and its effect on effective and peak throughput will be observed. TCP will be able to transmit at its peak window size for a duration directly proportional to the size of the switch buffer. Once the buffer approaches overflow, the TCP source will change to congestion control mode and drop its throughput rate. The rate at which the TCP source will recover from congestion is influenced by the presence or absence of the Fast Recovery algorithm. We use a RENO flag (named after the TCP RENO implementation) to activate and deactivate the Fast Recovery steps.

The slow-start threshold value is set to a value proportional to the connection Delay-Bandwidth product. This value is chosen in anticipation of reaching the steady state quicker. We expect to obtain 8-12 steady state cycles through a simulation time of 40 seconds. The goal is to obtain the effective TCP throughput in steady state and observe the evolution of the TCP congestion window.

Hence, the sets of simulations that are run with this simple configuration are as follows.

- UBR; RENO flag active; 9 different switch buffer sizes
- UBR; RENO flag inactive; 9 different switch buffer sizes

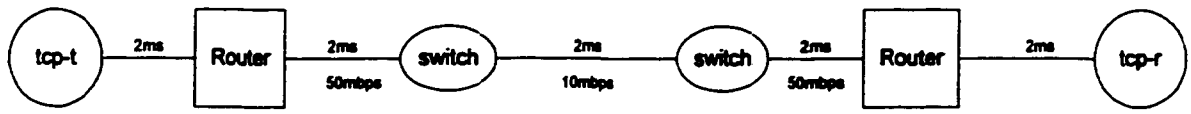


Figure 6: Experiment 1 – TCP behavior over a single connection

TCP Behavior with RENO off

X	buffer size (bytes)	throughput (pkts/sec)	(mbps)	loss (pkts)	Rxt (pkts)	Efficiency (%)
0.10	2500	749	3.07	1573	191	30.68
0.17	4250	805	3.30	1793	269	32.97
0.25	6250	876	3.59	1100	136	35.88
0.75	18750	876	3.59	1100	136	35.88
1.00	25000	1954	8.00	1221	295	80.04
1.50	37500	1997	8.18	1353	228	81.80
2.00	50000	2016	8.26	1540	267	82.58
2.50	62500	2035	8.34	1738	306	83.35
3.00	75000	2038	8.35	1936	345	83.48

Table 1: Experiment 1 – TCP behavior with RENO off

TCP Behavior with RENO on

X	Buffer size (bytes)	Throughput (pkts/sec)	(mbps)	loss (pkts)	rxt (pkts)	efficiency (%)
0.10	2500	1681	6.89	836	191	68.85
0.17	4250	1751	7.17	803	269	71.72
0.25	6250	1825	7.48	836	136	74.75
0.75	18750	1825	7.48	836	136	74.75
1.00	25000	2100	8.60	1243	295	86.02
1.50	37500	2100	8.60	1375	228	86.02
2.00	50000	2099	8.60	1540	267	85.98
2.50	62500	2099	8.60	1749	306	85.98
3.00	75000	2098	8.59	1936	345	85.93

Table 2: Experiment 1 – TCP behavior with RENO on

We ran one set of simulations with the TCP RENO implementation of the Fast Retransmit and Recovery and another set with out it. Using a bottleneck bandwidth of 20mbps and RTT of 20ms, we applied the following equation to obtain the switch buffer sizes, where **X** is the varying factor for determining what buffer size to use.

*Buffer = (X * RTT * Bandwidth)/ 8 bytes*

Tables 1 and 2 summarize the varying factor X, buffer sizes, throughput, loss, retransmitted packets and bandwidth usage efficiency. We use the efficiency calculations described in [10], which is described as follows:

Efficiency = (sum of TCP throughputs)/(maximum possible TCP throughput)

We varied the buffer sizes from 10% of the delay-bandwidth product up to 3 times the delay-bandwidth product.

Generally, the following deductions can be made from the data presented in Table 1. As the size of the switch buffer increases, the throughput also increases, but at the expense of greater packet loss and retransmitted packets. However, there is no increase in throughout from a buffer size of 25000 bytes to 75000 bytes. This range represents a factor of one to three times the delay bandwidth product, as shown by the factor X column. With X=1.0, a throughput of 80% is obtained, but with X=3.0 only an increase of 3% results, although 700 more packets are lost. This tends to show that the optimal throughput with minimal packet loss can be achieved when the switch buffer size is equivalent to the delay-bandwidth product, and any larger values yield a poorer **Cost-Benefit** (CB) ratio. This measures the cost of lost packets versus the efficiency of the use of the bandwidth. A summary of the ratios is provided below.

X	Packet Loss	% Efficiency	CB Ratio
1.0	1221	80.04	15.25
2.0	1540	82.58	18.65
3.0	1936	83.48	23.19

Table 3: CB Ratio

A lower CB ratio means better throughput at a lower cost. When X=1.0 a better CB ratio is obtained than when X=3.0.

When we ran the same experiment with the FAST Recovery implementation, Table 2, we obtained slightly better throughput. With the RENO flag ON, the source was able to recover from timeouts faster and send out more packets. This is evident in the range from X=0.1 to X=0.75, where a significant increase in throughput is seen. Comparing efficiency increases for X=1.0, 2.0, 3.0 shows a best CB ratio for X=1.0

X	RENO OFF	RENO ON	% increase	CB Ratio
1.0	80.04	86.02	5.98	14.45
2.0	82.58	85.98	3.40	17.91
3.0	83.48	85.93	2.48	22.53

Table 4: Efficiency comparisons

Graphs 1 to 10 show that the source is able to maintain maximum throughput for longer periods as the size of the switch buffer is increased. The sharp drops in the graphs show packet loss followed by timeout. Graphs 11 to 22, with RENO ON, show that the source is always able to maintain a high transmission rate.

If we look at a short time slice of the Congestion Vs Time traces for X=0.1, Graph 2, we can see that the Fast Retransmit mechanism is in effect. The time slice is taken from 10.5 to 12.5 seconds, in order to capture the display of unacknowledged segments and timeout. At 10.71 seconds, the congestion window (*cwnd*) is one segment, the slow start threshold (*ssthresh*) is two segments (1024 bytes), and the source begins in congestion avoidance phase. 0.95 seconds later, at 11.7 seconds, *cwnd* = 25049, *ssthresh* = 1024, a Fast Retransmit occurs and *ssthresh* is reduced to one-half of *cwnd*, whose own value is now set to one segment (512 bytes). The source is now in the slow-start phase. Another set of 3 duplicate ACKs at 11.8 seconds triggers another Fast Retransmit, again ssthresh is set to one-half of cwnd, and *cwnd* starts at 512 bytes. Multiple packet loss is experienced here and cannot be handled by the Fast Retransmit mechanism. This forces the source to timeout, because no ACKs were received. The connection begins in congestion avoidance at 12.4 seconds with *ssthresh* = 1024, *cwnd* = 512.

The difference between Graph 2 and Graph 12, for congestion VS time [X=0.1], is that the Fast Recovery mechanism eliminates the slow start phase after the duplicates ACKs are received. A shorter slice from 10.5 seconds to 11.5 seconds is taken this time. A Fast Retransmit and Recovery (FRR) is issued at 10.58 seconds, with *cwnd* = (*ssthresh* + 3 times segment size), after the FRR. The spike peaks at 37 Kbytes as compared to the previous non-FRR time slice, which only peaked at 25 Kbytes. *Cwnd* is incremented each time another duplicate ACK arrives, up until the

packet that acknowledges new data is received. At this point *cwnd* is set to *ssthresh* = 12288 and the source enters the congestion avoidance phase. Since there is still data flowing between the two ends, it would be useful not to reduce the flow abruptly by going into slow start. The Fast Recovery mechanism allows the source to continuously increase its congestion window size with the reception of each acknowledgement. The congestion window is allowed to grow beyond the theoretical capacity value of the TCP pipe, because the congestion window size does not necessarily reflect the actual number of packets on the pipe, at a particular time. The pipe capacity is calculated as the Delay-Bandwidth product plus the switch buffer size. The congestion window increases for every ACK received and in slow-start mode, the increase is exponential. In addition, when there is packet loss, the source will not stop its increase until three duplicate ACKs are received. Hence, the TCP pipe will actually have more space that is available at that time when the congestion window equals the pipe capacity value. Fast Recovery will make use of this space and keep the source from timing out. We have just compared the behavior of TCP when Fast Recovery was used and when it was not. We looked at a particular case where the switch buffer size was set at 10% of the Delay-Bandwidth product.

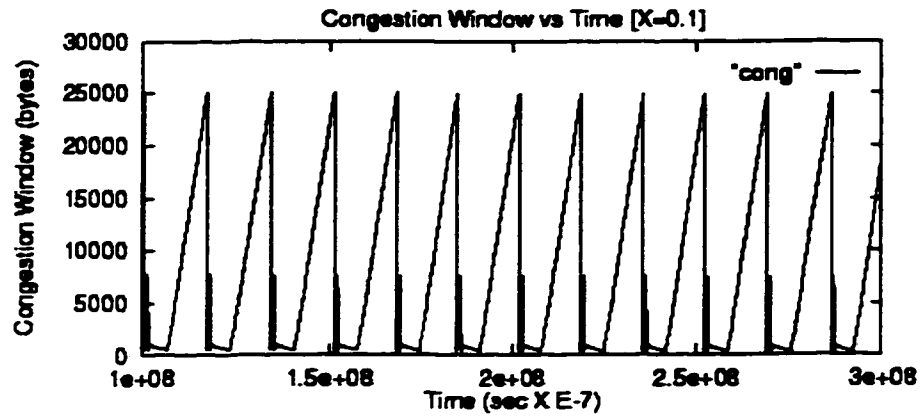
The set of Graph 3 to 5 show the congestion window, as the switch buffer is increased. In these cases, Fast Recovery is not activated. We see that as the buffer size is increased, the source is able to reach higher peak values as well as experience less packet loss. For X=1.0, the peak value is 45 Kbytes and 12 Fast Retransmits are required due to packet loss. For X=2.0, the peak value is 67 Kbytes and 6 Fast Retransmits are required. Finally for X=3.0, the peak value is 89 Kbytes and only 3 Fast Retransmits are required. To summarize Graph 1 to 5, we see that increased buffer size yields increased congestion window and fewer packet losses. All graphs exhibit the same behavior of increasing window size while in congestion avoidance mode. This continues until packet loss is experienced at which time the source switches to slow-start mode. At this point, *cwnd* = 512 bytes. The congestion window increases until *ssthresh* is reached, where *ssthresh* = (1/2)(previous *cwnd* value). The source then resumes in congestion avoidance mode. The Graph 6to 10 are the corresponding throughput plots, for the experiments performed without Fast

Recovery. The Graph 6 and 7 correspond to Graph 1 and 2, which show congestion window traces. Graph 7 is a time slice taken from Graph 6. It shows the throughput changes as the source changes from congestion avoidance to slow-start after a Fast Retransmit. The throughput patterns of Graph 8 to 10 match those of the congestion window Graph 3 to 5. It is seen that as the switch buffer increases so does the periods at which maximum throughput are maintained. The drops correspond to the Fast Retransmits on the congestion window graphs.

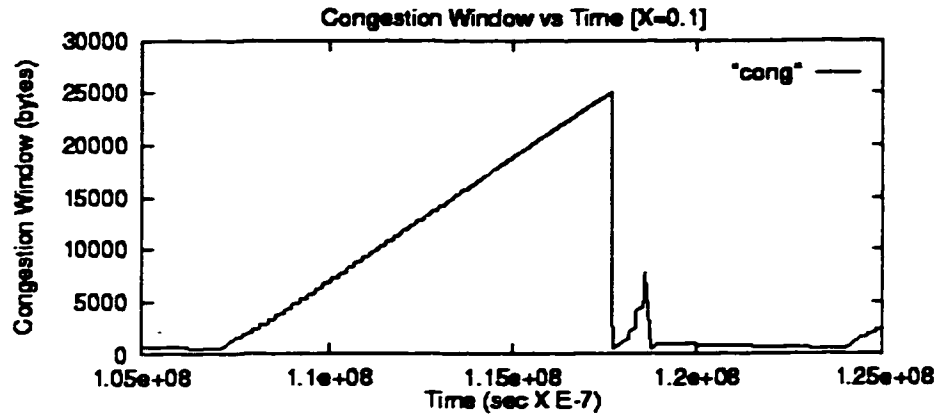
Graph 11 to 16 illustrate the congestion window for various buffer sizes when Fast Recovery is used. We have already closely looked at the case where the switch buffer is 10% of the Delay-Bandwidth product. This is represented by the X=0.1 plots. The remaining graphs for X=1.0, 2.0 and 3.0 show that increased buffer capacity gives greater peak window sizes. The patterns are in accordance with the set of experiments when Fast Recovery was not activated. Recapping, as the buffer size gets larger, fewer duplicate ACKs are received by the transmitting source and hence we see fewer Fast Retransmit activation. Another aspect of the Fast Recovery mechanism is that it does not start the congestion window at one segment (512 bytes) after a Fast Retransmit is issued. These plots (Graph 11 to 16) show that the window size never drops to one segment and hence a large congestion window is always maintained. This observation is replicated on the corresponding throughput Graph 17 to 22. The plot range (10 to 30 seconds) show that for all buffer sizes, the throughput never drops to small values, when Fast Recovery is activated. For X=0.1, the lowest value is 120 packets and for X=1.0 the lowest value is 213 packets. Graph 21 and 22 are plotted for the entire simulation range. Apart, from the transient period of the first 2 seconds, the throughput remains high at approximately 215 packets.

We have thoroughly examined the behavior of TCP over a simple configuration using UBR, with and without Fast Recovery. We saw how the switch buffer size had an effect on throughput and how Fast Recovery mechanism allowed the source to maintain a peak throughput for longer periods. This experiment provides a good foundation for understanding basic TCP performance over ATM networks. However, we would now like to examine the performance and bandwidth allocation

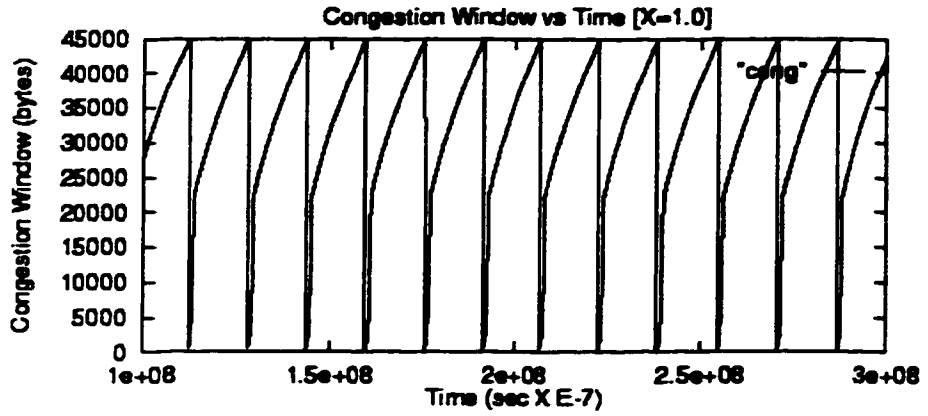
fairness when more than one source must traverse the ATM bottleneck link. The next experiment investigates this case.



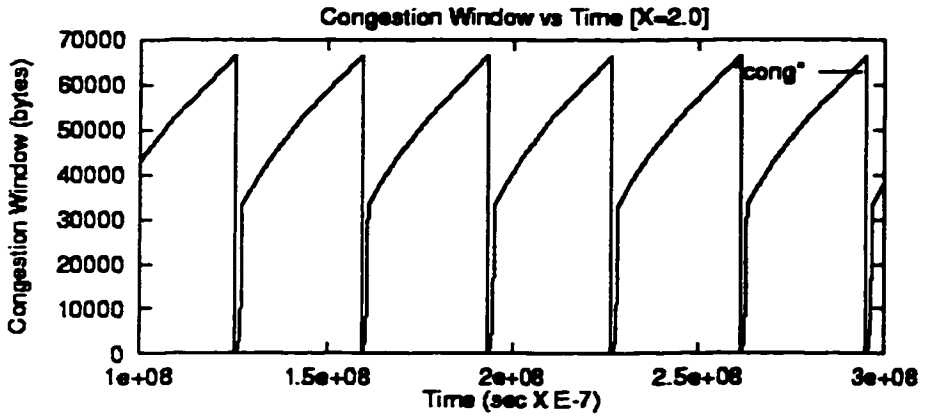
Graph 1: Exp1(RENO off) – Congestion vs Time [X=0.1]



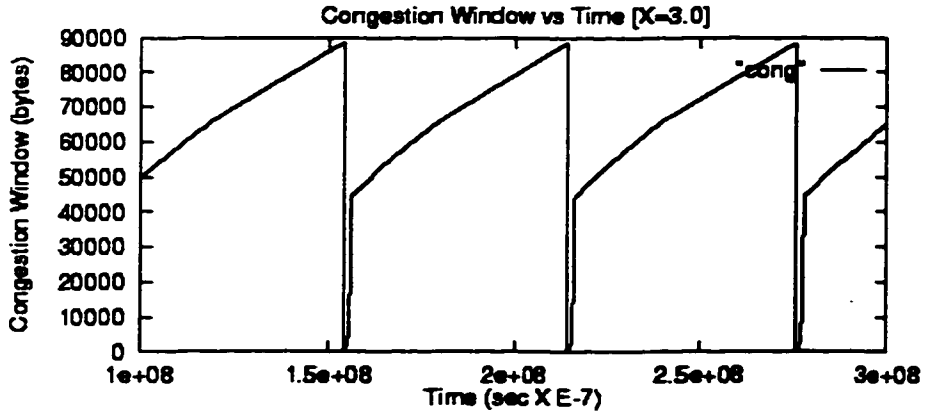
Graph 2: Exp1(RENO off) – Congestion vs Time [X=0.1]; time slice = 10.5 to 12.5 sec



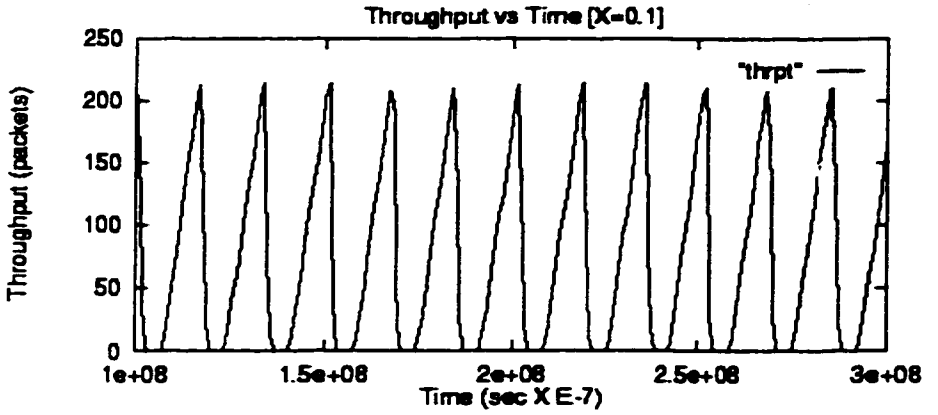
Graph 3: Exp1(RENO off) – Congestion vs Time [X=1.0]



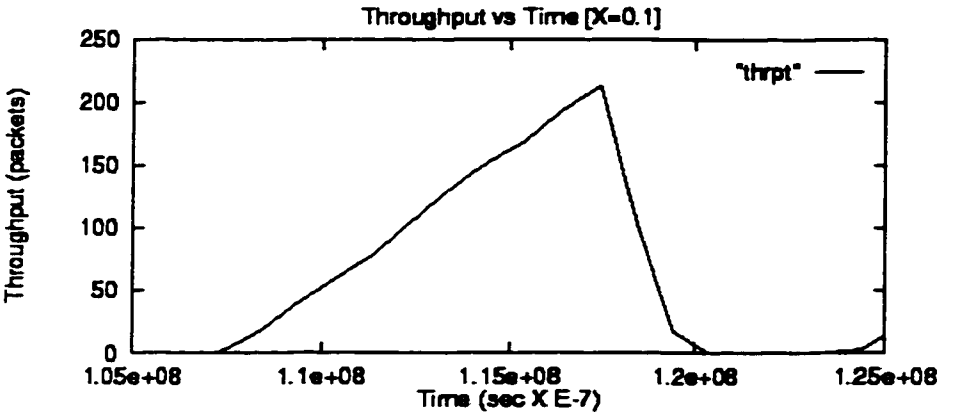
Graph 4: Exp1(RENO off) – Congestion vs Time [X=2.0]



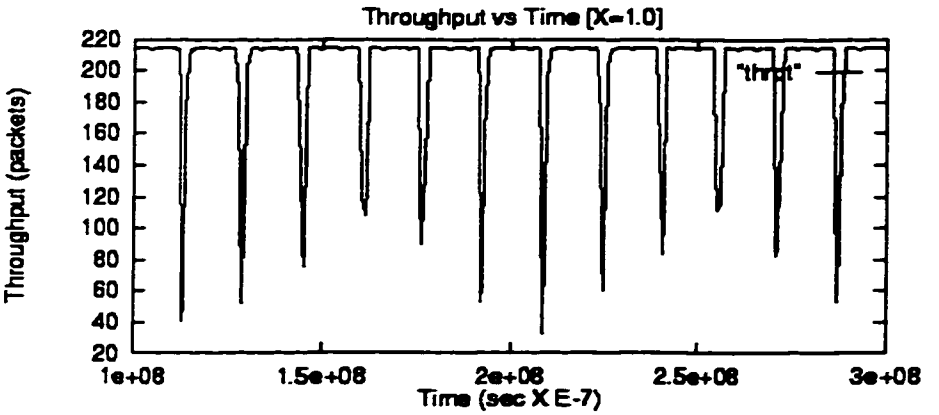
Graph 5: Exp1(RENO off) – Congestion vs Time [X=3.0]



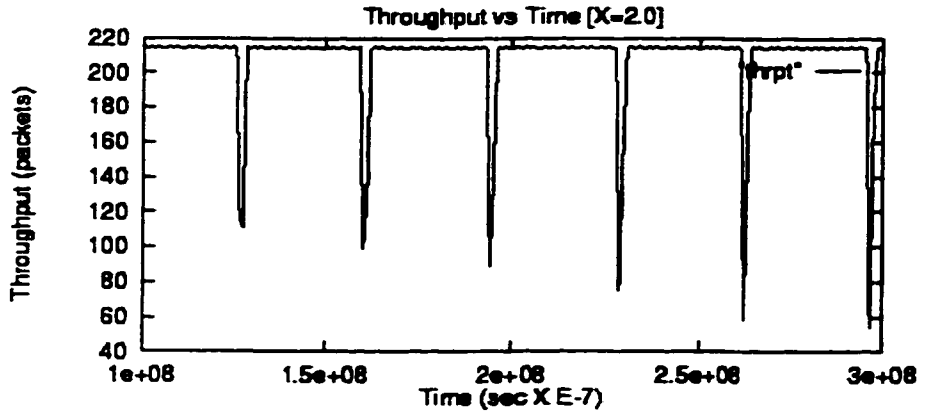
Graph 6: Exp1(RENO off) – Throughput vs Time [X=0.1]



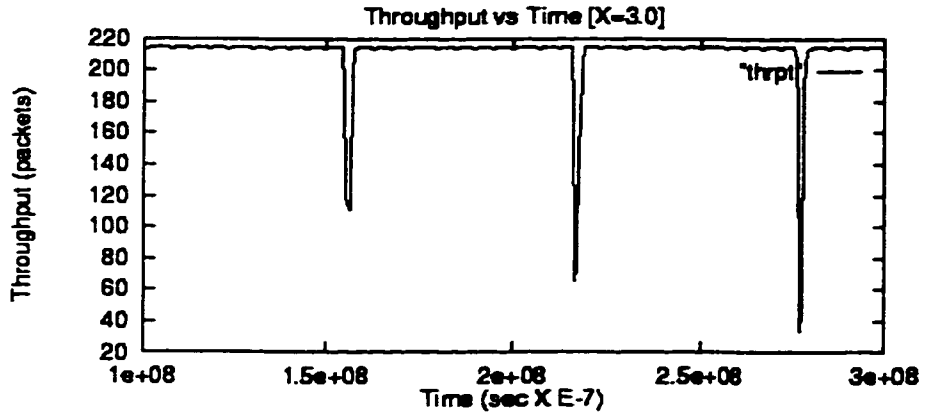
Graph 7: Exp1(RENO off) – Throughput vs Time [X=0.1]; time slice = 10.5 to 12.5 sec



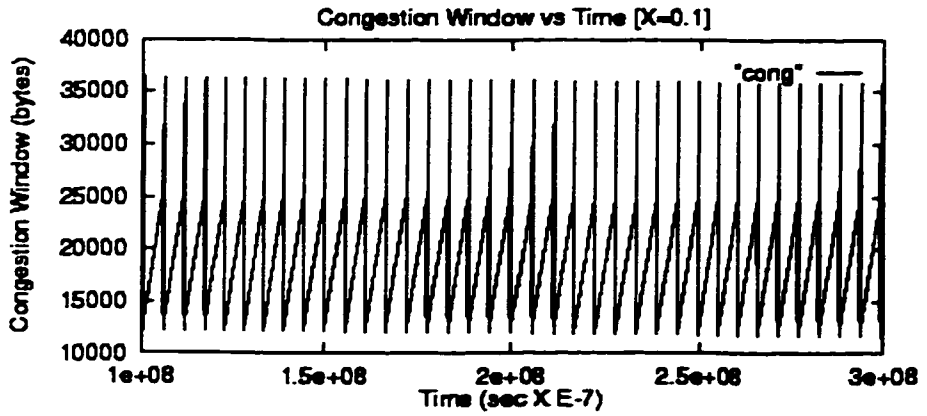
Graph 8: Exp1(RENO off) – Throughput vs Time [X=1.0]



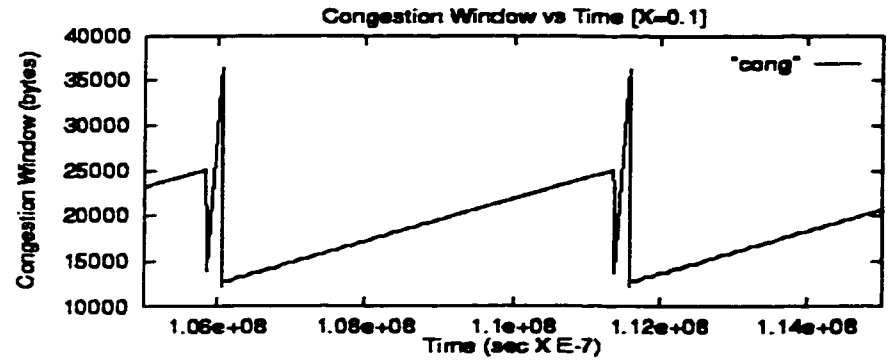
Graph 9: Exp1(RENO off) – Throughput vs Time [X=2.0]



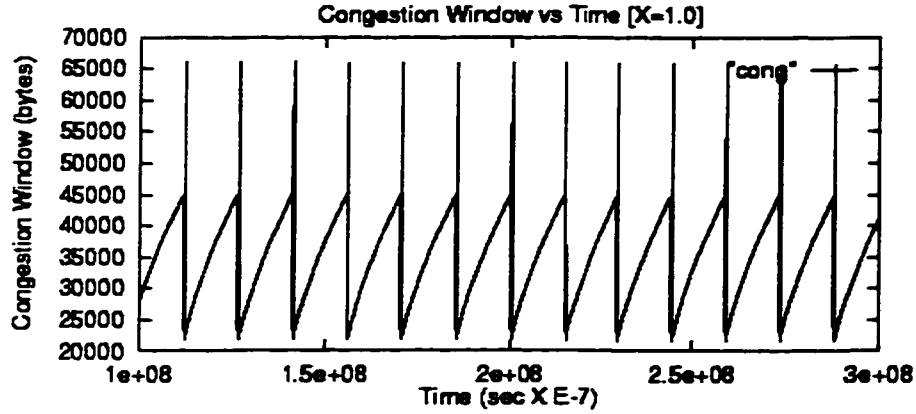
Graph 10: Exp1(RENO off) – Throughput vs Time [X=3.0]



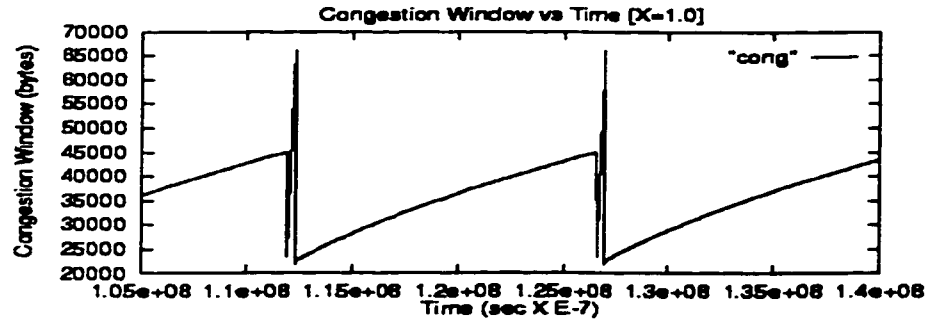
Graph 11: Exp1(RENO on) – Congestion vs Time [X=0.1]



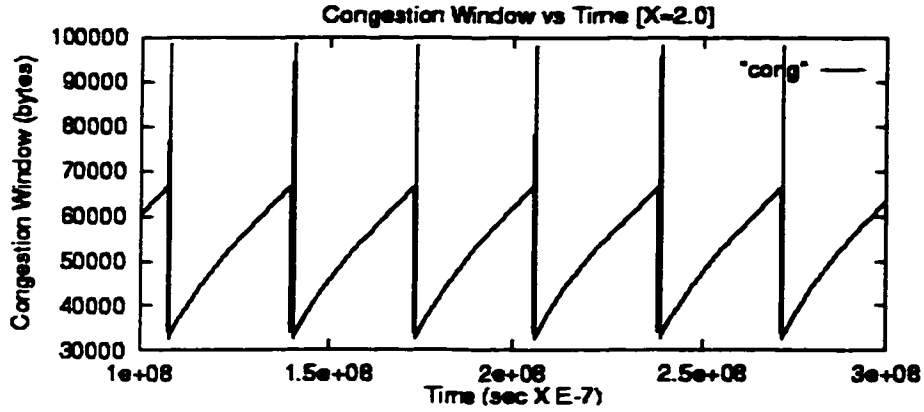
Graph 12: Exp1(RENO on) – Congestion vs Time [X=0.1]; time slice = 10.5 to 11.5 sec



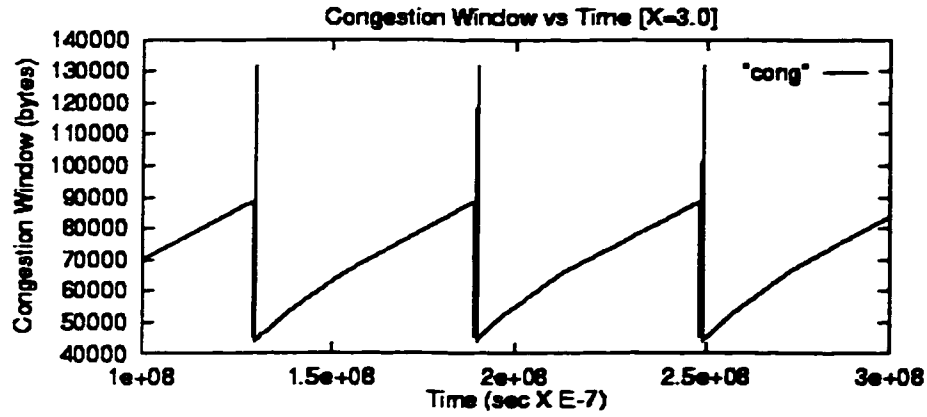
Graph 13: Exp1(RENO on) – Congestion vs Time [X=1.0]



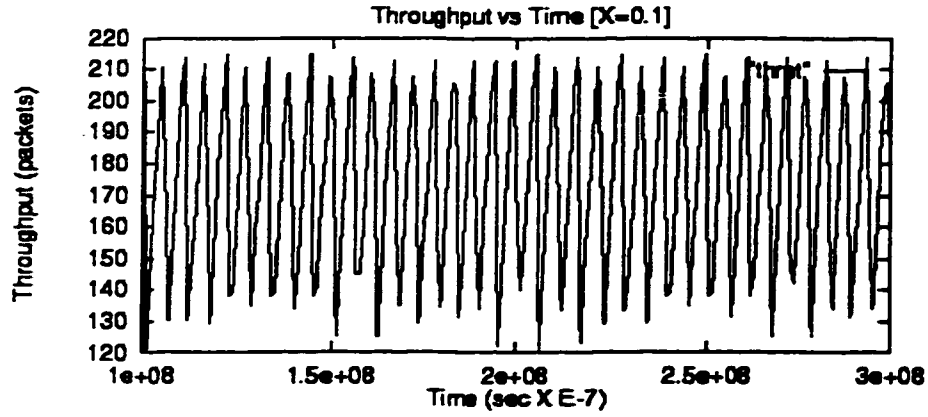
Graph 14: Exp1(RENO on) – Congestion vs Time [X=1.0]; time slice = 10.5 to 14.0 sec



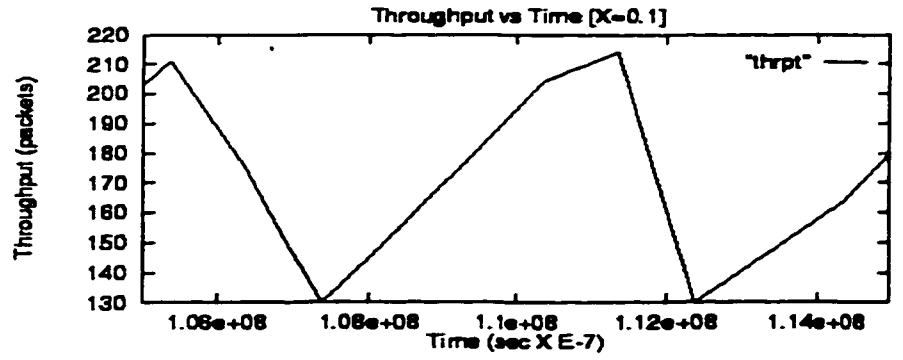
Graph 15: Exp1(RENO on) – Congestion vs Time [X=2.0]



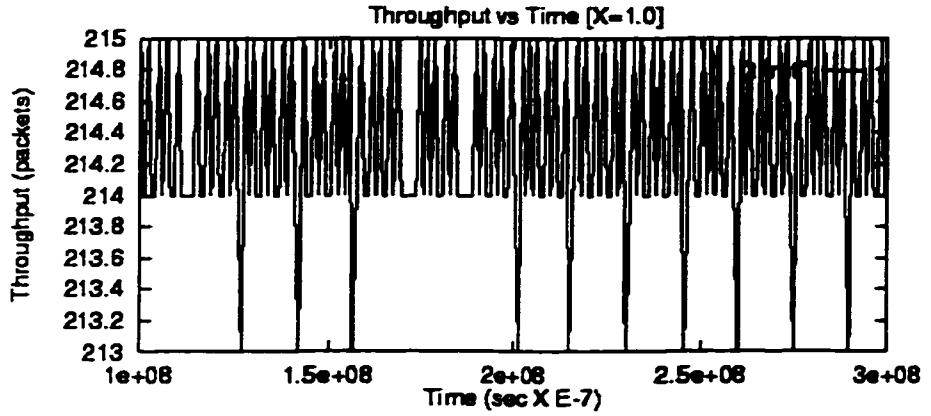
Graph 16: Exp1(RENO on) – Congestion vs Time [X=3.0]



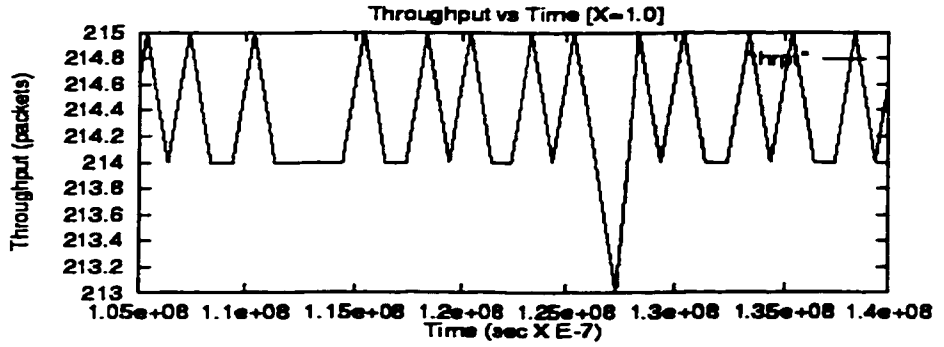
Graph 17: Exp1(RENO on) – Throughput vs Time [X=0.1]



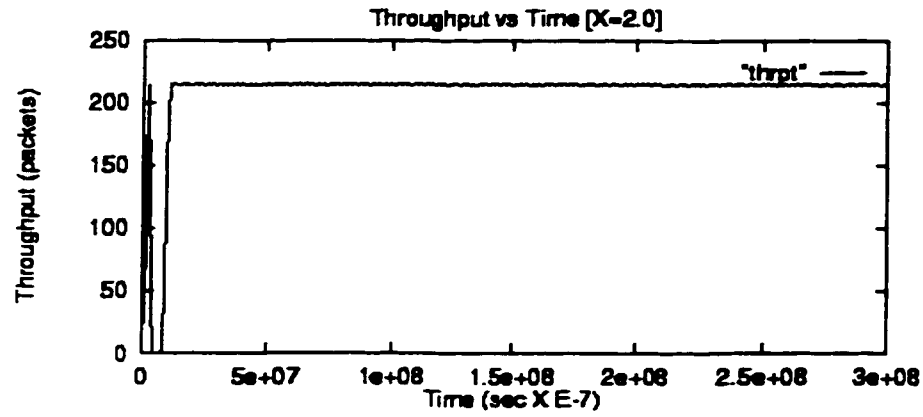
Graph 18: Exp1(RENO on) – Throughput vs Time [X=0.1]; time slice = 10.5 to 11.5 sec



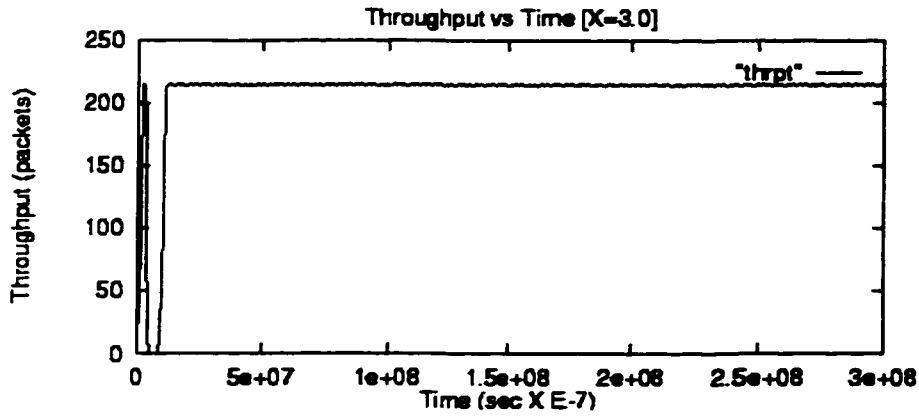
Graph 19: Exp1(RENO on) – Throughput vs Time [X=1.0]



Graph 20: Exp1(RENO on) – Throughput vs Time [X=1.0]; time slice = 10.5 to 14.0 sec



Graph 21: Exp1(RENO on) – Throughput vs Time [X=2.0]



Graph 22: Exp1(RENO on) – Throughput vs Time [X=3.0]

3.2.2 Experiment 2: 2 connections

The previous experiment showed how TCP behaved over a basic fixed bandwidth link, with no competition for bandwidth, since only one TCP host was sending to only one other receiving TCP host. When introducing another host sending/receiving pair, the single link between the two switches becomes the bottleneck. The allocation of bandwidth among the two connections is tested with simulation runs using the UBR service contract and RENO flag. We will once again vary the switch buffer sizes, but only for a few values since the true interesting results will be in the bandwidth fairness and efficiency of use for each connection. In

addition, we will be measuring the evolution of TCP throughput over time while the simulation runs from cycle to cycle of steady state performance. ABR simulations are performed just for the sake of completeness. The experiments involving finite buffers will be the true test scenario for ABR benefits/shortcomings.

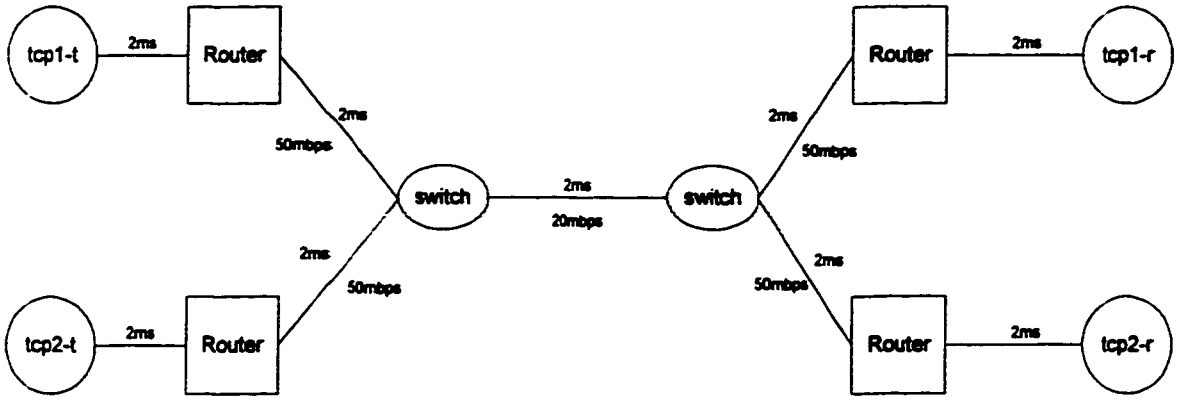


Figure 7: Experiment 2 – 2 connections with equal RTT

UBR with RENO off

X	Buffer Size	Throughput	Throughput	loss	Loss	rxt	rxt	Fairness	Efficiency
	(bytes)	TCP1 (mbps)	TCP2 (mbps)	TCP1 (pkts)	TCP2 (pkts)	TCP1 (pkts)	TCP2 (pkts)	(%)	(%)
0.10	5000	3.10	3.10	1177	1188	155	156	1.00	31
0.75	37500	8.05	7.69	1298	1045	204	150	0.99	79
1.00	50000	8.03	8.04	1331	1100	215	173	1.00	80
2.00	100000	8.29	8.29	1540	1529	267	265	1.00	83

Table 5: Experiment 2 – UBR with RENO off

UBR with RENO on

X	Buffer Size	Throughput	Throughput	loss	Loss	rxt	rxt	Fairness	Efficiency
	(bytes)	TCP1 (mbps)	TCP2 (mbps)	TCP1 (pkts)	TCP2 (pkts)	TCP1 (pkts)	TCP2 (pkts)	(%)	(%)
0.10	5000	7.07	6.90	825	836	76	77	0.99	70
0.75	37500	8.37	8.57	1331	1067	209	158	0.99	85
1.00	50000	8.64	8.57	1353	1122	217	175	0.99	86
2.00	100000	8.62	8.59	1540	1529	267	265	0.99	86

Table 6: Experiment 2 – UBR with RENO on

ABR with RENO off

X	Buffer Size	Throughput	Throughput	loss	Loss	rxt	rxt	Fairness	Efficiency
	(bytes)	TCP1 (mbps)	TCP2 (mbps)	TCP1 (pkts)	TCP2 (pkts)	TCP1 (pkts)	TCP2 (pkts)	(%)	(%)
0.10	5000	8.48	8.48	0	0	0	0	1.00	85
0.75	37500	8.48	8.48	0	0	0	0	1.00	85
1.00	50000	8.48	8.48	0	0	0	0	1.00	85
2.00	100000	8.48	8.48	0	0	0	0	1.00	85

Table 7: Experiment 2 – ABR with RENO off

ABR with RENO on

X	Buffer Size	Throughput	Throughput	loss	Loss	rxt	Rxt	Fairness	Efficiency
	(bytes)	TCP1 (mbps)	TCP2 (mbps)	TCP1 (pkts)	TCP2 (pkts)	TCP1 (pkts)	TCP2 (pkts)	(%)	(%)
0.10	5000	8.48	8.48	0	0	0	0	1.00	85
0.75	37500	8.48	8.48	0	0	0	0	1.00	85
1.00	50000	8.48	8.48	0	0	0	0	1.00	85
2.00	100000	8.48	8.48	0	0	0	0	1.00	85

Table 8: Experiment 2 – ABR with RENO on

When only one source is transmitting over a network, all allocable resources are assigned to that source. When multiple sources are using the same communications link, there becomes a competition for bandwidth. In this section, we examine the characteristics of network bandwidth usage and fairness for multiple TCP sources. Experiment 2 was performed with 2 sources, each with the same RTT and causing a bottleneck in the ATM network on the inter-switch link. Pages 41 and 42 display the tables created from the data obtained from experiment 2. For these tables, we are interested in the link usage efficiency and allocation fairness provided by the network to the multiple sources. The calculations for the fairness index [10], is as follows:

$$Fairness\ Index = (\sum x_i/y_i)^2/(n*\sum (x_i/y_i)^2)$$

where x_i is the allocated throughput, y_i is the optimal throughput and n is the number of TCP sources.

Simulations were run for a few different switch buffer sizes, and also with and without Fast Retransmit & Recovery (FRR). Table 5 and Table 6 show that complete fairness and high efficiency was achieved for 2 TCP sources employing UBR, with

slightly higher efficiency values when RENO is turned ON. These results are in agreement with the trends discovered in the previous experiment when only one TCP source was used. The Fast Recovery makes use of the available space in the TCP pipe, due to packet loss, to raise the congestion window well above the Delay-Bandwidth plus buffer size value. Also, the source does not start in slow-start mode and does not reset the *cwnd* down to one segment. This allows the source to maintain its high throughput. Table 7 and Table 8 show the results of using ABR. The results do not provide a fair comparison to UBR findings, because infinite buffering is employed. Experiments 4 to 6 will provide scenarios that are more realistic.

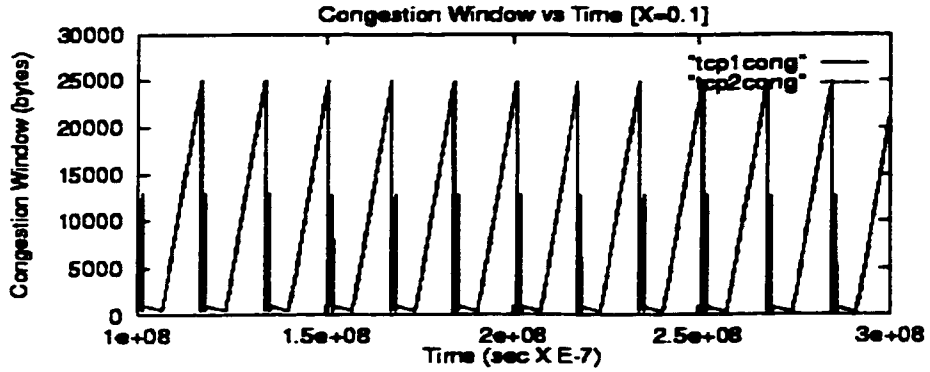
Pages 45 and 46 display congestion window traces for different switch buffer sizes. The experiments are done without Fast Recovery. They show that both traces, for TCP1 and TCP2, are superimposed. This implies that both are being allocated an equal amount of the bandwidth. This is in accordance with the fairness column of Table 5. For the graph X=0.1, both traces experience network congestion and timeout at the same time. As the buffer size is increased, the window size peak value increases but the frequency of packet loss and Fast Retransmit decreases. When packet loss is experienced the window drops to one segment and the source enters slow-start mode until it reaches *ssthresh*, at which time it switches to congestion avoidance. This pattern is seen in all four Congestion vs Time Graphs (Graph 23 to 26), as the buffer size is increased. Comparing these results with the corresponding ones from experiment 1, we see that not only do the patterns match but so do the peak values of the congestion window. The corresponding throughput traces are illustrated on pages 46 to 45. There are two traces for X=0.1 and two traces for X=0.7. Graph 28 and 30 are plotted over shorter time slices, showing more detail. The traces do not lineup exactly, but show that the sources are still receiving an equal amount of bandwidth when averaged over the entire simulation period. Examining Graph 28, we see that both sources timeout at 10.25 seconds and restart in congestion avoidance mode at 10.70 seconds, until a Fast Retransmit is issued at 11.75 seconds. During this congestion avoidance phase, the sources take turns in receiving slightly more throughput over the other. Graph 31and 32 display similar characteristics, however the larger buffer sizes keep the sources from timing out. The low points of the traces

correspond to the Fast Retransmit activation. The graphs that correspond to the results of Table 6 are displayed on pages 48 to 50. Again, we see the expected outcome of using Fast Recovery and increasing the buffer size.

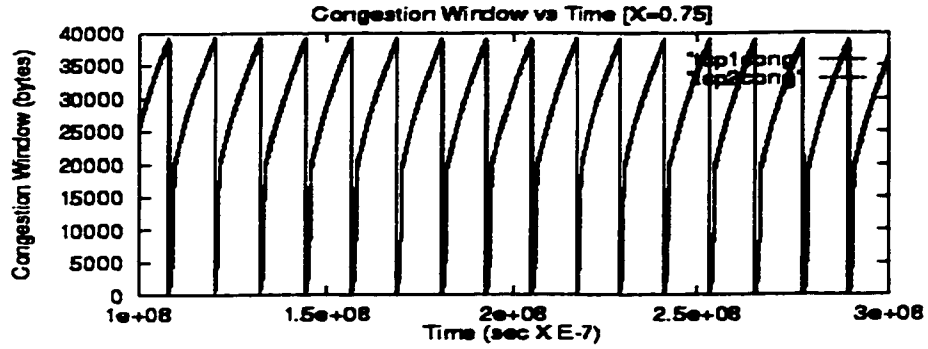
Fast Recovery is recognized by the sharp peaks that allows the source to increase its window size beyond the pipe capacity value and to avoid slow-start mode after a packet loss is experienced. The frequency at which this occurs is decreased as the buffer size is increased. The trace characteristics are equivalent, in most cases, to the corresponding one of experiment 1. Graph 34 shows that TCP2 is receiving more bandwidth between 15 and 20 seconds. The seemingly erratic throughput traces reflect the bandwidth sharing frequency of both sources. The sources are obtaining an equal amount of bandwidth when averaged over the simulation period. The only visible exception is where TCP2 is able to transmit at a higher throughput is between 15 and 20 seconds.

The ABR simulations from Table 7 and 8 do not show any interesting results. The corresponding graphs are shown on page 51. Since all congestion window traces were the same, only one was given for display. The same is true for the throughput traces. Experiments 4 to 6 will involve ABR and finite buffering, representing realistic situations,

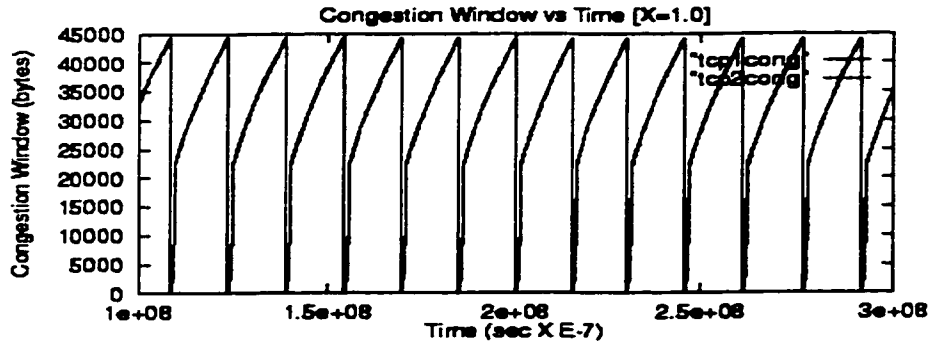
This experiment showed us that the behavior of two TCP sources sharing the same bottleneck link, exhibit the same characteristics as the situation when only one source is present. The same throughput patterns and cycles of congestion controls are seen. However, it remains to be seen if changing the Round Trip time of one source will effect the full fairness we saw in this experiment. Experiment 3 will explore this scenario and compare them with the results obtained in this simulation.



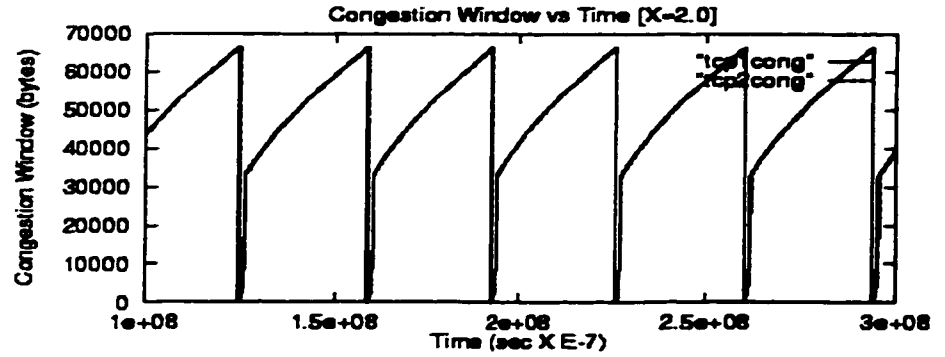
Graph 23: Exp2(RENO off) – Congestion vs Time [X=0.1]



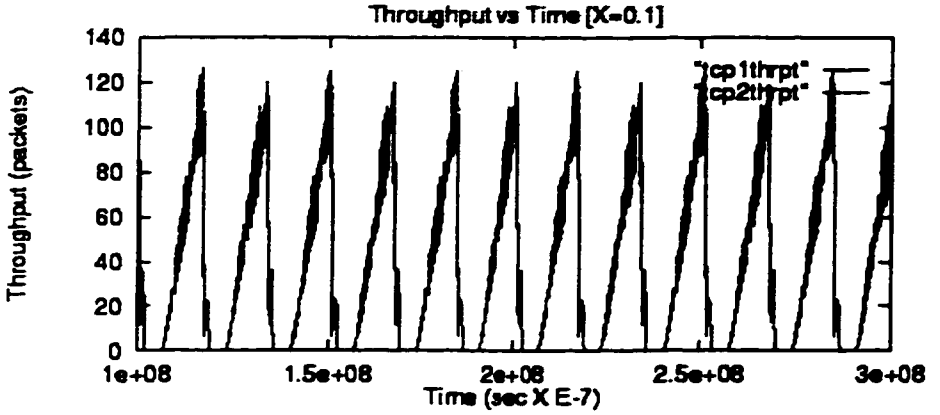
Graph 24: Exp2(RENO off) – Congestion vs Time [X=0.75]



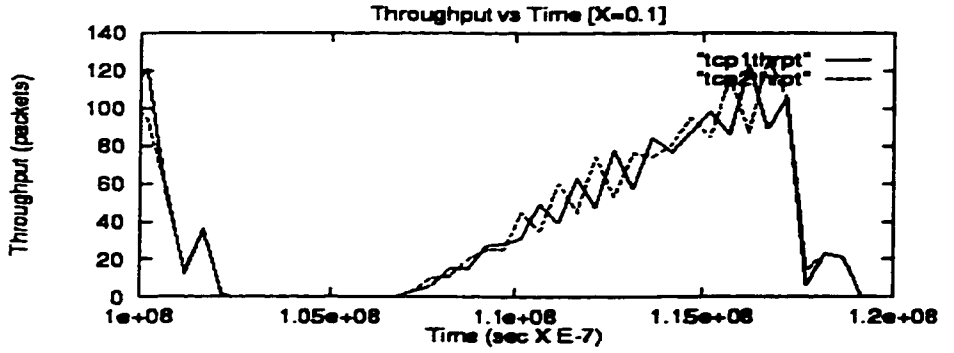
Graph 25: Exp2(RENO off) – Congestion vs Time [X=1.0]



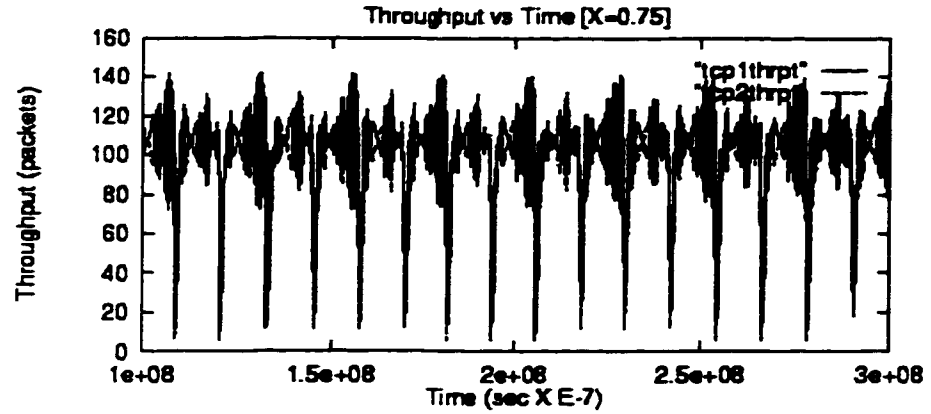
Graph 26: Exp2(RENO off) – Congestion vs Time [X=2.0]



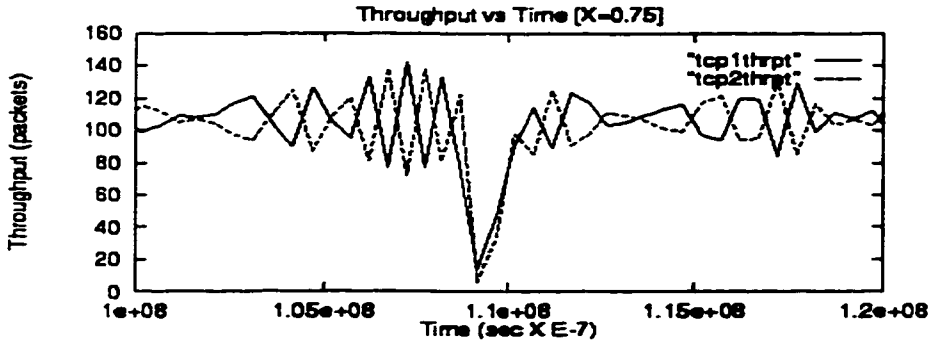
Graph 27: Exp2(RENO off) – Throughput vs Time [X=0.1]



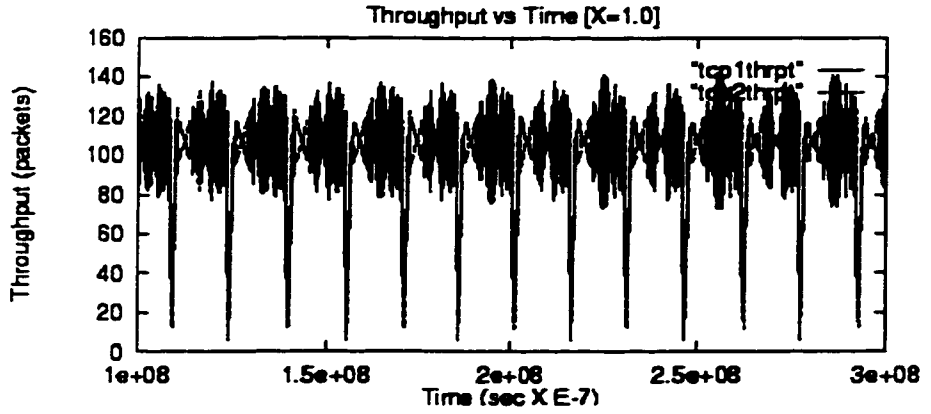
Graph 28: Exp2(RENO off) – Throughput vs Time [X=0.1]; time slice = 10 sec to 12 sec



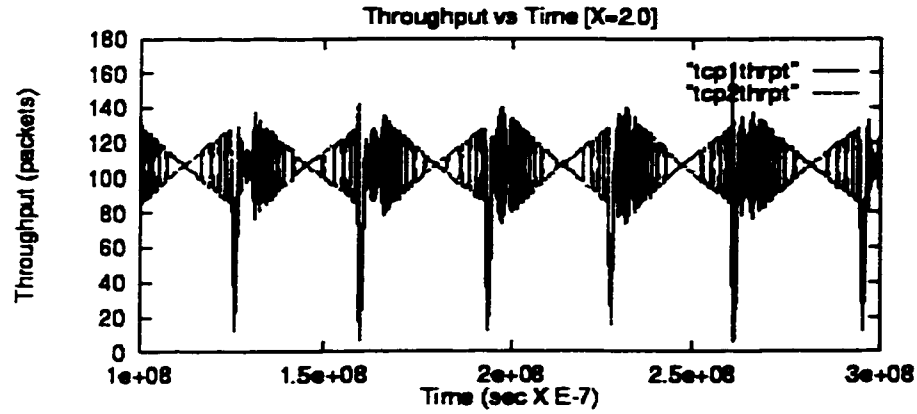
Graph 29: Exp2(RENO off) – Throughput vs Time [X=0.75]



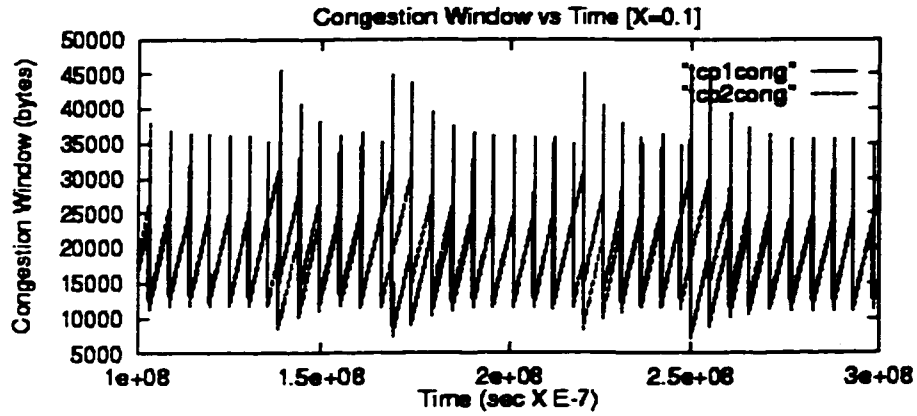
Graph 30: Exp2(RENO off) – Throughput vs Time [X=0.75]; time slice = 10 to 12 sec



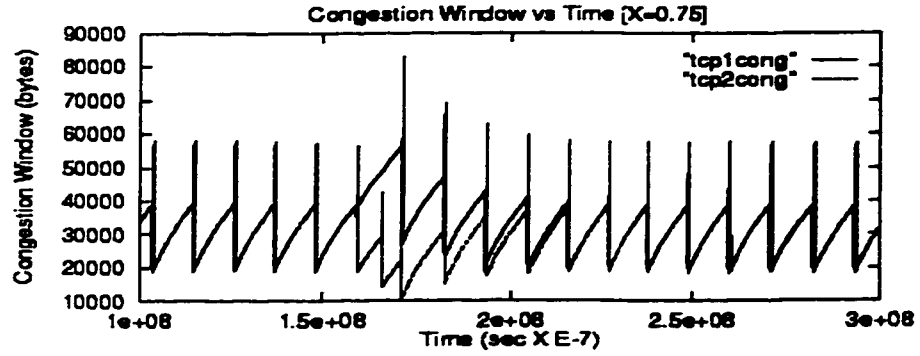
Graph 31: Exp2(RENO off) – Throughput vs Time [X=1.0]



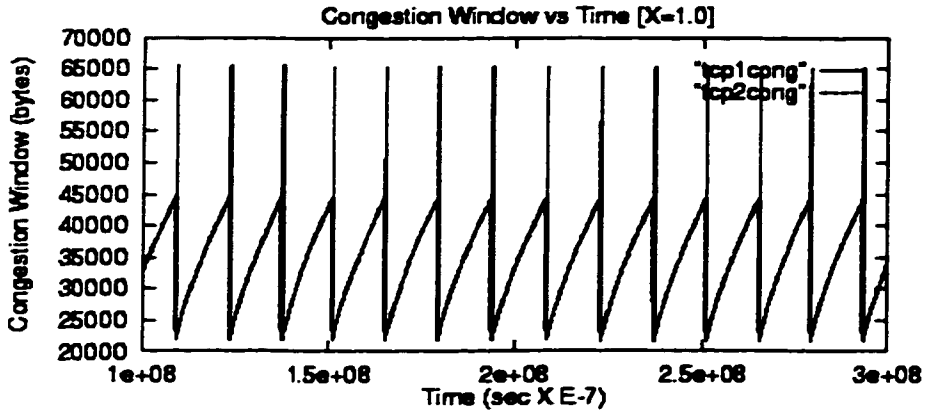
Graph 32: Exp2(RENO off) – Throughput vs Time [X=2.0]



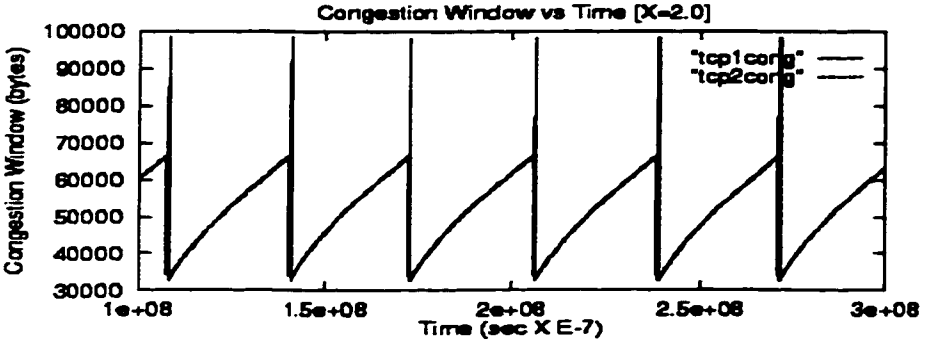
Graph 33: Exp2(RENO on) – Congestion vs Time [X=0.1]



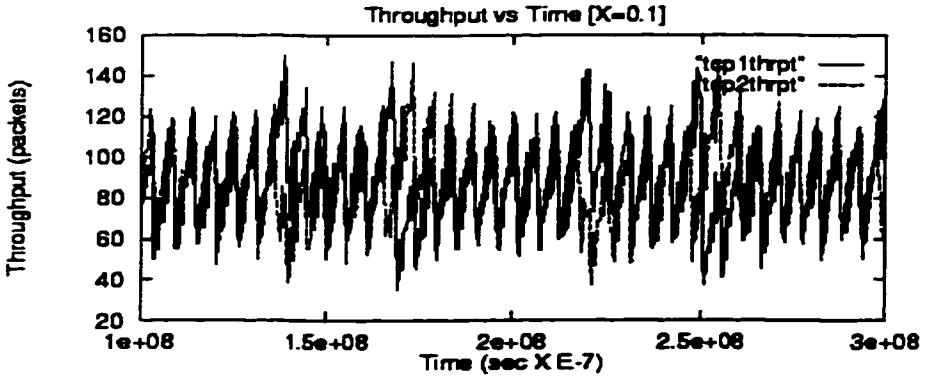
Graph 34: Exp2(RENO on) – Congestion vs Time [X=0.75]



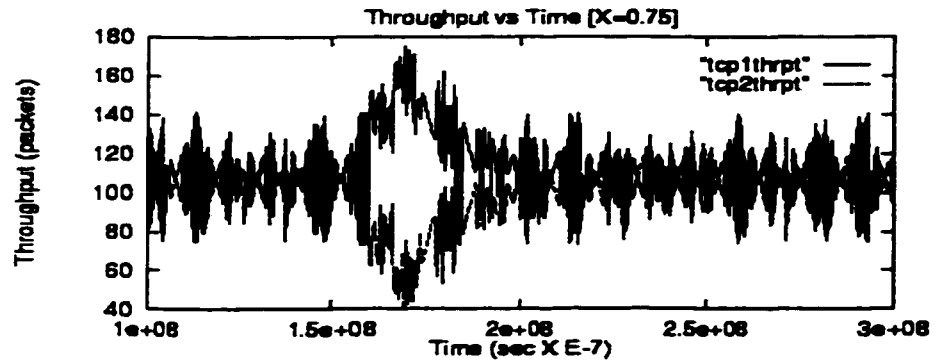
Graph 35: Exp2(RENO on) – Congestion vs Time [X=1.0]



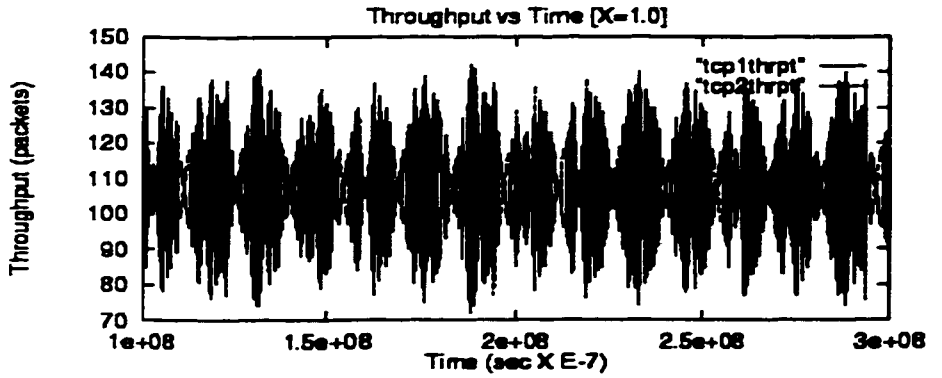
Graph 36: Exp2(RENO on) – Congestion vs Time [X=2.0]



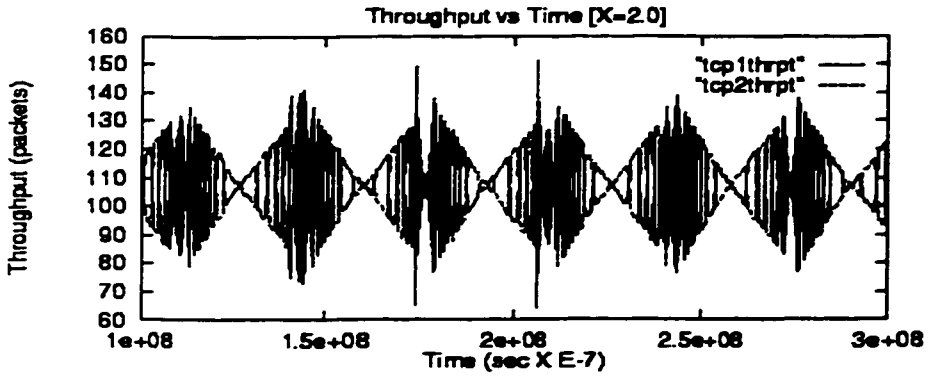
Graph 37: Exp2(RENO on) – Throughput vs Time [X=0.1]



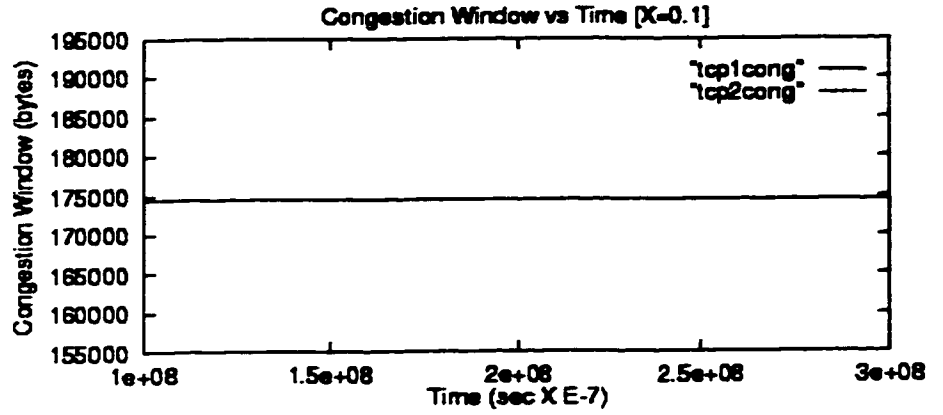
Graph 38: Exp2(RENO on) – Throughput vs Time [X=0.75]



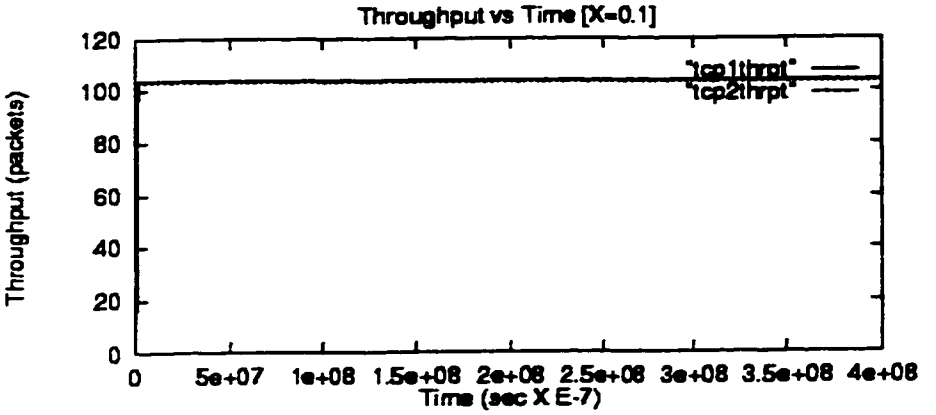
Graph 39: Exp2(RENO on) – Throughput vs Time [X=1.0]



Graph 40: Exp2(RENO on) – Throughput vs Time [X=2.0]



Graph 41: Exp2(ABR) – Congestion vs Time [X=0.1]



Graph 42: Exp2(ABR) – Throughput vs Time [X=0.1]

3.2.3 Experiment 3: 2 connections with different RTTs

The setup and final goals for this experiment are the same as for experiment 2, except that one connection will have an RTT value smaller than the other one. It is expected that the UBR service will not show fair bandwidth. Again, simulations runs will be executed for both RENO activated and deactivated. The ABR service is used for completeness only and the results from the finite buffered router experiments will be compared with the data obtained here.

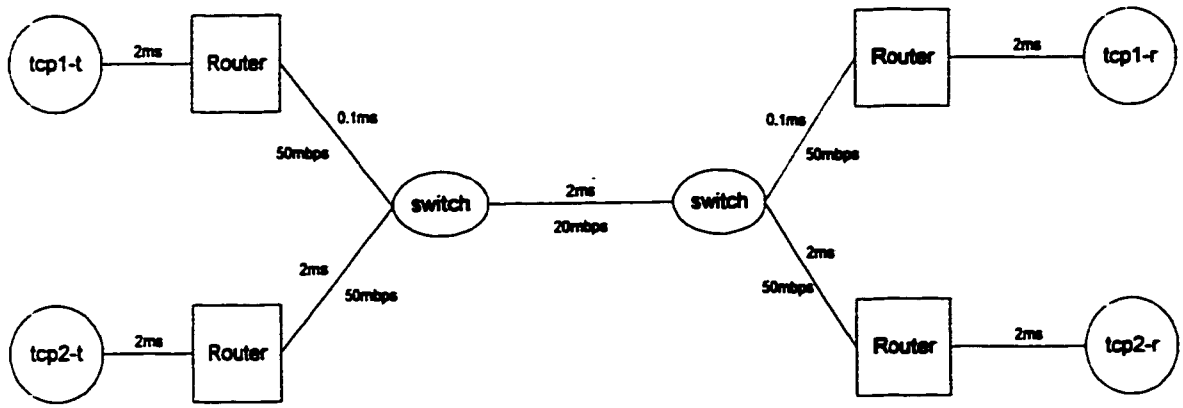


Figure 8: Experiment 3 – 2 connections with different RTT

UBR with RENO off

X	Buffer Size (bytes)	Throughput TCP1 (mbps)	Throughput TCP2 (mbps)	loss TCP1 (pkts)	loss TCP2 (pkts)	rxt TCP1 (pkts)	rxt TCP2 (pkts)	fairness	Efficiency (%)
0.10	5000	3.09	1.62	2321	1738	314	231	0.91	24
0.75	37500	5.07	9.82	1738	407	234	50	0.91	74
1.00	50000	12.44	3.21	1848	781	295	99	0.74	78
2.00	100000	12.61	4.01	1661	715	310	99	0.79	83

Table 9: Experiment 3 – UBR with RENO off

UBR with RENO on

X	Buffer Size (bytes)	Throughput TCP1 (mbps)	Throughput TCP2 (mbps)	loss TCP1 (pkts)	loss TCP2 (pkts)	rxt TCP1 (pkts)	rxt TCP2 (pkts)	fairnes s	Efficiency (%)
0.10	5000	4.08	3.07	2508	1672	261	172	0.98	36
0.75	37500	5.73	10.99	2189	473	292	56	0.91	84
1.00	50000	13.72	3.42	1881	924	298	106	0.73	86
2.00	100000	13.47	3.65	1672	792	337	98	0.75	86

Table 10: Experiment 3 – UBR with RENO on

ABR with RENO off

X	Buffer Size (bytes)	Throughput TCP1 (mbps)	Throughput TCP2 (mbps)	loss TCP1 (pkts)	loss TCP2 (pkts)	rxt TCP1 (pkts)	rxt TCP2 (pkts)	fairness	efficiency (%)
0.10	5000	8.50	8.48	0	0	0	0	1.00	85
0.75	37500	8.50	8.48	0	0	0	0	1.00	85
1.00	50000	8.50	8.48	0	0	0	0	1.00	85
2.00	100000	8.50	8.48	0	0	0	0	1.00	85

Table 11: Experiment 3 – ABR with RENO off

ABR with RENO on

X	Buffer Size (bytes)	Throughput TCP1 (mbps)	Throughput TCP2 (mbps)	loss TCP1 (pkts)	loss TCP2 (pkts)	rxt TCP1 (pkts)	rxt TCP2 (pkts)	fairness	efficiency (%)
0.10	5000	8.50	8.48	0	0	0	0	1.00	85
0.75	37500	8.50	8.48	0	0	0	0	1.00	85
1.00	50000	8.50	8.48	0	0	0	0	1.00	85
2.00	100000	8.50	8.48	0	0	0	0	1.00	85

Table 12: Experiment 3 – ABR with RENO on

TCP 1 has an RTT of 12.4ms while TCP2 remains at 20ms. The delays for the links between the edge routers and ATM switches were reduced from 2ms to 0.1ms to create the smaller RTT for TCP1.

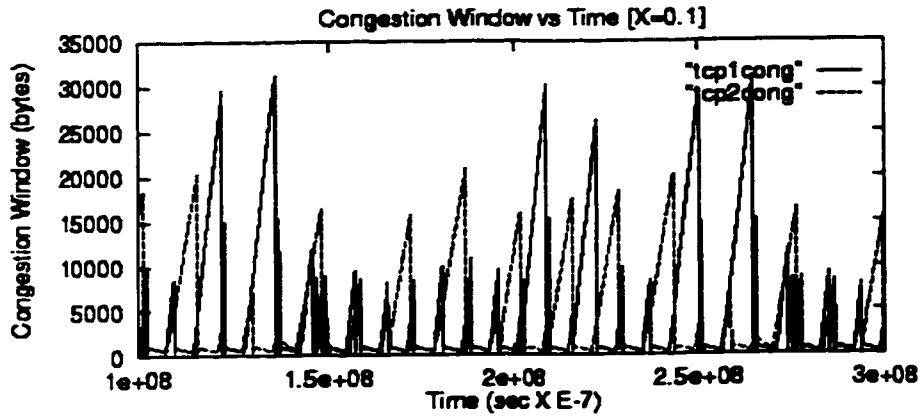
Table 9 summarizes the UBR without Fast Recovery results for the same four different buffer sizes. As expected, the fairness has dropped and the bandwidth efficiency has remained about the same. The previous experiment has showed a fairness index of 0.99 as compared to 0.74 in this simulation, for X=1.0. This unfairness is more evident from the throughput columns. TCP1, which has a smaller RTT, receives faster acknowledgement of packets and recovers faster from timeout and lost packets. The only case where this not so, is when the buffer size is slightly less than the Delay-Bandwidth product as shown on the X=0.75 row. In this situation, the buffer fills to a point where TCP1 congestion window is large and already has a significant number of packets on the path. It then has to stop transmitting and recover from packet loss, due to a timeout. In the meantime, TCP2 can benefit from the allocated bandwidth, while TCP1 recovers. The congestion window Graph 43 to 46 and throughput Graph 47 to 50 also show the unfairness. The traces are not superimposed, as was the case in experiment 2 when both sources had the same RTT. Page 55 shows the two graphs for X=0.1. The second displays a shorter time slice for time 10 to 15 seconds. The TCP1 trace is receiving more of the bandwidth, indicated by the greater peak window sizes. Page 56 shows the traces when the buffer is increased to a factor of X=0.75 and X=2.0. We noticed in the previous experiment that the only occurrence of timeouts was in the case when X=0.1. However this time TCP1 experiences a timeout in X=0.75 and TCP2 experiences a timeout in X=2.0.

This is shown on the graphs at the points where the traces flatten out at the bottom. Just before the source timeouts, it had issued a Fast Retransmit because of packet loss. The graphs for X=2.0 shows TCP1 peaking at four points. The following is observed when examining the final three peaks of this graph. At peak2, the source issues a Fast Retransmit and continues in slow-start mode until *cwnd* = ½(*cwnd of peak2*). At this point it switches to congestion avoidance and reaches peak3. The same steps occur and the phase switch occurs when *cwnd* = ½(*cwnd of peak3*). This pattern conforms to the basic Congestion Avoidance mechanism of TCP. The corresponding throughput traces for X=0.1 are given on page 56 and 54. Again, a shorter time slice shows that TCP1 is transmitting at a higher throughput than TCP2. The flat portions at the bottom indicate timeouts. The graph for X=1.0,on page 57, shows that both sources timeout together at 19.5 seconds and 23.0 seconds. It is quite evident here that TCP1 is receiving most of the bandwidth. The graph for X=2.0, illustrates the amount of extra throughput received by TCP1 over TCP2 and resulting in fewer timeouts.

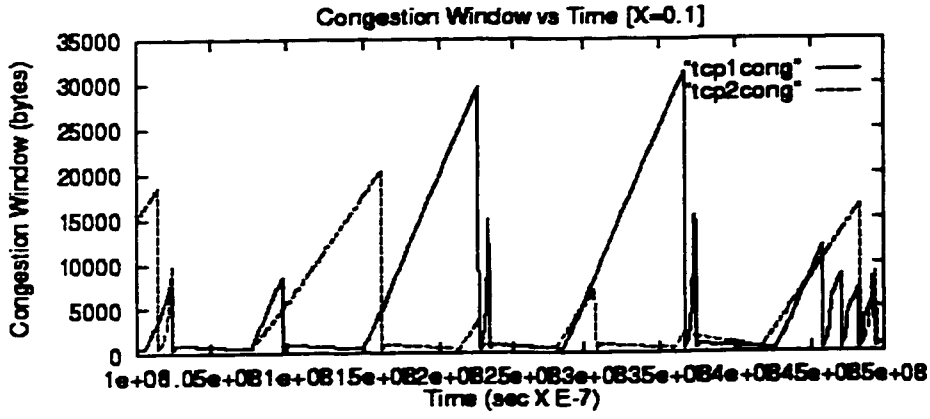
The results in Table 10 summarize the activation of Fast Recovery. Approximately the same amount of unfairness is seen. However, efficiency is increased slightly. The congestion window graphs on page 58 & 56 show the Fast recovery characteristics of sharp peaks and the absence of slow-start phases. The individual source traces for each graph match the patterns of the corresponding congestion window traces of Experiment 2. The difference is the obvious unfairness, indicated by the traces not being superimposed. The same can be shown for the throughput graphs on page 59 & 57. We can see from graphs X=0.75, X=1.0 and X=2.0 that as one source increases its throughput, the other decreases it. The maximum combined throughput can be calculated by adding the throughput of each source at any particular point in time. For example, on graph X=0.75 at 13.5 seconds, TCP2 = 215 packets and TCP1 = 0 packet. At 15 seconds, TCP2 = 82 packets and TCP1 = 133 packets. Each sum adds up to 215 packets. This is the maximum peak throughput seen with the previous two experiments. This shows how efficiency has remained the same for the three experiments.

ABR simulations are summarized in Table 11 & 12. These show complete fairness and efficiency. Each row is identical and therefore the resulting graphs would not be interesting to view. However, the congestion window and throughput plots for X=0.1 are shown on Graph 59 & 60.

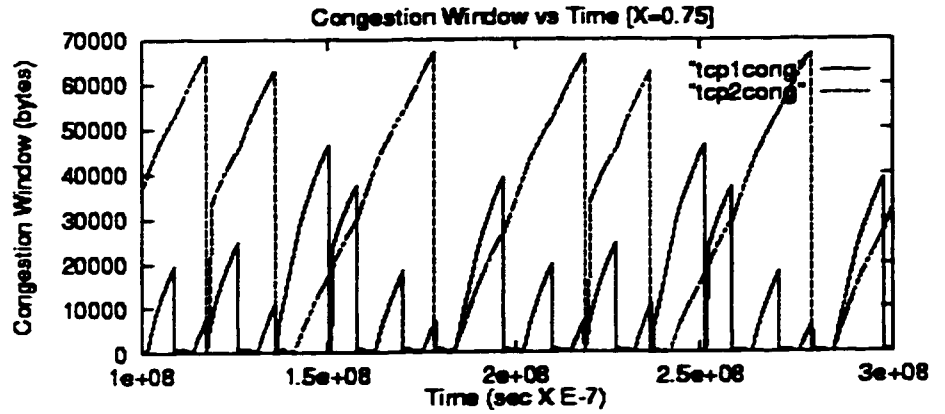
Experiments 2 and 3 provided an understanding of TCP behavior over UBR. Now we must investigate the effects over ABR. However, under the current configuration of infinite router buffering, we would not be simulating ABR in a realistic environment. The next experiment looks at TCP behavior over ABR with finite buffering.



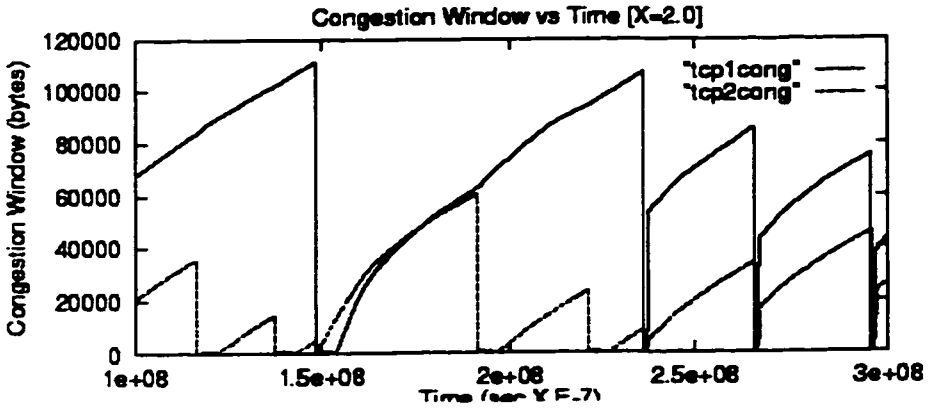
Graph 43: Exp3(RENO off) – Congestion vs Time [X=0.1]



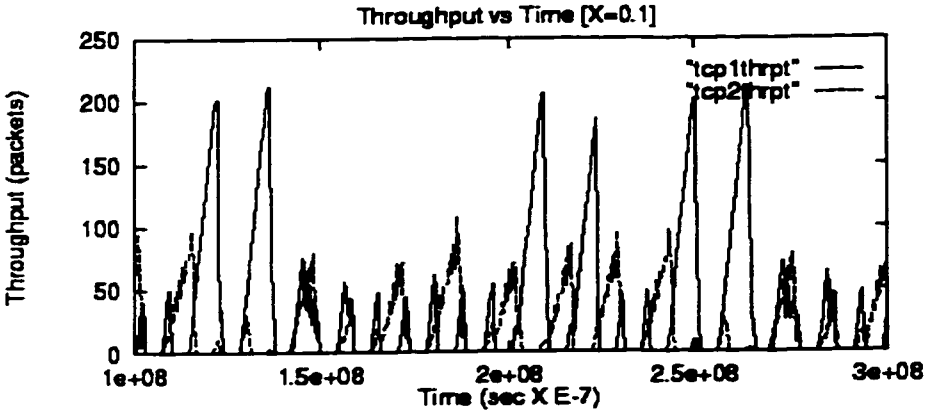
Graph 44: Exp3(RENO off) – Congestion vs Time [X=0.1]; time slice = 10 sec to 15 sec



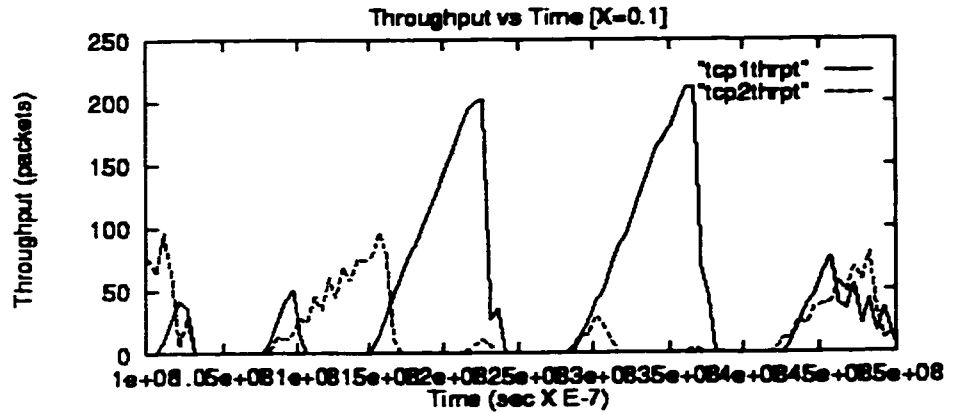
Graph 45: Exp3(RENO off) – Congestion vs Time [X=0.75]



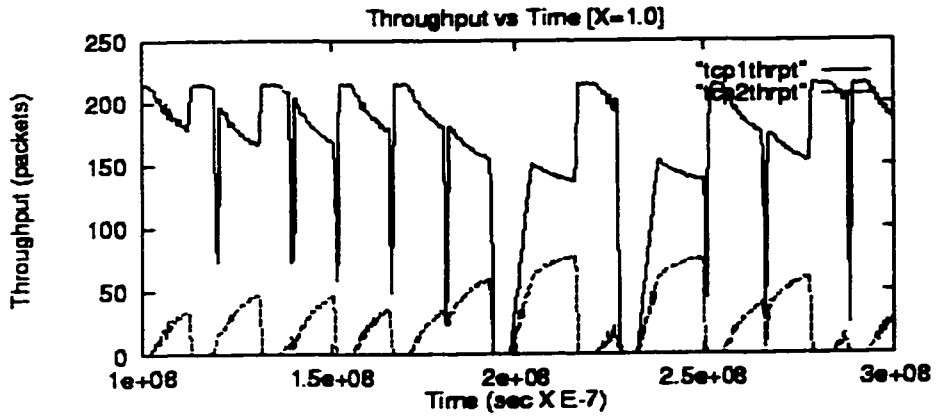
Graph 46: Exp3(RENO off) – Congestion vs Time [X=2.0]



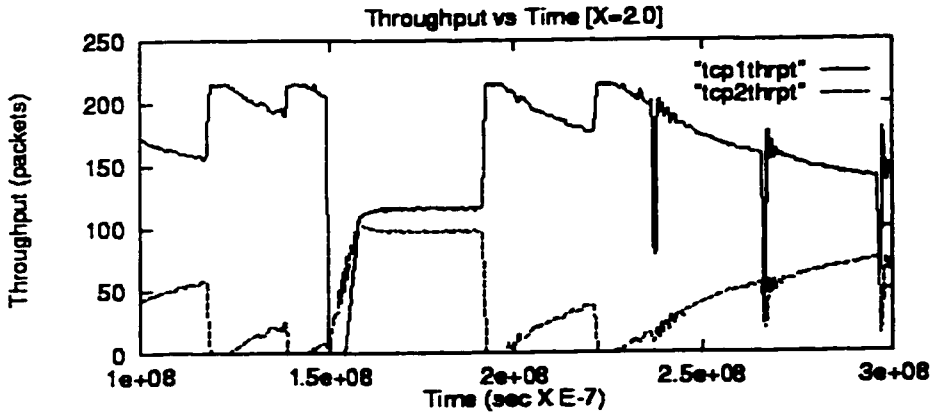
Graph 47: Exp3(RENO off) – Throughput vs Time [X=0.1]



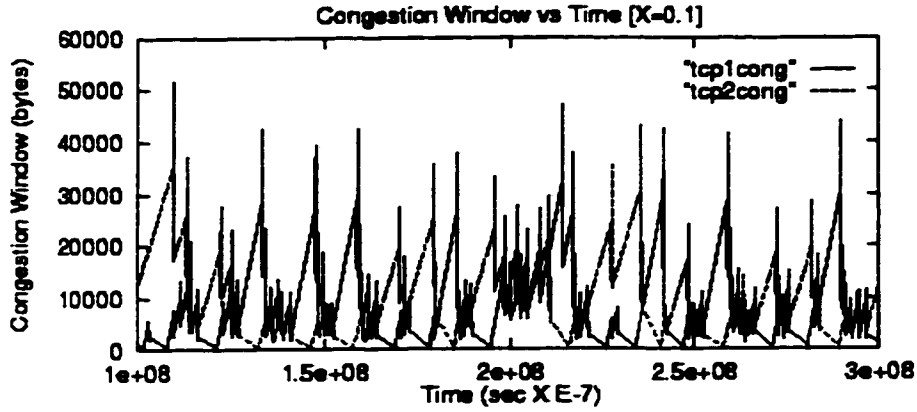
Graph 48: Exp3(RENO off) – Throughput vs Time [X=0.1]; time slice = 10 sec to 15 sec



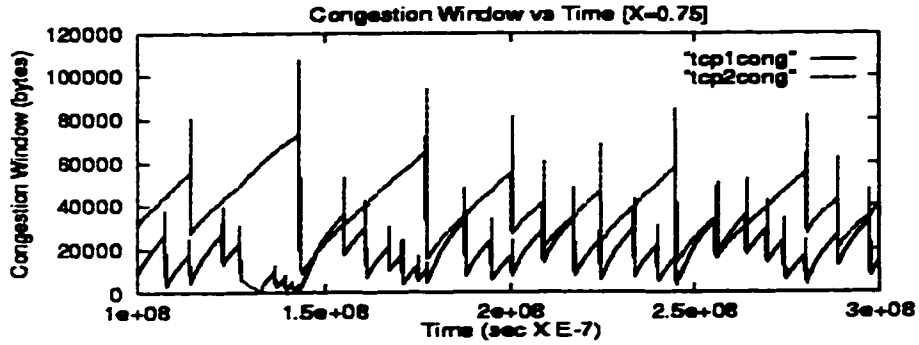
Graph 49: Exp3(RENO off) – Throughput vs Time [X=1.0]



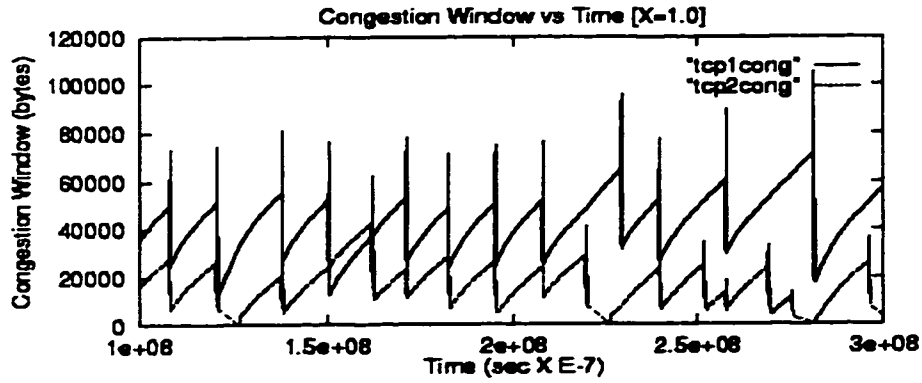
Graph 50: Exp3(RENO off) – Throughput vs Time [X=2.0]



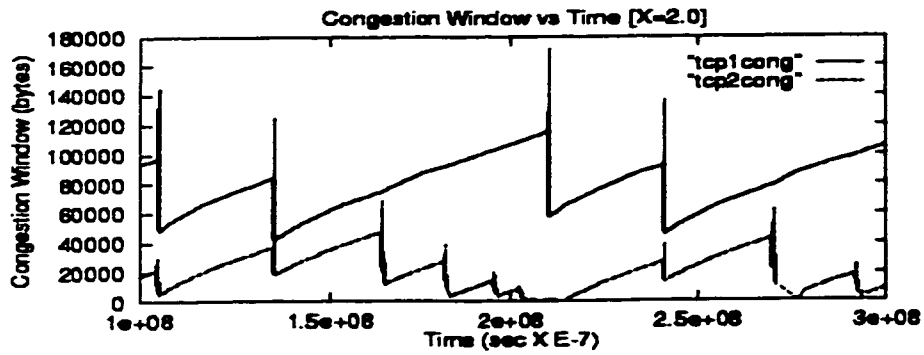
Graph 51: Exp3(RENO on) – Congestion vs Time [X=0.1]



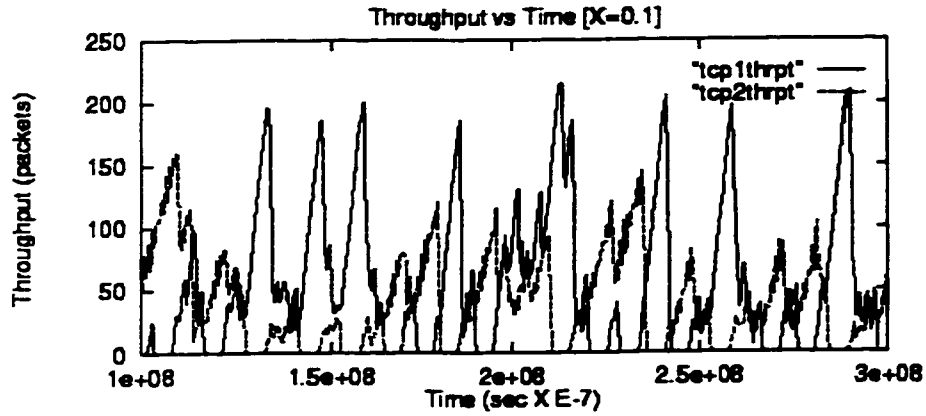
Graph 52: Exp3(RENO on) – Congestion vs Time [X=0.75]



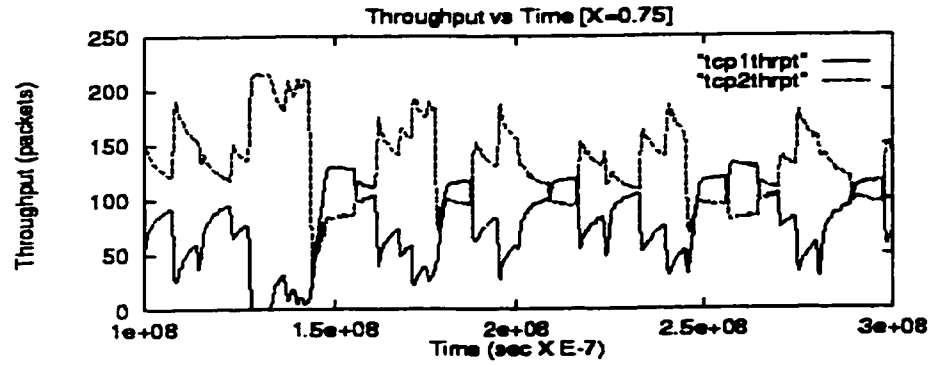
Graph 53: Exp3(RENO on) – Congestion vs Time [X=1.0]



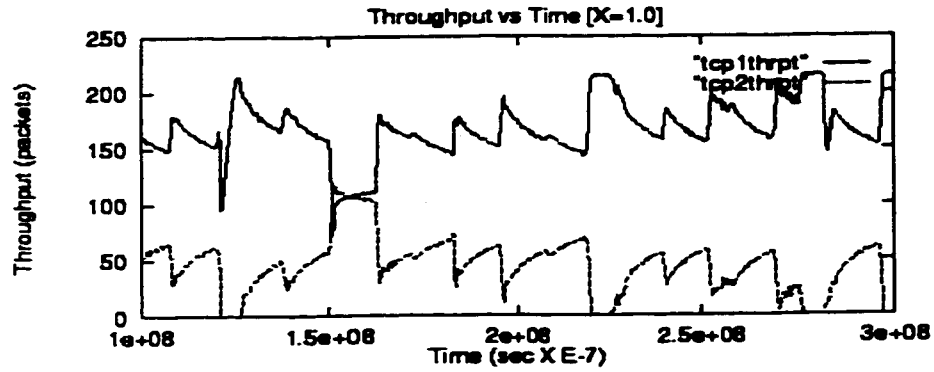
Graph 54: Exp3(RENO on) – Congestion vs Time [X=2.0]



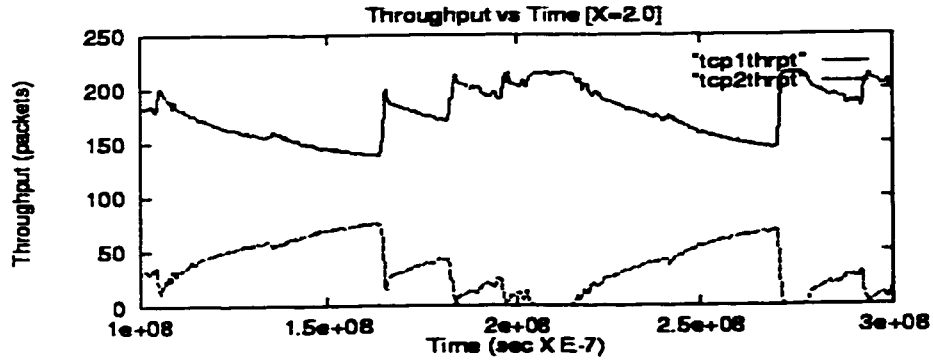
Graph 55: Exp3(RENO on) – Throughput vs Time [X=0.1]



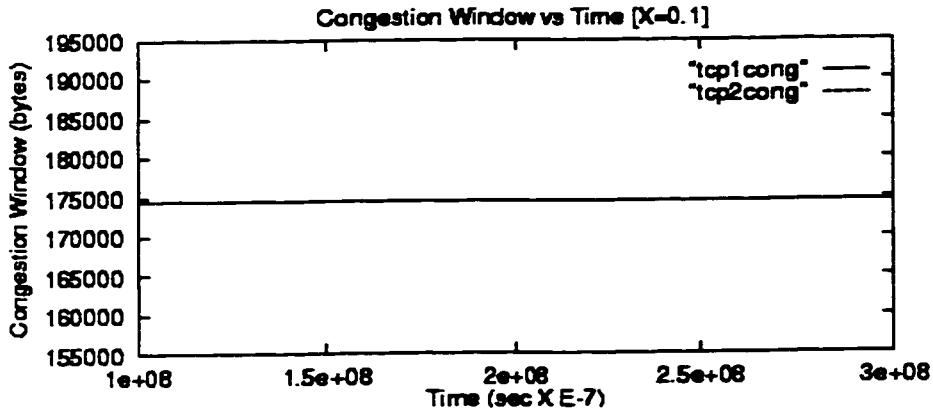
Graph 56: Exp3(RENO on) – Throughput vs Time [X=0.75]



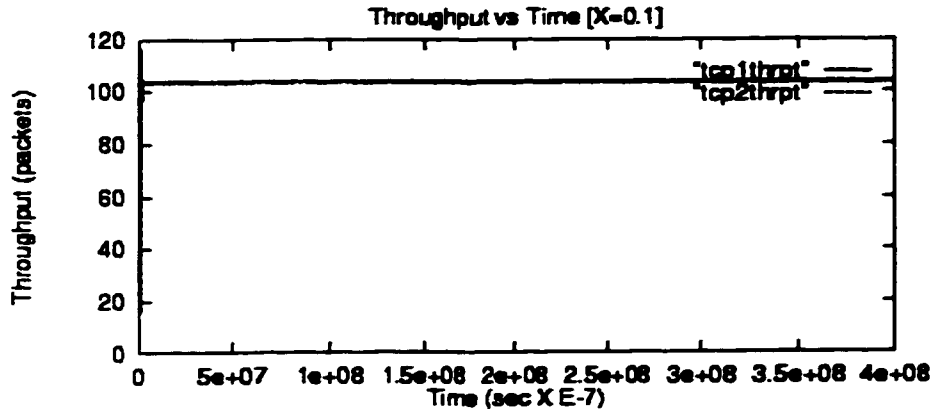
Graph 57: Exp3(RENO on) – Throughput vs Time [X=1.0]



Graph 58: Exp3(RENO on) – Throughput vs Time [X=2.0]



Graph 59: Exp3(ABR) – Congestion vs Time [X=0.1]



Graph 60: Exp3(ABR) – Throughput vs Time [X=0.1]

3.2.4 Experiment 4: 2 Connections with Finite Buffered Separate Routers

In experiment 2 we looked at how the bandwidth was allocated among two connections. In that scenario we assumed infinite buffering at the edge devices and put the load-balancing task on the ATM switch. Simulations using only ABR and without the RENO flag are used in this experiment. We are interested in a more realistic situation where congestion control is pushed to the ATM network edge devices employing finite buffering. The evolution of TCP throughput and the congestion window is measured and tabulated below. We will again look at the fairness and efficiency factors.

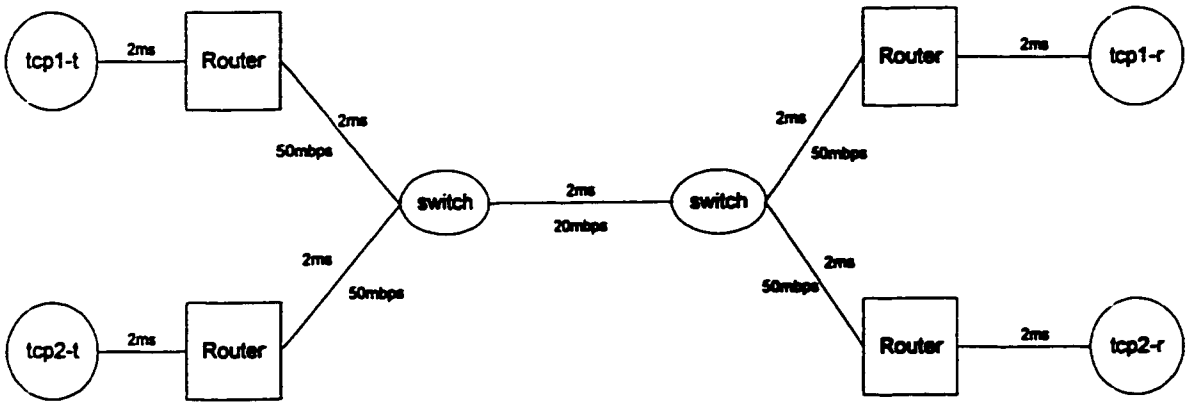


Figure 9 : Experiment 4 – 2 Connections with Separate Routers

ABR with Separate Routers

X	Router Size (bytes)	Buffer Size (bytes)	Throughput TCP1 (mbps)	Throughput TCP2 (mbps)	loss TCP1 (pkts)	loss TCP2 (pkts)	Pkts sent TCP1 (pkts)	Pkts sent TCP2 (pkts)	fairness (%)	efficiency (%)
0.10	2500	5000	2.99	2.99	1749	1749	28326	28326	1.00	30
0.75	18750	37500	7.58	7.58	946	946	74070	74070	1.00	76
1.00	25000	50000	7.76	7.76	1199	1199	75848	75848	1.00	78
2.00	50000	100000	8.02	8.02	1518	1518	78481	78481	1.00	80

Table 13: Experiment 4 - 2 Connections with Separate Routers

The previous two sections analyzed throughput evolution when infinite buffered routers were assumed. This is suitable for studying the effects on TCP when UBR is used, since the ATM switch will be controlling the bandwidth allocation. This section examines the information gathered when performing the same experiments, over finite buffered routers, but this time ABR is the focus. The ABR feedback loop is controlling the flow of packets onto the network. Therefore, measurement is useful only if finite buffers are used. In previous experiments we performed ABR simulation for the sake of completeness and found that no useful information was produced. The graphs showed continuous peak throughput and a sustained congestion window for the duration of the simulation.

The router buffer sizes are varied in the same fashion as the switch buffer size. All previous experiments used a switch buffer size that was equal to a factor of the Delay-Bandwidth product. The same is true in this case. In the scenarios where separate routers exist, the sum of the router buffer sizes is equal to the size of the switch buffer. Hence, in this setup with two connections, the size of each router buffer is equal to one-half of the ATM switch buffer. The inter-switch bottleneck link is set to 20 mbps and most connection RTT values have been set to 20 ms, based on a 2ms link delay.

Table 13 shows the switch and router sizes used as the X factor is varied. It is shown that as the router buffer size is increased, so does the TCP throughput and the number of packets sent. Complete fairness is seen, because each connection is being routed through separate devices and hence cells from only that connection are being

buffered in the router. When the X factor is 2, the TCP sources experience approximately 80% combined usage of the bandwidth. The maximum possible usage is around 85%, as explained in sub-section 3.3. Graphs on pages 64 to 63 show the throughput and congestion evolution over time for each X factor. The graph for X=0.1 is the only one that indicates that the source timeouts. All others illustrate a sustained throughput for longer periods, with Fast Retransmissions in between each congestion avoidance phase. As X gets larger, so do the periods during which the sources obtain maximum throughput. The Congestion vs Time Graph 61 to Graph 65, show that the two TCP traces are superimposed on top of one another. This reflects the complete fairness values shown in Table 13. Graph 62 exhibits the same characteristics as all other congestion plots of previous experiments with UBR. The source is in congestion avoidance phase after restarting from a timeout and receives three duplicate ACKs before issuing a Fast Retransmit at 11.6 seconds. The connection then goes into the slow start phase and continues until another Fast Retransmit is issued a short time later, after which it timeouts again. Graph 63 to Graph 65, show no timeouts and follow the same expected pattern of starting in slow start until *cwnd* = *ssthresh* and then continuing in congestion avoidance mode until three duplicate ACKs invoke a Fast Retransmit. The throughput Graph 66 to Graph 70, show the same expected traces for full fairness among 2 source having the same RTT. However, we notice that there is less change in throughput values between the sources. For example, if we compare Graph 70 and Graph 32, there is less fluctuation when ABR is used. The UBR traces show changes that are more drastic in throughput as compared to the smooth ABR trace. This shows the better control that the ABR flow control has on polling and reacting to network congestion.

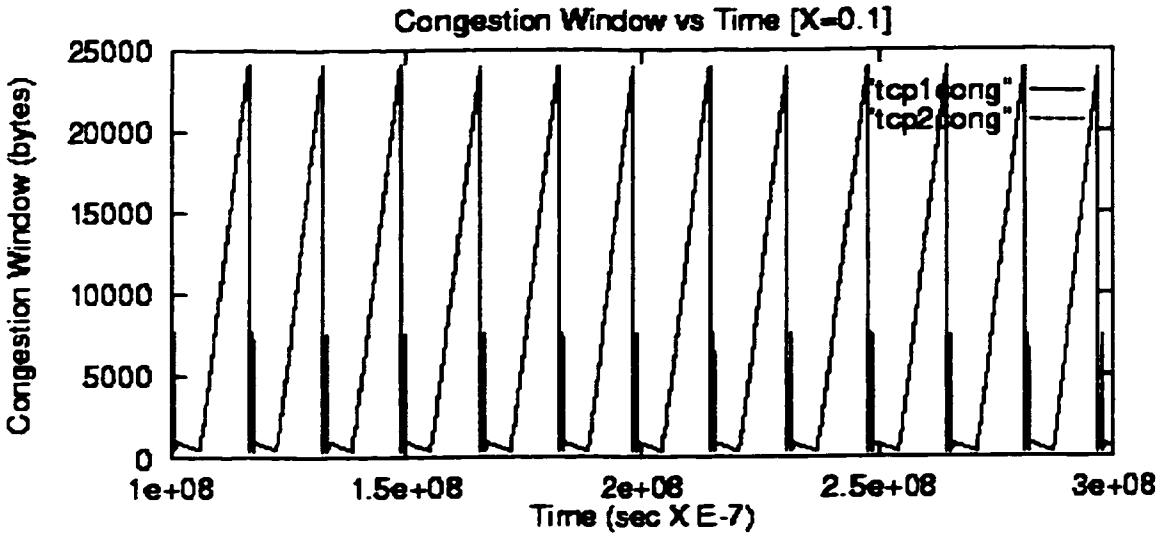
Table 14 summarizes the CB ratios calculated for experiments involving 2 connections having the same RTT and passing through separate routers. The simulations for both UBR without Fast Recovery and ABR without Fast Recovery are displayed in this table. The CB ratios are almost the same and complete fairness is displayed for both service classes. In this case, where two sources have unique but equally delayed paths, both UBR and ABR scenarios show very similar results and

hence there is no advantage of one setup over the other, in terms of throughput and efficiency.

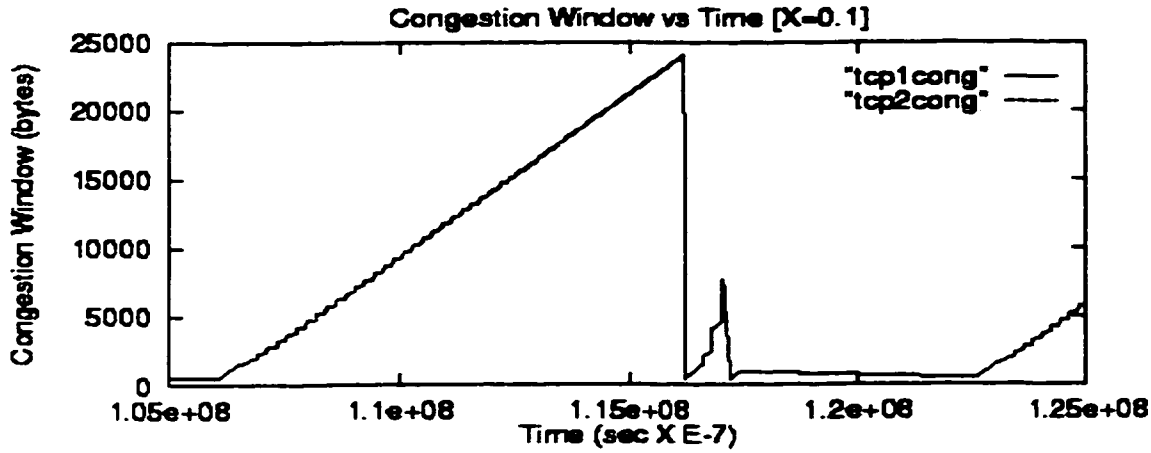
X	Packet Loss		Fairness Index		% Efficiency		CB Ratio	
	UBR	ABR	UBR	ABR	UBR	ABR	UBR	ABR
0.10	2365	3498	1.00	1.00	31	30	76.29	116.60
0.75	2343	1892	0.99	1.00	79	76	29.65	24.89
1.00	2431	2398	1.00	1.00	80	78	30.39	30.74
2.00	3069	3036	1.00	1.00	83	80	36.98	37.95

Table 14: CB Ratio comparison for 2 connections with same RTT

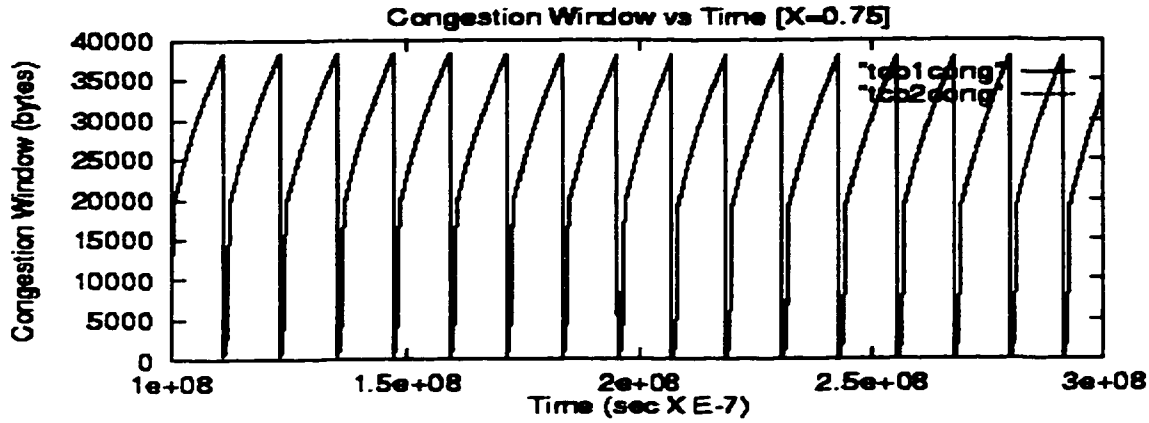
We have just seen how ABR compares with UBR in the behavior exhibited by 2 sources with the same RTT. The next logical step is to compare the UBR results of experiment 3 with ABR, when 2 sources with different RTTs are implemented. This is the purpose of experiment 5.



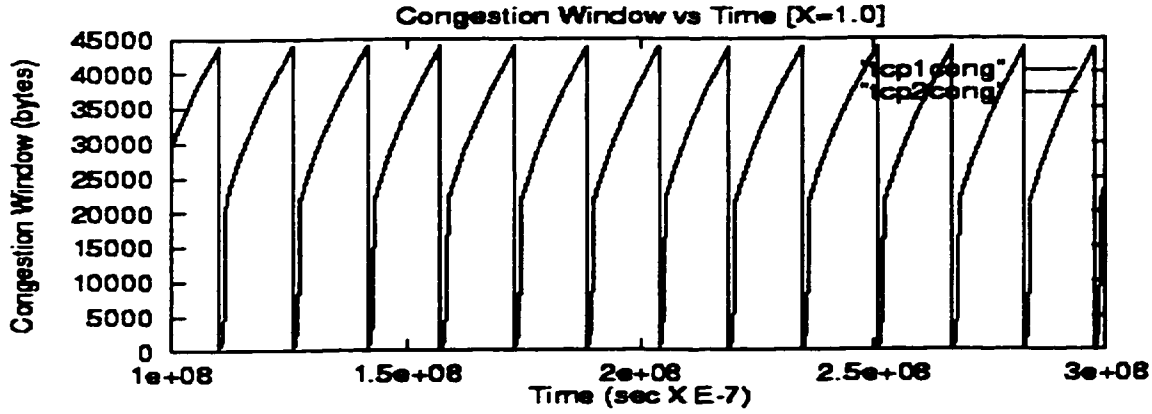
Graph 61: Exp4 – Congestion vs Time [X=0.1]



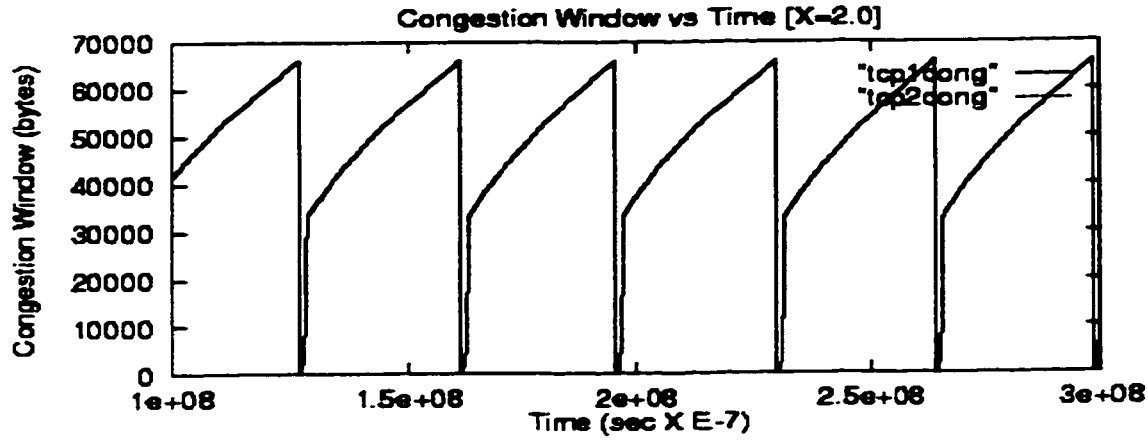
Graph 62: Exp4 – Congestion vs Time [X=0.1]; time slice = 10.5 sec to 12.5 sec



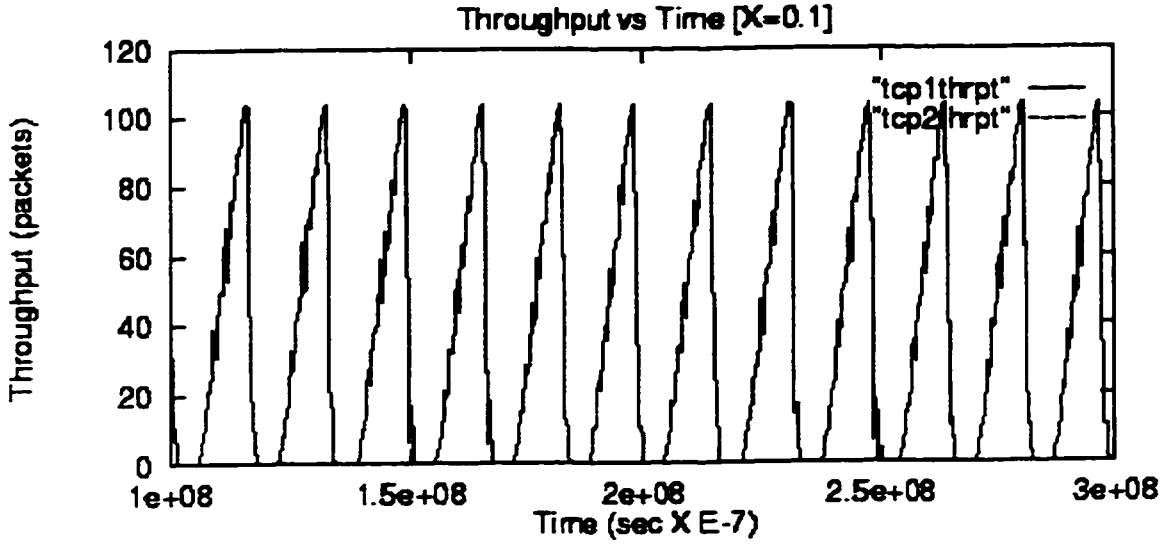
Graph 63: Exp4 – Congestion vs Time [X=0.75]



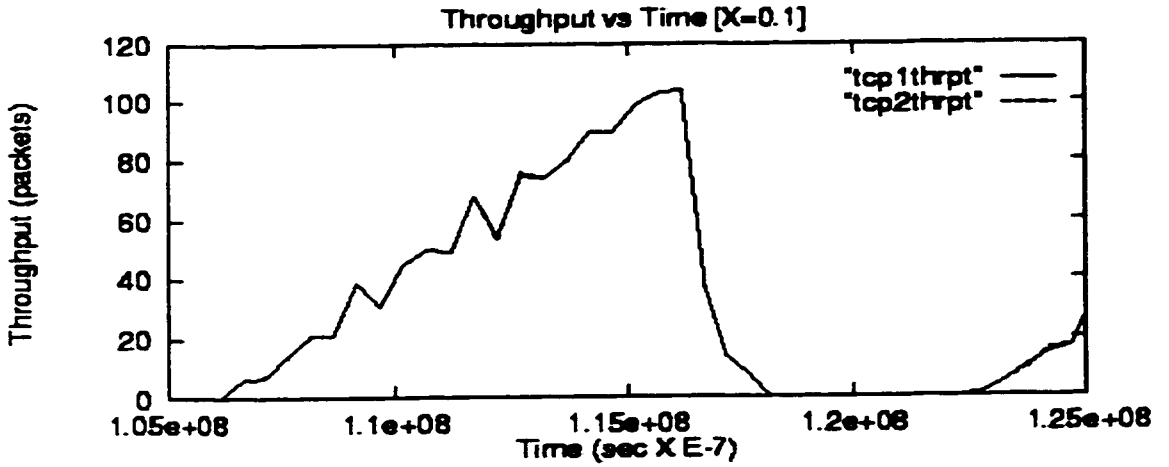
Graph 64: Exp4 – Congestion vs Time [X=1.0]



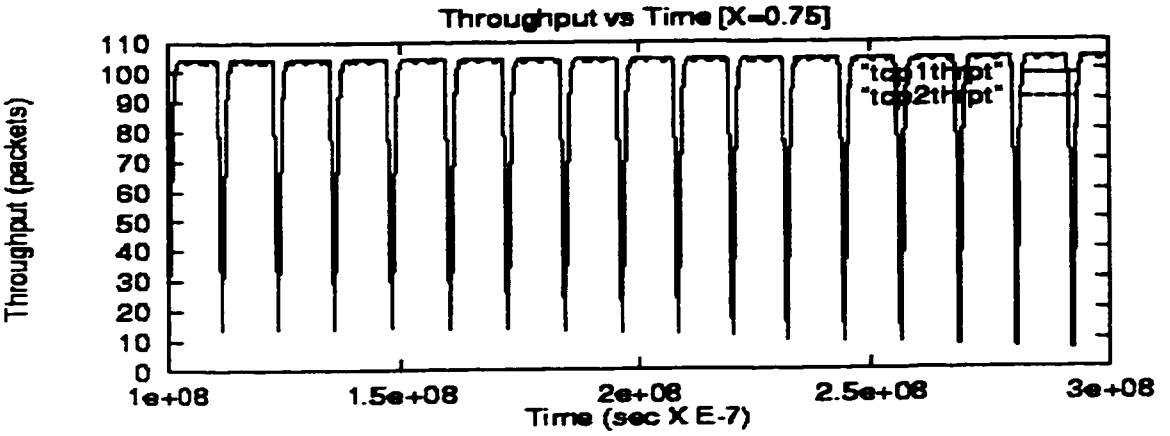
Graph 65: Exp4 – Congestion vs Time [X=2.0]



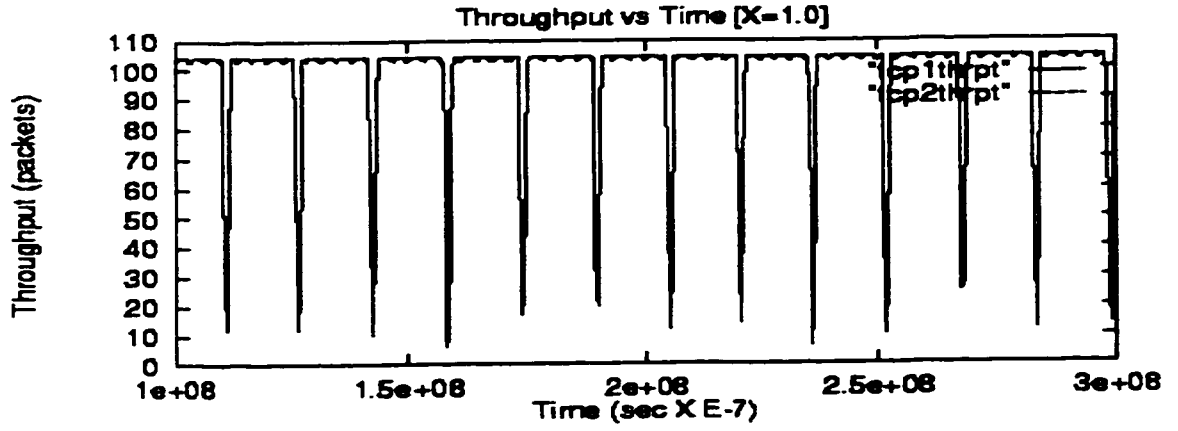
Graph 66: Exp4 – Throughput vs Time [X=0.1]



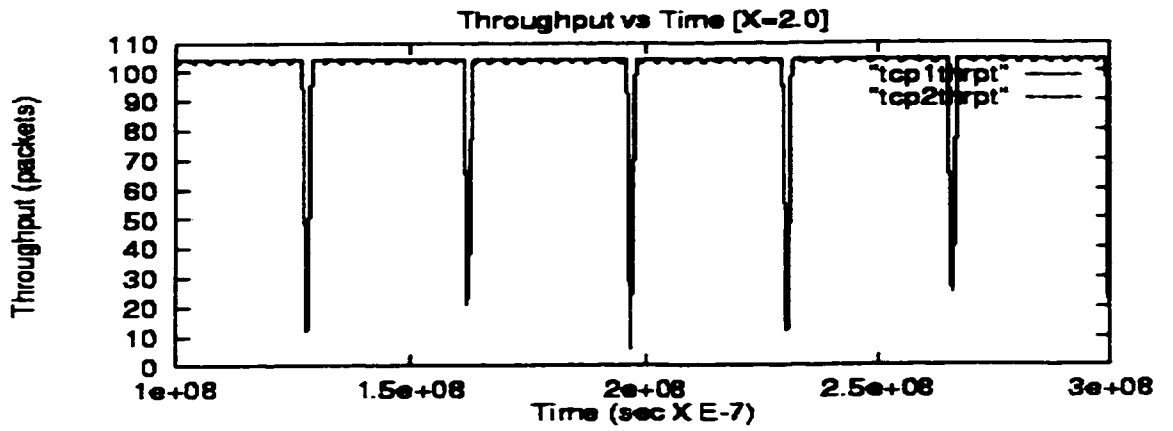
Graph 67: Exp4 – Throughput vs Time [X=0.1]; time slice = 10.5 sec to 12.5 sec



Graph 68: Exp4 – Throughput vs Time [X=0.75]



Graph 69: Exp4 – Throughput vs Time [X=1.0]



Graph 70: Exp4 – Throughput vs Time [X=2.0]

3.2.5 Experiment 5: 2 Connections with Different RTT & Finite Buffered Separate Routers

Experiment 4 measured TCP throughput for 2 connections with the same RTT values. Experiment 5 continues that scenario with sources with different RTT. The setup is identical to experiment 4, except that TCP source 1 has an RTT value of 12.4ms while TCP source 2 retained an RTT of 20ms. This simulation employs only ABR with the RENO flag off. The simulation will run for 40 seconds, as all other experiments.

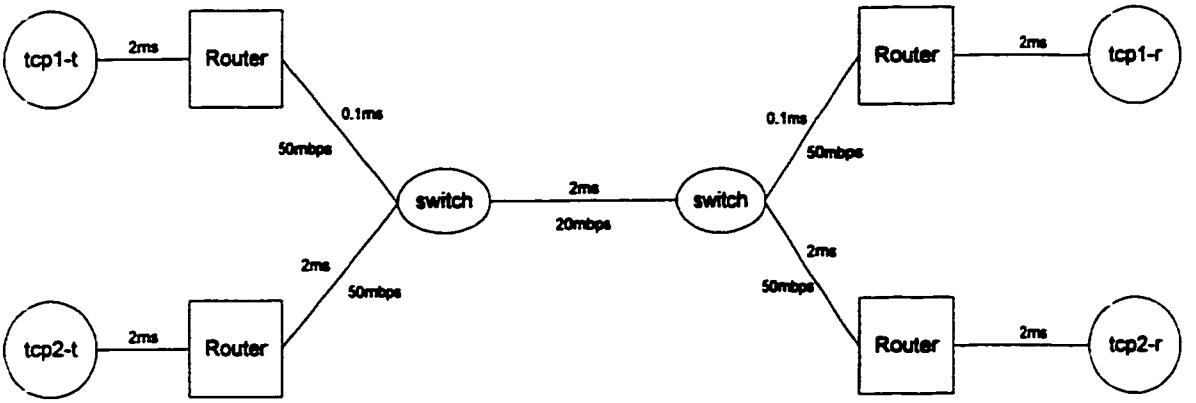


Figure 10: Experiment 5– 2 Connections with Different RTT and Separate Routers

Table 15 summarizes this scenario. TCP1 has a smaller RTT value than TCP2 and obtains a significantly higher throughput value only for the X=0.10 row. The rows show much smaller differences in throughput, as the router and buffer sizes are increased. The increased buffer sizes allow more packets to be sent, but also increase the number of lost packets. The fairness column indicates that ABR is able to allocate a fair share of the bandwidth to both sources, regardless of the difference in RTT.

ABR with Separate Routers

X	Router Size (bytes)	Buffer Size (bytes)	Throughput TCP1 (mbps)	Throughput TCP2 (mbps)	loss TCP1 (pkts)	loss TCP2 (pkts)	Pkts sent TCP1 (pkts)	Pkts sent TCP2 (pkts)	Fairness (%)	Efficiency (%)
0.10	2500	5000	4.61	1.81	1254	1452	45068	17673	0.84	32
0.75	18750	37500	7.78	7.58	1287	946	76042	74070	0.99	77
1.00	25000	50000	7.93	7.76	1364	1199	77555	75856	0.99	78
2.00	50000	100000	8.17	8.02	1452	1518	75869	78496	0.99	81

Table 15 : Experiment 5 – 2 Connections with different RTT and Separate Routers

Table 16 summarizes the CB ratios calculated for this experiment and for experiment 3 when UBR services were employed on sources having different RTT values. Buffering is being controlled at the router by ABR flow control mechanism and not by the switch buffer policy. Packets are dropped at the router and therefore the switch is able to place cells onto the link from each source in an equal manner, as shown by the fairness index column. In the UBR scenario, source cells flood the switch buffer before they can be dropped.

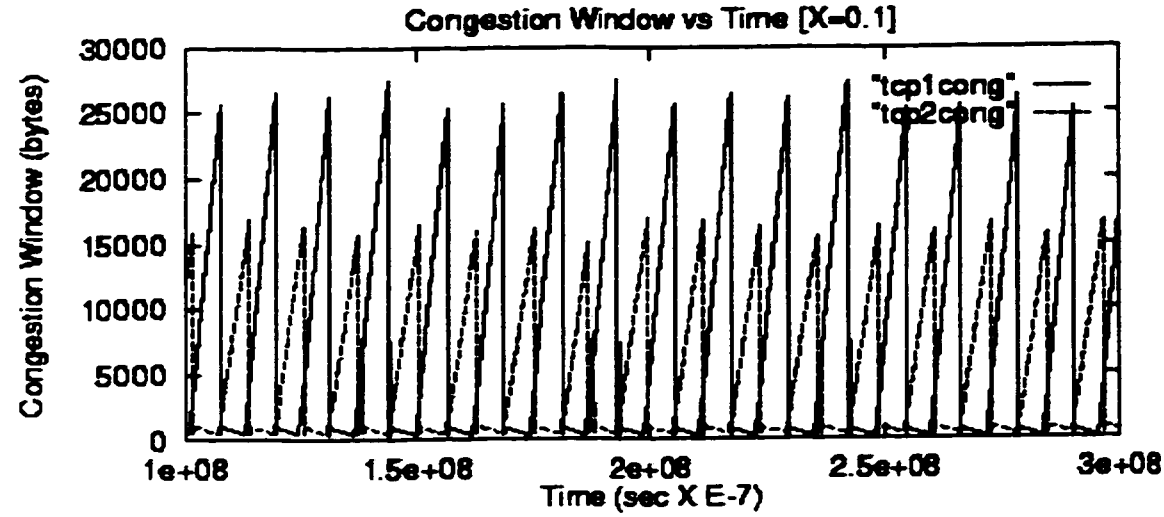
X	Packet Loss		Fairness Index		% Efficiency		CB Ratio	
	UBR	ABR	UBR	ABR	UBR	ABR	UBR	ABR
0.10	4059	2706	0.91	0.84	24	32	169.13	84.56
0.75	2145	2233	0.91	0.99	74	77	28.99	29.00
1.00	2629	2563	0.74	0.99	78	78	33.71	32.86
2.00	2376	2970	0.79	0.99	83	81	28.63	36.67

Table 16: CB Ratio comparison for 2 connections with different RTT

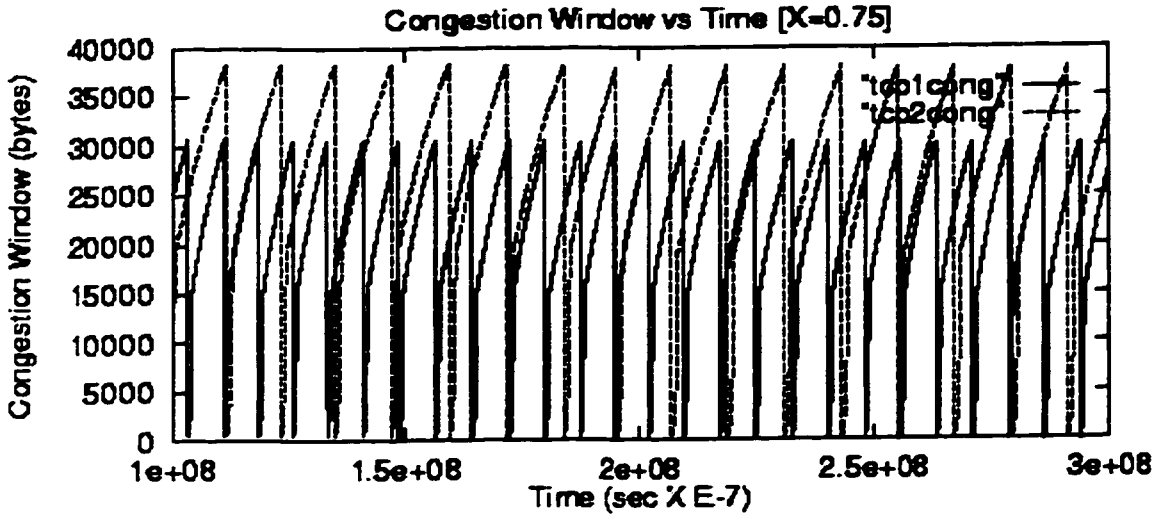
The lack of strong flow control in the UBR experimental results allows TCP1, which has a smaller RTT, to occupy more of the switch buffer. ABR prevents TCP1 from occupying more of the buffer, because of this stronger flow control employed at the routers. Congestion window vs Time Graph 71 to Graph 74 are compared to Graph 43 to Graph 46, of experiment 3. The difference between the peak values of TCP1 and the peak values TCP2, among the ABR plots. For example, the peaks of Graph 74 are much closer together than those of Graph 46. The ABR graph (X=2.0) shows a congestion window difference of 9 Kbytes between TCP1 and TCP2. The UBR graph (X=2.0) show a difference of 75 Kbytes. This again is because of the tighter flow control mechanisms of ABR that react to network congestion faster than UBR mechanisms.

We witness the same general trends in these graphs, as in previous experiments. As the X factor is increased, the peak congestion window increases as well. However, the number of times a Fast Retransmit is issued decreases. The fairness of ABR is further shown by the throughput traces of Graph 75 to Graph 78. The graphs X=0.75 to X=2.0 show how the throughput is shared. TCP2 is able to transmit for the same number of peak periods as TCP1, at similar throughput rates. In contrast, the equivalent UBR throughput Graph 47 to Graph 50 do not show this.

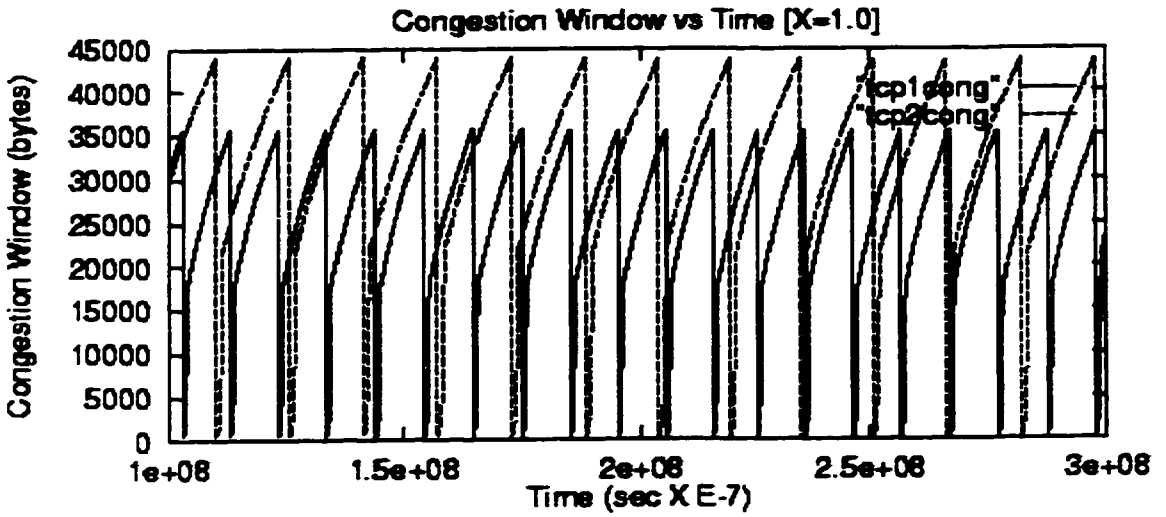
In this experiment, we saw that two sources having different RTT values were able to obtain near equal throughput when ABR services were used to control flow. In this case, each source traveled through a separate router before arriving at the bottleneck switch. We will next investigate the behavior of the sources if they are routed through a single device before reaching the switch. This will cause the shared router to buffer and manage two source paths.



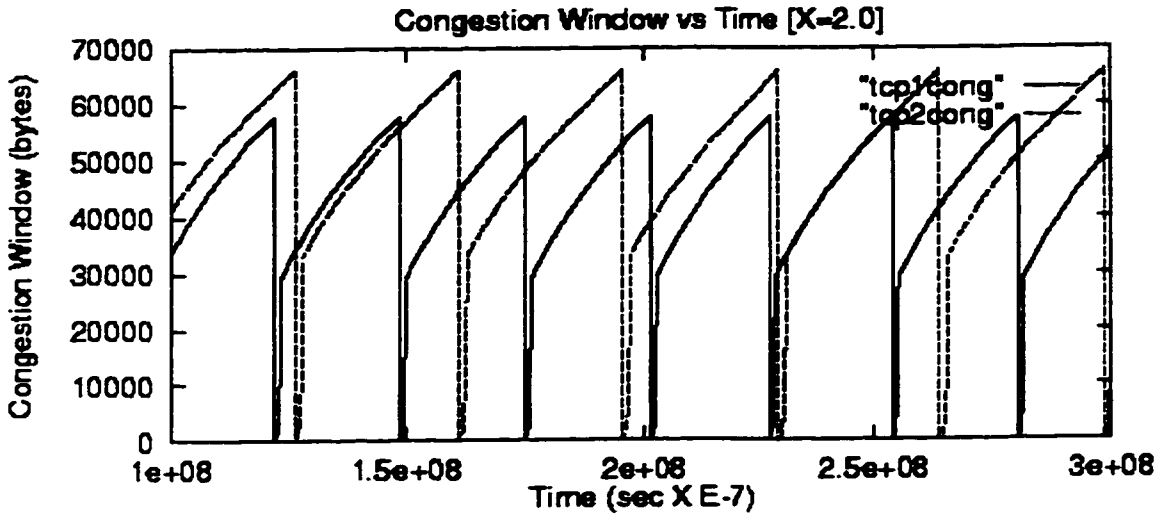
Graph 71: Exp5 – Congestion vs Time [X=0.1]



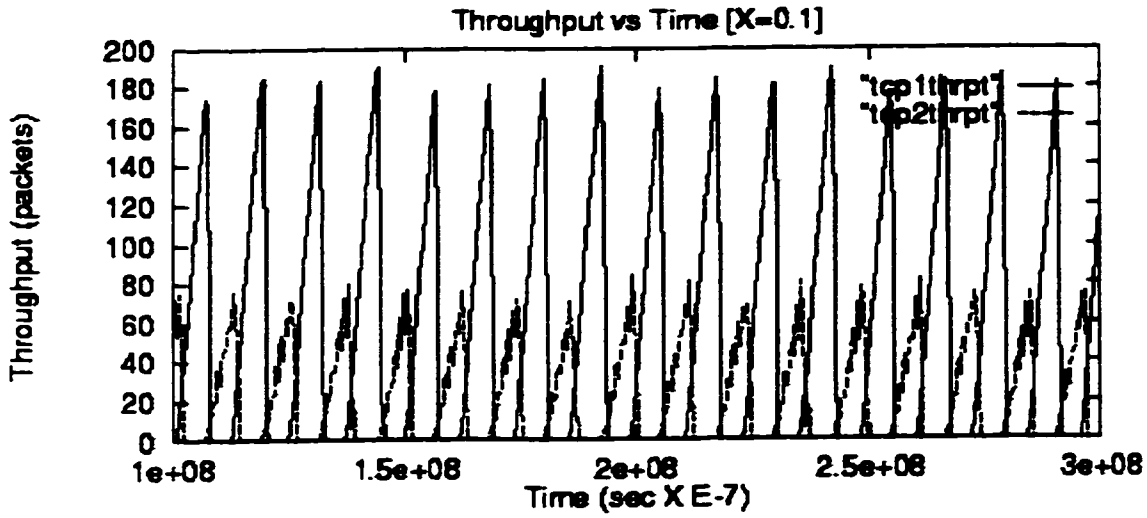
Graph 72: Exp5 – Congestion vs Time [X=0.75]



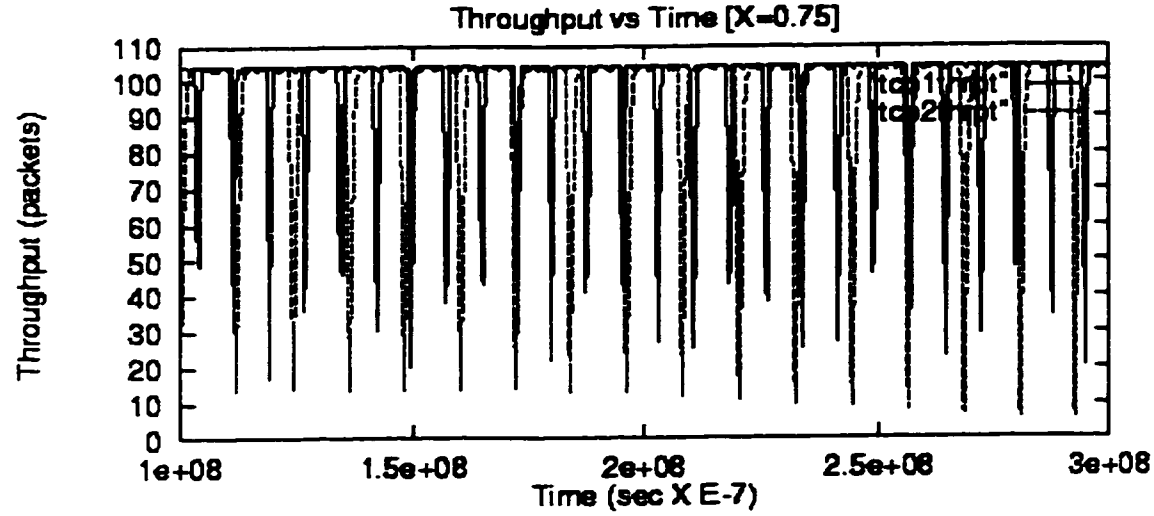
Graph 73: Exp5 – Congestion vs Time [X=1.0]



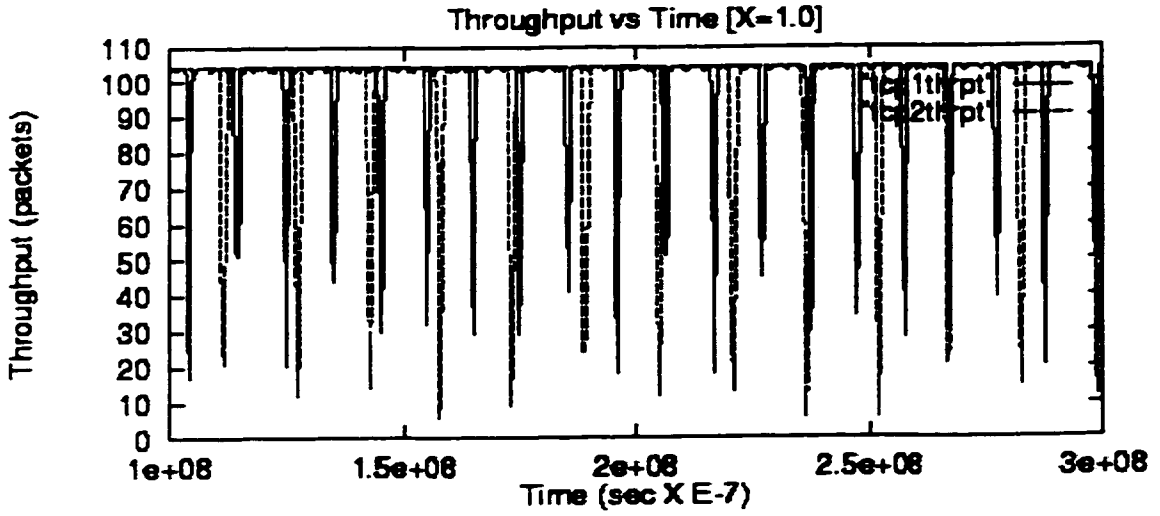
Graph 74: Exp5 – Congestion vs Time [X=2.0]



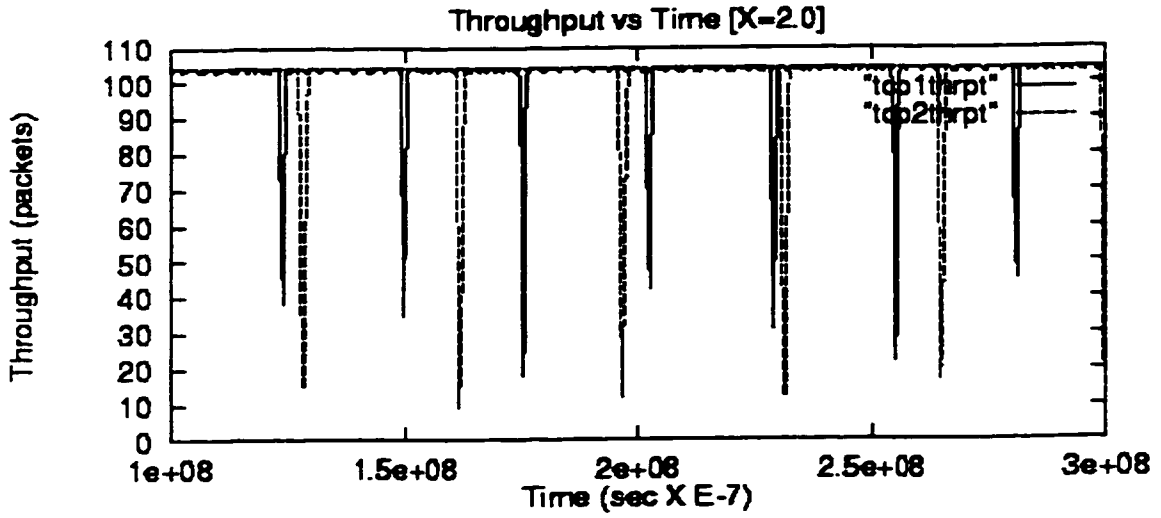
Graph 75: Exp5 – Throughput vs Time [X=0.1]



Graph 76: Exp5 – Throughput vs Time [X=0.75]



Graph 77: Exp5 – Throughput vs Time [X=1.0]



Graph 78: Exp5 – Throughput vs Time [X=2.0]

3.2.6 Experiment 6: 2 Connections with Finite Buffered Shared Router

Previously we looked at the case where each host was transmitting via its own unique router. Here we study the allocation fairness and efficiency when all hosts share a single finite buffered router. It becomes necessary for the router to take on the task of buffering and allocating bandwidth to all sources. We use ABR congestion

control mechanism in this experiment and will look at the TCP throughput and congestion window evolution over time. Again, we will not use the RENO flag for Fast Recovery.

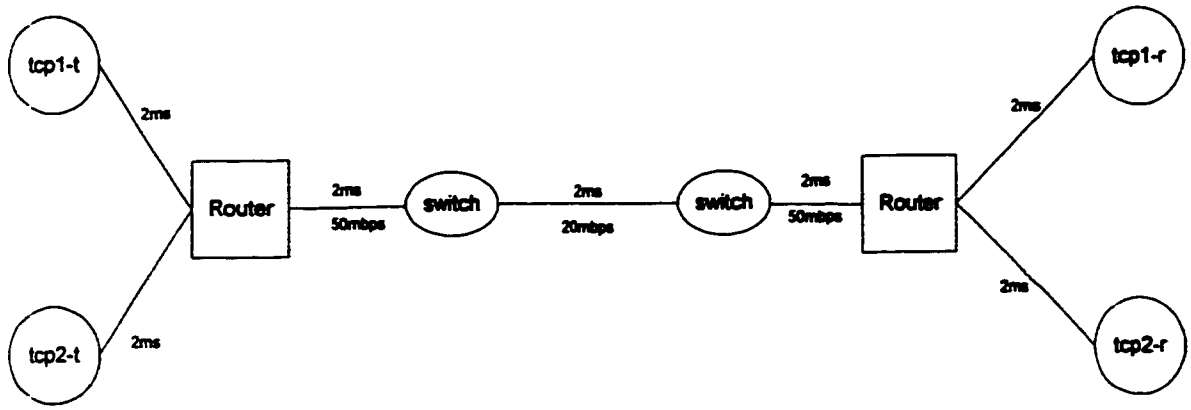


Figure 11 : Experiment 6 – 2 Connections with Single Router

ABR with Single Router

X	Router size (bytes)	Buffer Size (bytes)	Throughput TCP1 (mbps)	Throughput TCP2 (mbps)	loss TCP1 (pkts)	loss TCP2 (pkts)	Pkts sent TCP1 (pkts)	Pkts sent TCP2 (pkts)	fairness (%)	efficiency (%)
0.10	5000	5000	3.06	3.24	1463	1298	29896	31712	1.00	32
0.75	37500	37500	6.42	6.69	1573	1683	62689	65533	1.00	66
1.00	50000	50000	7.26	7.40	1705	2299	70997	72381	1.00	73
2.00	100000	100000	8.43	7.20	572	2783	82380	70473	0.99	78

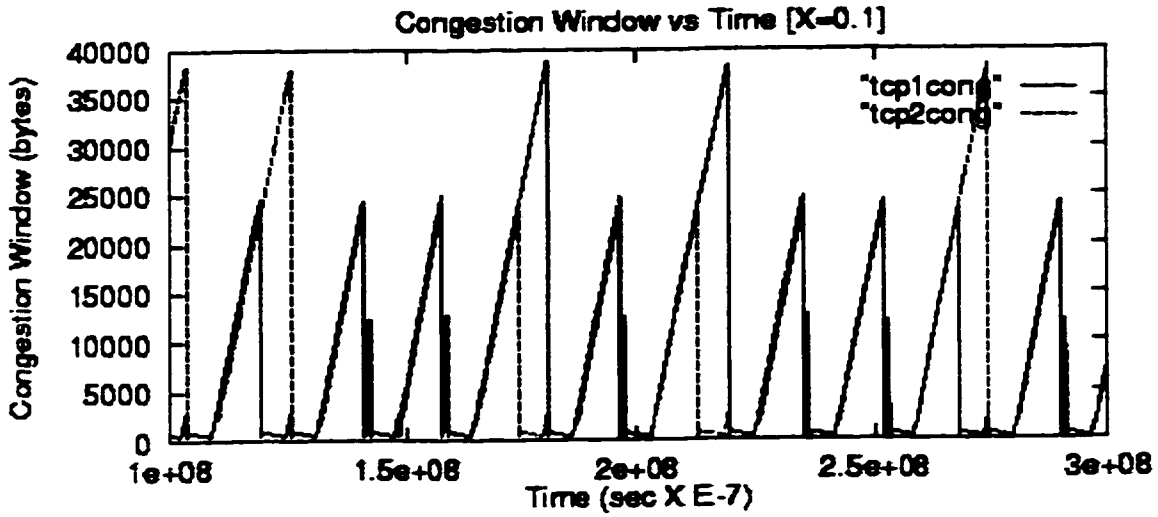
Table 17 : Experiment 6 - 2 Connections with Single Router

Table 17 summarizes the data collected from the simulations involving two connections sharing the same router for passing through the ATM network. Here both TCP1 and TCP2 have the same RTT and must share the same ABR path. The fairness column indicates near equal bandwidth allocation. We also see that the efficiency is close to the values obtained in Experiment 4. In that simulation, TCP1 and TCP2 have the same RTT but were buffered through separate routers. The efficiency column of Table 13 showed values of 30%, 76%, 70% and 80%, as compared to 32%, 66%, 73% and 78% of Table 17.

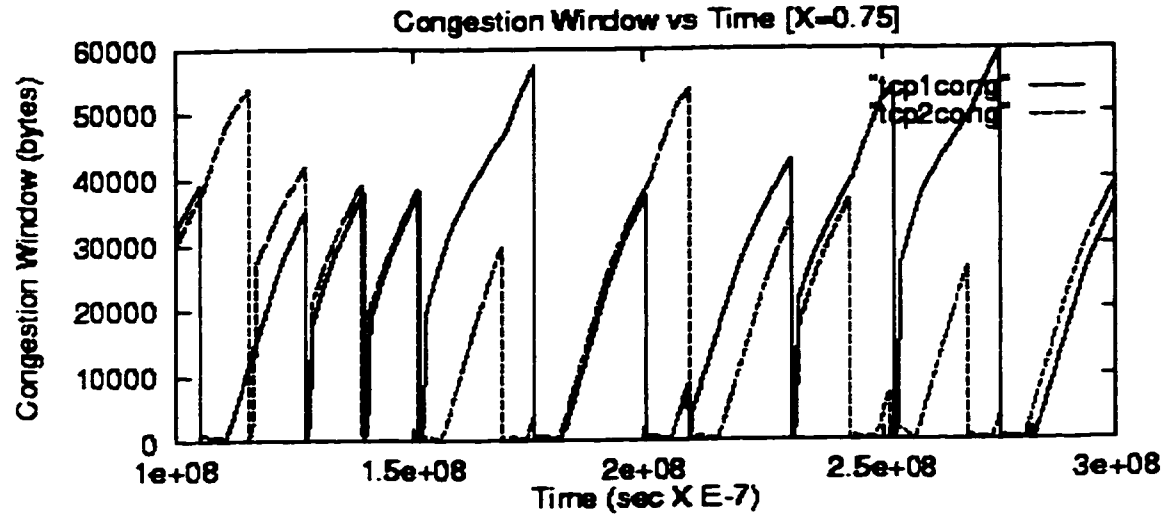
The congestion window Graph 79 to Graph 82 are compared to the equivalent graphs of Experiment 4 (Graph 61 to Graph 65). We see that when the sources are buffered through the same router, the traces are not superimposed. The graph X=0.1 shows TCP2 attains a higher peak throughput at times 10.5, 13 and 27 seconds. This is also indicated by the corresponding throughput Graph 83. Each throughput peak occurs at the same time as the congestion window peaks. Of course, this is expected for all experiments. At these particular points, TCP1 is only able to reach at maximum values of 2.5 Kbytes, but TCP2 reaches a value of 38 Kbytes. At times 17 and 22 seconds TCP1 reaches its peak values, but TCP2 only reaches 2.5 Kbytes. At most other peaks points, both sources share the same trace and attain a maximum congestion window of 24.5 Kbytes. This value is consistent with peak values obtained for graph X=0.1 in Experiment 4. The exception is at times 18 and 21.7 seconds, where TCP1 receives more of the bandwidth. This causes TCP2 packets to be dropped, because of the lack of available resources currently used by TCP1. Again, regular timeouts occur due to the small buffer sizes. Proceeding to the two graphs (X=0.75) on pages 65 and 78, we see that when the sources share the same path, timeouts are more frequent. TCP1 timeouts at 10.5, 17, 20 and 27 seconds. TCP2 timeouts at 15, 16.7, 21, 24.3, 25.2, 26.5 and 27 seconds. The graph on page 65 for Experiment 4 (X=0.75), shows no timeouts. The shared buffer is still not large enough to accommodate all the packets from both sources and therefore must drop packets from one or the other. This also accounts for the inconsistent throughput spikes shared by the two sources, as displayed on throughput Graph 84 (X=0.75). The source traces follow the same pattern at times 14 and 15 seconds and have identical throughputs of 107 packets. When one source timeouts, the other is able to increase its throughput, as shown by the peak points at times 10.5, 17, 20, 24.3 and 27 seconds. Graph 81 (X=1.0) shows a pattern more consistent with that of Graph 64 (X=1.0) of Experiment 4. The shared buffer graph displays that only one timeout occurs at time 16 seconds, by TCP2. At this point TCP1 is able to obtain a higher throughput, indicated by the large spike of the throughput Graph 85 (X=1.0). When the shared buffer is increased to a factor X=2.0, Graph 86 displays that TCP1 obtains more throughput than TCP2, which timeouts more frequently at times 12.7, 18, 20 &

28 seconds. The buffer in this case dropped more of TCP2's packets and allowed TCP1 to increase its congestion window, Graph 82. The TCP1 traces shows three cycles in which it changes from congestion avoidance to Fast Retransmit to slow-start and back to congestion avoidance. Leading up to time 20 seconds, TCP1 is in congestion avoidance and its *cwnd* = 109 Kbytes. After a Fast Retransmit is issued, it starts and continues in slow-start mode until it reaches *ssthresh* = 109/2 = 54.5 Kbytes. It then continues in congestion avoidance again, until the next Fast Retransmit.

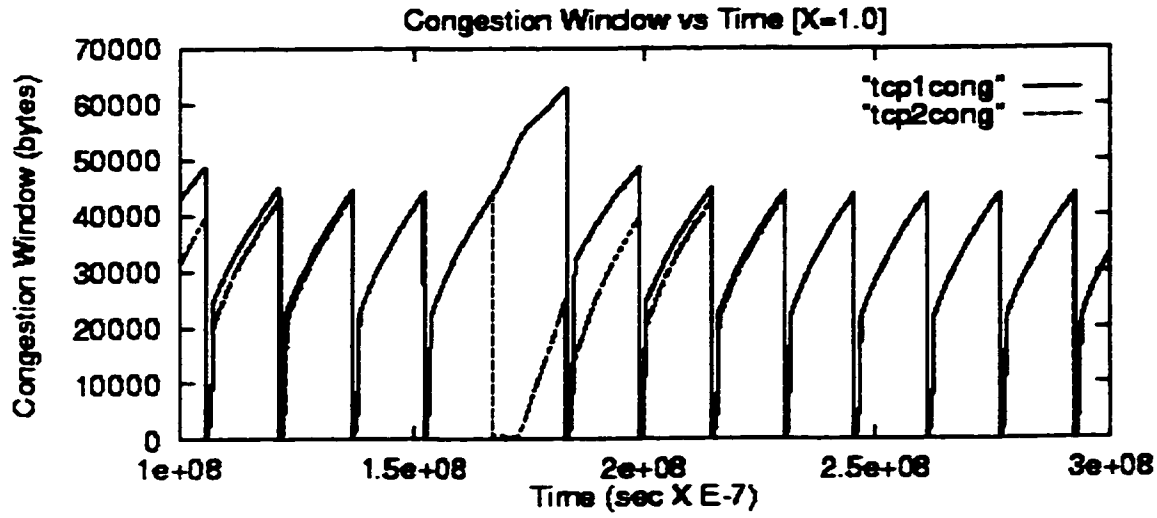
We have just seen the characteristics of the TCP traces when two sources having the same RTT are forced to pass over the same ABR path, by being buffered through a common router. We now would like to perform a similar analysis in comparing experiments involving sources with different RTT values. Experiment 5 explored the case, where two sources have different RTT, transmitted through separate routers. The scenario where two sources having different RTT are routed through the same device would be compared next. However, the limitations of the simulator and time constraints prevent us from performing this experiment. Normally, in order to create a smaller RTT we would decrease the delay on the router-to-switch link of the connection to be affected. However, in the case where all connections must be routed through the same device and hence travel over the same router-to-switch link, we cannot change the delay of the link without affecting all connections. In other words, the scenario would still be end up as multiple connections with the same RTT flowing through a single finite buffered router. We leave this experiment for another paper.



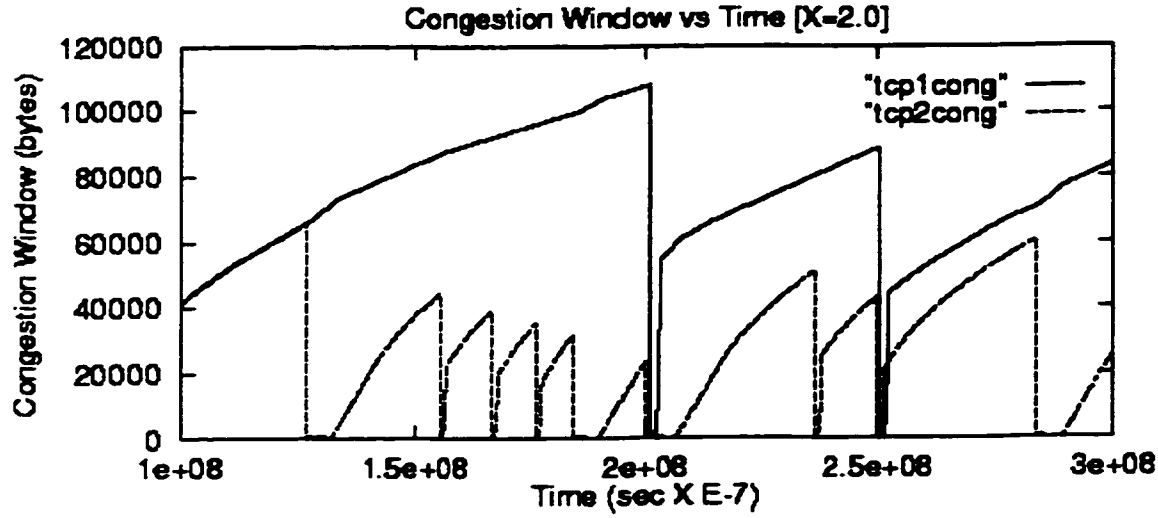
Graph 79: Exp6 – Congestion vs Time [X=0.1]



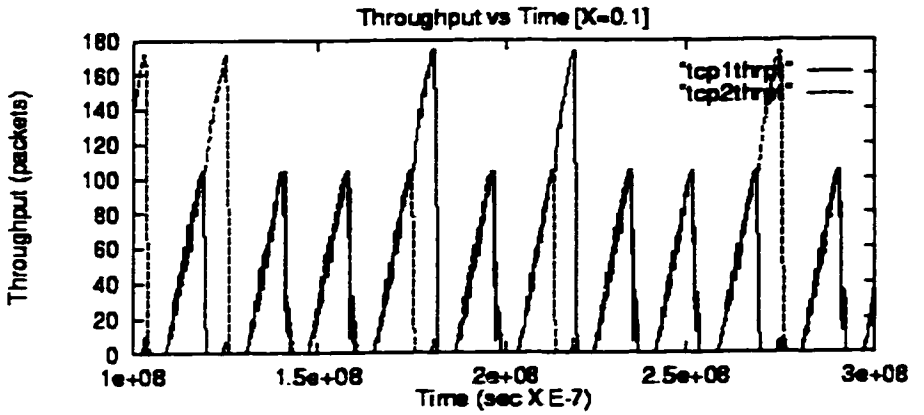
Graph 80: Exp6 – Congestion vs Time [X=0.75]



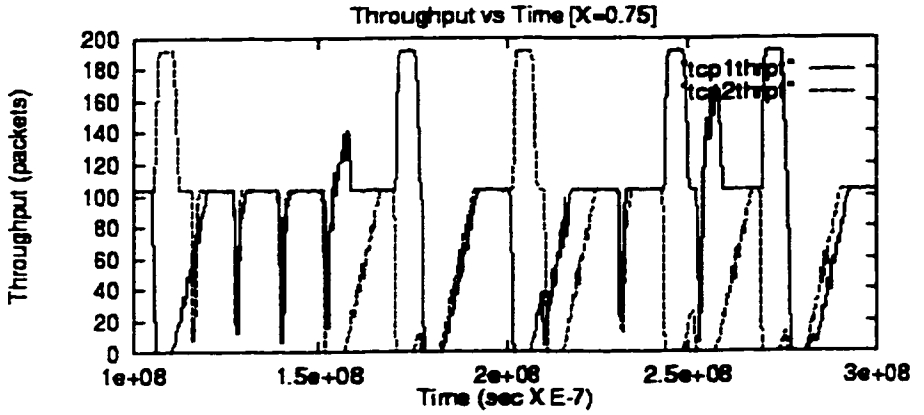
Graph 81: Exp6 – Congestion vs Time [X=1.0]



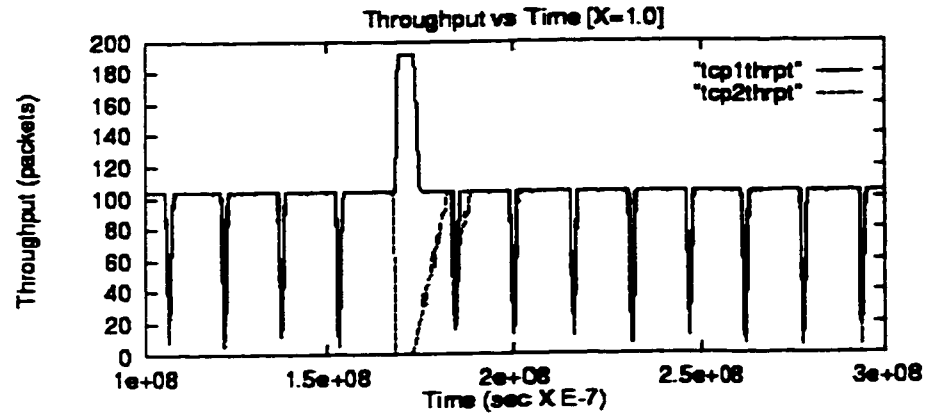
Graph 82: Exp6 – Congestion vs Time [X=2.0]



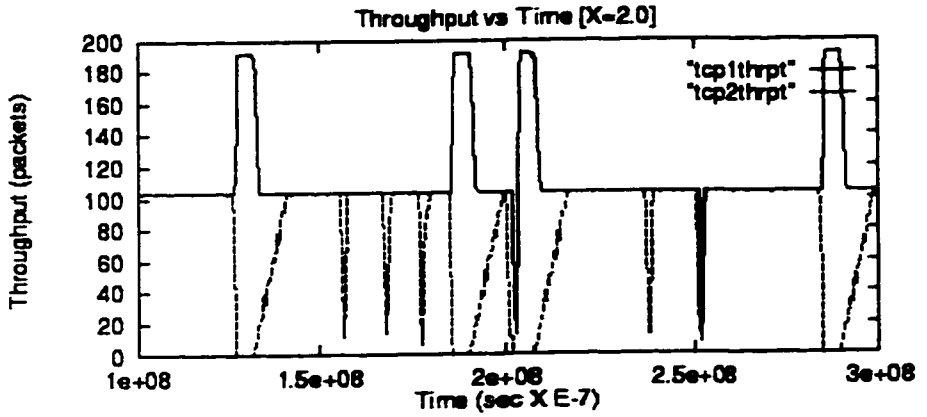
Graph 83: Exp6 – Throughput vs Time [X=0.1]



Graph 84: Exp6 – Throughput vs Time [X=0.75]



Graph 85: Exp6 – Throughput vs Time [X=1.0]



Graph 86: Exp6 – Throughput vs Time [X=2.0]

3.3 Maximum Throughput and Protocol Overhead

Results of the experiments show that a maximum TCP throughput of 87% is obtained in any situation. With the ABR service providing complete fairness and buffer overflow and the TCP RENO implementation providing Fast Recovery, there is still some lack of throughput.

This is due to the various protocol header and trailer overheads, dropping of cells when switch buffers overflow, congestion control and feedback Resource Management cells of the ABR service, plus TCP ACKs and retransmitted packets sent back to the transmitting source.

Each ATM cell is 53 bytes with 48 bytes going to payload and 5 bytes going to a header, giving a 10% overhead in each cell. There are two sources of overhead in AAL5. The 8-byte trailer at the end of each data unit and the fact that an AAL5 PDU needs to use a number of ATM cells to complete a maximum of 65 Kbytes. Encapsulation for IP over AAL5 needs a header of 8 bytes per datagram and a TCP header contributes approximately 20 bytes of overhead.

For UBR the maximum possible throughput is calculated as follows:
*(512 byte segment)/(11 cells per segment*53 bytes per cell)* $\approx$ *87%*

For ABR the maximum possible throughput is calculated as follows:
1 out of every 32 cells is a Resource Management (RM) cell, hence
(31/32)(512 byte segment/(11 cells per segment*53 bytes per cell))* $\approx$ *85%*

A study done by [14] shows that for OC-3 (155.52 mbps) and OC-12 (622 mbps) a range of 75% to 87% of the raw bandwidth is actually used.

4 Conclusion

The preceding experiments showed that ABR provided good throughput and high fairness for sources with and without difference in round-trip times. A thorough analysis of the congestion window showed its evolution over time as the buffer was increased. In the case of the UBR experiments, the switch was the focal point for bandwidth allocation. In the ABR experiment, feedback flow control mechanism at the routers dictated the allocation of network resources.

The proceeding section will summarize the findings of the experiments performed and cite some scenarios for future work. The subsequent sections will then present a global perspective of current uses of ATM and how it may evolve in the future.

4.1 Summary and Future Work

We compared the effectiveness of both service classes in providing fairness and efficiency. When sources had identical round trip times, both services yielded similar CB ratios and complete fairness. When sources with different RTT values were simulated, both services again displayed similar CB ratios, however the ABR results showed that it was able to provide better fairness than UBR. The use of Fast Retransmit and Recovery increased the overall throughput, but did not increase the fairness index. Detailed time slices of the congestion window showed how the *ssthresh* and *cwnd* values were set as the TCP source changed congestion control mechanisms. The Fast Retransmit triggered the retransmission of a lost packet before the normal timeout period and Fast Recovery used the existing ACKs in the pipe to clock the sending of packets. This effectively kept *cwnd* from restarting at one segment and the TCP source out of slow start. We noticed that all traces, both congestion window and throughput graphs, for all experiments exhibited cyclical behavior. The graphs and hence the tabulated data reflect the points occurring during this steady state period. The nature of the TCP congestion control mechanism and how they are activated and the traffic model used in all the simulations contribute to this recurring pattern. We witnessed in experiment 1 the emerging pattern as one source transmitted packets to one receiver. The subsequent experiments had a basic

pattern but had different peaks and minimum values only because different loads were introduced into the network. However, the hosts were equipped with the same TCP mechanisms for congestion control and the reaction mannerisms were identical, only the values varied. The rate at which packets were introduced into the systems also affects the patterns observed. In all of the experiments, we employed an infinite constant stream of packets. Therefore, the transmission rate remained constant. Once the sender-receiver pair(s) discovered the maximum capacity of the network, steady state was reached and was maintained because the transmission rate remains constant. The manner in which congestion was discovered and relayed back to the sending host was changed when ABR mechanism were activated and hence a slightly different pattern was observed but the cyclical behavior remained.

The extra complexity in the routers for the ABR based sub-network was useful only for providing better fairness in the case where sources had different RTT values. Otherwise, the much cost effective UBR based sub-network is sufficient in providing very good throughput. The CB ratios showed that both types of services drop approximately the same amount of packets while delivering similar efficiency. These conclusions are valid only for scenarios in which an EPD policy is implemented. It remains to be seen what outcome results if other buffering policies like Fair Buffer Allocation and Random Early Detect are used. This study can also be enhanced by carrying out experiments on additional network configurations like the "parking-lot" and other benchmark configurations established by the ATM Forum and also testing the fairness index on more than 2 sources. Another set of interesting experiment would be to use a dynamic TCP-to-VC channel mapping. The two extremes of the allocation would be one VC–per-TCP source and one VC for all TCP sources. An aggregate model would test the best allocation mapping for the combination of high throughput, high fairness and low loss. These simulations can be run for both UBR and ABR with sources having identical and different RTT.

4.2 Current ATM Competitors

ATM is not always the technology of choice. There are other alternatives that sometimes provide a much more suitable solution to the problem, however this is determined by the service requirements and need of the users.

In the LAN workspace contention based protocols like Ethernet, Fast Ethernet and Gigabit Ethernet provide a head on competition to ATM. Most pre-ATM and LAN infrastructures are made up of some combination of Ethernet, Token-Ring or FDDI. More modern LANs or those being upgraded seem to be evolving to a switched backbone, but not completely over an ATM fabric. There are many mixed ATM/Gigabit Ethernet or Gigabit Ethernet only environments. From the perspective of dedicated bandwidth and non-shared traffic, Gigabit Ethernet is a wise choice. It provides the throughput desired at a lower cost than to deploy ATM in the LAN. However, most applications and services today require a much higher grade of efficiency, guarantee and reliability. Even with the entire debate as to whether to go with ATM or Gigabit Ethernet, ATM still provides a viable solution in the LAN environment. ATM was designed to handle multimedia and has the capability to establish connection-oriented virtual circuits that guarantee bandwidth and quality. Ethernet frames are sent on some shared media in a connectionless broadcast manner, where user contention management for the shared bandwidth is dependent on its collision detection mechanism. For time-sensitive and loss-less services, ATM is the definite choice. Ethernet has no quality of service, on its own, and is distance limited where as ATM provides end-to-end quality of service and has no distance limitations.

Over the WAN, ATM competes with private lines, frame relay and other frame based technologies. Single dedicated physical point-to-point channels are far too rigid for bandwidth choices when compared to ATM offering each point multiple logical connections to a number of endpoints. More over, fine bandwidth granularity and multiple service categories across logical virtual path and channel connections are provided by ATM. High-speed Frame Relay in some respects can be an equal to ATM, for example throughput (scales up to 45mbps duplex) when only cost-effective WAN data transport is required. The prioritization of traffic capability in high-speed Frame Relay introduces some minor bandwidth guarantee, but not to the extent that

ATM can provide. In addition, Frame Relay-to-ATM service internetworking is possible. Other frame based data protocols like Switched Multi-Megabit Data Services (SMDS) may have a better protocol-to-packet size efficiency, but again they do not provide the Quality of Service capable with ATM. The choice is application driven.

Quality of Service (QoS) is important for effective support of multimedia applications, combining telephony, voice and data over a WAN and for consolidating voice and data infrastructures. ATM along with switched Ethernet using RSVP and Cells in Frames (CIF) provide varying degrees of QoS. RSVP is supported by major router vendors in a proprietary way and by industry standard APIs. In both cases, shaping and policing functions must be implemented to control bandwidth usage to ensure fair QoS allocation. The main drawback is that as the number of flow requests increase, so do the routing tables. ATM, on the other hand, can control and enforce traffic contracts starting at the edge of the network, due to its underlying connection-oriented network paradigm. CIF is a software-based solution that places ATM cells into Ethernet frames and then forwards it to a separate CIF device that interfaces to the ATM network. This actually helps to quicken the reach of ATM to the desktop for supporting real-time traffic. So it would seem that CIF can bridge the gap between legacy and ATM end systems, until that time when the price per port of ATM devices in the LAN environment decreases to a cost that is considered competitive to Gigabit Ethernet.

4.3 ATM Technology in the Future

The outlook for ATM in the LAN for the next few years has switched Ethernet and Gigabit Ethernet coming on top. This is because of the industry forecast for increased sales in the hub and switch market, equipment availability and pricing, and because of the requirements for multimedia not dominating the LAN. However, the pricing gap is rapidly narrowing and major network equipment vendors who are selling Ethernet products are providing ATM solutions as well. The mix of voice, video and data to the desktop has not evolved fast enough, but when it does users of gigabit Ethernet will have to rely heavily on pre-standard quality of service offerings

like Resource Reservation Protocol (RSVP) to assist IP's shortcomings. Perhaps IPv6 with its added Priority and Flow Label fields will allow for a better implementation of the QoS protocol. However, with the continued development of CIF, ATM to the desktop may be coming quicker, while allow existing (Ethernet) hardware to remain.

The steadily increasing amount of multimedia traffic that is traversing over the Internet is leading Internet Service Providers to use ATM and it is foreseen that ABR service and RSVP will compete over the great public network as well. Continued increase for multimedia type traffic will fuel the need for a higher degree of reliability, guarantee and efficiency, which is a strong feature of ATM. However, these multimedia applications can only be created over a network that supports multiple QoS classes. So there exists a co-dependence between the two. In addition, local workstation software and hardware must provide cost-effective support.

The key component to ATM's success in the future, may be due to the need of Switched Virtual Circuits (SVC) by some key protocols, like Classical IP over ATM, LAN Emulation (LANE), Multiprotocol over ATM (MPOA) and IP multicast over ATM. SVCs provide a bridge between the IP and LAN connection-less packet-by-packet processing and ATM's connection oriented protocol that switches only once for an entire stream of packets. ATM's SVC-based support for higher layer protocols, however, is efficient only for long-lived traffic flows. In addition to signaling similar to that needed for telephone call establishment, ATM must also signal messages conveying requests for different service categories, traffic parameters, information about higher-layer protocols and topology information. All this makes for a very long call-set-up time and does not lead to a cost-effective solution for short-lived traffic flows. The technology curves for computing and storage should allow ATM switching to achieve high call setup rates and low call setup times cost-effectively, since ATM signaling is CPU and RAM intensive. [21]

Dynamic virtual networks with ubiquitous, secure and authenticated access is the model of coming LAN, WAN and internetworks. ATM has the capability to support such a model with its flexibility, internetworking ability with LANs and Internet protocols. Most network service providers have already deployed an ATM switched fabric and technological advances in ATM encryption and authentication

protocols (IPSEC, x.509) enhance the move of corporations utilizing Virtual Private Networks (VPN) for their everyday needs. The increasing number of VPNs deployed over public networks requires the need for intelligent switching and routing. ATM can provide this. ATM's future depends on its ability to succeed in several key areas such as support of multimedia (voice, video, data), integrated voice and data over the WAN through a single infrastructure, lower cost customer equipment, development of QoS aware APIs, internetworking with IP and RSVP and the collective co-operation of vendors and committees to develop and follow standards.

Bibliography

[1] D. Minoli, M. Vitella, *ATM and Cell Relay Service for Corporate environments*, McGraw-Hill, 1994.

[2] L.L. Peterson, B.S. Davie, *Computer Networks: A systems Approach*, Morgan Kauffman Series in Networking, 1996.

[3] R. Jain, *Congestion Control and Traffic Management in ATM Networks: Recent Advances and A Survey*, Department of Computer and Information Science, The Ohio State University, October 15, 1995.

[4] A. Romanow, *TCP over ATM: Some Performance Results*, ATM Forum/93-784, Traffic Management Sub-Working Group, July 29, 1993.

[5] K-Y. Siu, R. Jain, *A Brief Overview of ATM: Protocol Layers , LAN Emulation, and Traffic Management*, Department of Electrical Engineering, University of California, Department of Computer and Information Systems, The Ohio State University.

[6] H. Li, K-Y. Siu, H-Y. T, C. Ikeda, H. Suzuki, *TCP Performance over ABR and UBR Services in ATM*, Department of electrical Engineering, University of California, 1995.

[7] A. Romanow, S. Floyd, *Dynamics of TCP Traffic over ATM Networks*, IEEE Journal on Selected Areas In Communication, May, 1995.

[8] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, F. Lu, S. Srindhi, *Performance of TCP/IP over ABR*, Department of Computer and Information Science, The Ohio State University,

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

[10] R. Goyal, R. Jain, S. Kalyanaraman, S. Fahmy, *Performance of TCP over UBR+*, ATM Forum/96-1269, October, 1996.

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

[12] N. Yin, *End-to-End Traffic Management in IP/ATM Internetworks*, ATM Forum/96-1406, October 1996.

[13] H.Y. Gong, *ATM Congestion Control*, Recent Advances in Networking, Summer Quarter, 1996.

[14] J.D. Cavanaugh, *Protocol Overhead in IP/ATM Networks*, Minnesota Supercomputer Center, Inc., 1994.

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

[16] M. Laubach, *Classical IP and ARP over ATM*, RFC 1577 , 1994.

[17] R.J. Atkinson, *Default IP MTU for use over ATM AAL*, RFC 1626, May, 1994.

[18] R. Cole, D. Shur, C. Villamizar, *IP over ATM: A Framework Document*, RFC 1932, April, 1996

[19] *Traffic Management Specification*, Version 4.0, The ATM Forum Technical Committee, af-tm-0056.00, April ,1996.

[20] *User-Network Interface Specification*, Version 3.1, Prentice Hall, 1996.

[21] D.E. McDysan, D.L. Spohn, *Hands-On ATM*, McGraw Hill Series on Computer Communications, 1998.