COMBINED USE OF CONGESTION CONTROL AND FRAME DISCARDING FOR INTERNET VIDEO STREAMING

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Ongun Yücesan

January 2003

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Nail Akar(Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Gözde Bozdağı Akar

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Ezhan Karaşan

Approved for the Institute of Engineering and Sciences:

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

COMBINED USE OF CONGESTION CONTROL AND FRAME DISCARDING FOR INTERNET VIDEO STREAMING

Ongun Yücesan
M.S. in Electrical and Electronics Engineering
Supervisor: Assist. Prof. Dr. Nail Akar
January 2003

Increasing demand for video applications over the Internet and the inherent uncooperative behavior of the User Datagram Protocol (UDP) used currently as the transport protocol of choice for video networking applications, is known to be leading to congestion collapse of the Internet. The congestion collapse can be prevented by using mechanisms in networks that penalize uncooperative flows like UDP or employing end-to-end congestion control. Since today's vision for the Internet architecture is based on moving the complexity towards the edges of the networks, employing end-to-end congestion control for video applications has recently been a hot area of research. One alternative is to use a Transmission Control Protocol (TCP)-friendly end-to-end congestion control scheme. Such schemes, similar to TCP, probe the network for estimating the bandwidth available to the session they belong to. The average bandwidth available to a session using a TCP-friendly congestion control scheme has to be the same as that of a session using TCP. Some TCP-friendly congestion control schemes are highly responsive as TCP itself leading to undesired oscillations in the estimated bandwidth and thus fluctuating quality. Slowly responsive TCP-friendly congestion control schemes to prevent this type of behavior have recently been proposed in the literature. The main goal of this thesis is to develop an architecture for video streaming in IP networks using slowly responding TCP-friendly end-to-end congestion control. In particular, we use Binomial Congestion Control (BCC). In this architecture, the video streaming device intelligently discards some of the video packets of lesser priority before injecting them in the network in order to match the incoming video rate to the estimated bandwidth using BCC and to ensure a high throughput for those video packets with higher priority. We

demonstrate the efficacy of this architecture using simulations in a variety of scenarios.

Keywords: Congestion Control, Transmission Control Protocol, TCP-friendly congestion control, video streaming, temporal scalability

ÖZET

VIDEO AKTARIMI İÇİN BİRLEŞİK YOĞUNLUK DENETLEYİCİ VE AKTARIM HIZI ŞEKILLENDİRİCİ

Ongun Yücesan
Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Assist. Prof. Dr. Nail Akar
January 2003

Internet üzerinde gün geçtikçe artan video uygulamaları vede bu uygulamalarda tercih edilen Kullanıcı veri protokolu (UDP) nin yogunluk denetim mekanizmalarından yoksun olması nedeni ile ağişlerin yogunlukları giderek artmaktadır. Bu artışın ağların yoğunluk nedeni ile çoküşüne neden olabildiği gözlenmiştir. Bu durumun engellenebilmesi için ağiş içerisinde bazı tedbirler alınabilineceği gibi yoğunluk denetim mekanizmaları kullanılarakta çözüm getirilinebilinmektedir. Günümüzde hakim olan genel yaklaşım ağlar üzerindeki akıllı işlevlerin daha çok ağlarin ucuna dogru taşinmasina yonelik olması nedeni ile yoğunluk denetim mekanizmaları uzerine yogun bir sekilde araştırma yapılmaktadır. Bir çözüm halen veri aktarımı için kullanılmakta olan Aktarım Kontrol Protokolu (TCP) kullanımı olarak onerilmektedir. Bir başka yaklaşım ise TCP dostu algoritmaların kullanımı olarak öne çıkmaktadır. Bu algoritmaların bazıları aynı TCP'nin kendisinin de olduğu gibi çok değişken aktarım hızları sağlamakta ve dolayısı ile izleyici açısından rahatsız edici değişken bir kaliteye neden olmatadırlar. Yavaş tepki gösteren denetim mekanizmaları bu konuda uzun zaman aralıklarında geçerli olan değişikliklere tepki gostererek daha sabit bir kalite seviyesini sağlamaya çalışmaktadırlar. Biz tez kapsamında gene böyle bir mekanizma olan Binomsal Yoğunluk Denetim (BCC) mekanizmalarını kullanmaktayız. Dinamik olarak belirlenen veri aktarım hızına video hızını uydurabilmek açısından bir hız şekillendirici mekanizma kullanılmaktadır. Bu mekanizma mevcut aktarım hızının daha onemli video parçaları tarafından kullanılmasına yonelik olarak ayrım yapmaktadır. Bu sitemin etkinliğini simulasyon yolu ile değişik senaryoların altında değerlendirmekteyiz. *Anahtar Kelimeler:* agiş, yoğunluk denetim, TCP, UDP, hat hızı, video hızı

ACKNOWLEDGMENTS

I would like to express my deep gratitude to my supervisor Assist. Prof. Dr. Nail Akar for his guidance, suggestions and invaluable encouragement throughout the development of this thesis.

I would like to thank Assoc. Prof. Dr. Gözde Bozdağı and Assist. Prof. Dr. Ezhan Karaşan for reading and commenting on this thesis.

Contents

List of Figures

List of Tables

To My Family and Friends ...

Chapter 1

INTRODUCTION

The Internet comprises a network of computer networks, which transmit messages to one another using a common set of communications protocols, or sets of operating rules. Networks comprise addressable devices or nodes (computers) connected by communication channels. Nodes are not limited to performing a single role; for example, some workstations may also be configured to act as servers for other workstations, and even as routers. For each of the roles that a particular node performs, it is assigned a unique identifier, called an IP-address. Any node can transmit a message to any other node, along the communications channels, via the intermediate nodes.

The term protocol is used to refer to the set of rules that govern the communications between nodes. A number of functions need to be performed, and hence there is a considerable number of involved protocols. The complete family of protocols is referred to as the Internet Protocol Suite. Sometimes the family is also referred to by the combined names of just the two most important protocols, TCP/IP (Transmission Control Protocol/IP Protocol).

To simplify matters, the functions are organised into a series of layers., the lowest layer being the link layer which specifies how the node interfaces with the communications channel. Link layer protocols convert the bits that make up packets into signals on channels. One layer above lies the network layer protocols which specify how packets are moved around the network. This includes the important questions of how to address the node that is being sought, and how to route each packet to that node. The key protocol at this level is IP (Internet Protocol). Other protocols at this level, which are closely related to and dependent on IP, include:

- ICMP (Internet Control Message Protocol), which is used to report errors and obtain information about the transmission of IP datagrams; and
- IGMP (Internet Group Management Protocol), which is mostly used in multicasting (transmitting a single message intended for multiple recipients).

The transport layer protocols specify whether and how the receipt of complete and accurate messages is to be guaranteed. In addition, if the message is too large to be transmitted all at once, it specifies how the message is to be broken down into segments. There are two major transport layer protocols:

- TCP (Transmission Control Protocol), which is the key protocol at this level, and provides a reliable message-transmission service;
- UDP (User Datagram Protocol), which provides a stateless, unreliable/best effort service.

The application layer protocols handle messages that are to be interchanged with other applications in nodes elsewhere on the Internet. They specify such details as the sequence and format of the data-items.

Of particular importance to the current thesis in this layered architecture is Transmission Control Protocol (TCP). An important property of TCP is, different flows under similar conditions get roughly the same bandwidth. Therefore, competing flows get the fair share of the bandwidth. However, TCP probes for available bandwidth, and halves its rate aggressively in response to congestion. While data communications can tolerate such bandwidth variations, unicast video and audio applications perform better if they are streamed with congestion control mechanisms that react slowly. The User Datagram Protocol (UDP) does not have mechanisms that will adapt its packet sending rate according to the network conditions. UDP also does not provide a guarantee that the packet will be delivered. It simply sends the data at the rate it has been instructed to. This unresponsive behavior of UDP may result in both unfairness among the competing flows, and congestion collapse of the Internet [2].

Since UDP and TCP are not very suitable for multimedia applications, the TCP friendliness concept has been raised. Such a TCP-friendly algorithm will therefore have roughly the same throughput with a TCP connection under similar long term conditions. This algorithm while interacting fairly with TCP, will

adapt its rate smoothly so that video applications using it, can benefit. TCP-friendly algorithms that are proposed are TCP-Friendly Rate Control (TFRC) [3], Binomial Congestion Control (BCC) [4], Rate Adaptation Protocol (RAP) [5], and others.

These techniques provide the necessary responsiveness for the healthy operation of the Internet. Since the available bandwidth changes over time, even if it is smoothly varying, there is a need to adapt the video rate requirements to this dynamically changing value. Quality adaptation schemes accomplish this task. The main types of quality adaptation mechanisms or filters are frequency filters, layer dropping filters, frame dropping filters, and codec filters. Frequency filter works on compression layer and may discard some of the high frequency components or some color information. Layer dropping filter is the mechanism that includes or discards the necessary amount of layers of a scalable coded video. Frame dropping filter discards necessary amount of frames according to their importance. It matches the rate of the video through adapting the frame rate of the video. A codec filter decodes and re-encodes the video.

Layer dropping filters work with the scalable encoded videos. The best known scalability methods for encoding of the videos are the spatial, temporal, and SNR scalabilities. The spatial scalability is the scalability of frame sizes. The SNR scalability is the scalability of the frame quality. For this scalability the video is encoded with various quantizer step sizes. Temporal scalability is scalability of frame rates. The video therefore can be viewed and streamed in various frame rates. The scalable videos consist of layers. The minimum sized, least frame rated, or lowest quality version of the video is named as the base layer. Over the base layer, enhancement layers are added and higher quality, higher frame rate, larger sized versions of the video is obtained. Each added layer enhances the plausibility of the content.

Examples of the techniques developed for layered quality adaptations are systems developed by Rejaie et.al. [6] employing RAP congestion control, the implementation of the same system for BCC by Feamster et. al. [7], and the system that employs a very advanced scalability technique (Fine Granular Scalability) FGS by Liu et. al. [8]. The quality adaptation mechanism developed by Reajie et. al. [6] adds and drops layers from a discrete set of layers to perform long term coarse grain adaptation, while using RAP to react to congestion on very short scales. The mismatches between two timescales are absorbed by buffering at the receiver. The system by Liu et. al. employs fine granular scalability. For a FGS

coded video, there are two layers. First layer is the base layer, second layer is the enhancement layer, which is coded by a bit-plane coding technique. Therefore, the adaptability of enhancement layer becomes very fine granular. There may be very little mismatches but they are compensated on the long run.

The layered video used in the work by Rejaie et. al. is not an actual video trace. Also FGS is not a widely deployed and well understood coding scheme. The temporal scalable encoded Mpeg-1 is a well known and widely deployed video. The BCC mechanism, which is a generalization of TCP RENO, is a suitable selection for a temporal scalable video, since it employs a window based congestion control. It uses the window to determine the amount of the packets that can be sent without being acknowledged. Therefore since this algorithm works on the packets, it has a major advantage over other rate adaptation algorithms that employ calculations for determining the sending rate; which is its simplicity.

Using BCC, we propose a novel scheme in this thesis on the streaming video problem with a quality adaptation mechanism that adapts the video rate to available rate dictated by the BCC. The selection of the window update parameters for the used congestion control mechanism is also considered to achieve both smooth and fair transmission. The BCC mechanism employs a window based congestion control mechanism, and it does not calculate an explicit rate. However during low available bandwidth periods, it will take longer to forward packets arriving at the quality adaptation buffer. Therefore, the delay of such packets at the server increases. If the available line rate is higher than the video packets, the frames will be forwarded immediately without delaying them. By observing this property of the buffer and just by observing the delay of the quality adaptation buffer, it is possible to understand the network conditions. If not the immediate but a longer term behavior of this delay is considered, a smooth adaptation for an also smoothly varying available bandwidth is obtained. We observe the quality adaptation buffer delay and admit the frame to the buffer, if it will not spend more than a fraction of the left time to its play-out deadline in the buffer. We have used the ns-2 [9] to show that this system achieves more plausible video representation at the receiver.

The rest of this thesis is organized as follows: Chapter 2 describes the related work and the background in this area; Chapter 3 discuses the selective frame discard mechanism and some implementation details of this system in the ns-2

simulator; Chapter 4 presents the numerical results of the experiments we have employed; Chapter 5 summarizes our conclusions and possible future work.

Chapter 2

Related Work and Background

2.1 TCP/IP Networks

The Internet Protocol (IP) is the basic carrier for all kinds of Internet communication protocols. It is the protocol software that makes the Internet appear as a single and seamless communication system. All the data gets transmitted as IP datagrams. An IP datagram is the name of the packets in IP networks.

IP provides connectionless, unreliable delivery of the datagrams over the existing packet switched network. By being unreliable, it is meant that IP networks do not provide any guarantee on the packet delivery to its destination. When sources are exhausted on the path, such as buffer space, the network will discard the packet. By being connectionless, it is meant that there is no state information about the successive datagrams. Each datagram is handled independent of others. Two datagrams, sourced at and destined for the same source-destination pair, may take different routes.

The process for transferring datagrams over the network is called IP routing. Each datagram contains its own IP header which contains the source and destination addresses. If the destination is directly connected to the source through a point to point link or a LAN, then the datagram is delivered to the destination host. Otherwise, the host delivers the datagram to its default router. Datagrams are treated on a hop-by-hop basis according to their destination IP addresses, where each hop is a router, that forwards the datagrams to a closer node or a hop towards the destination.

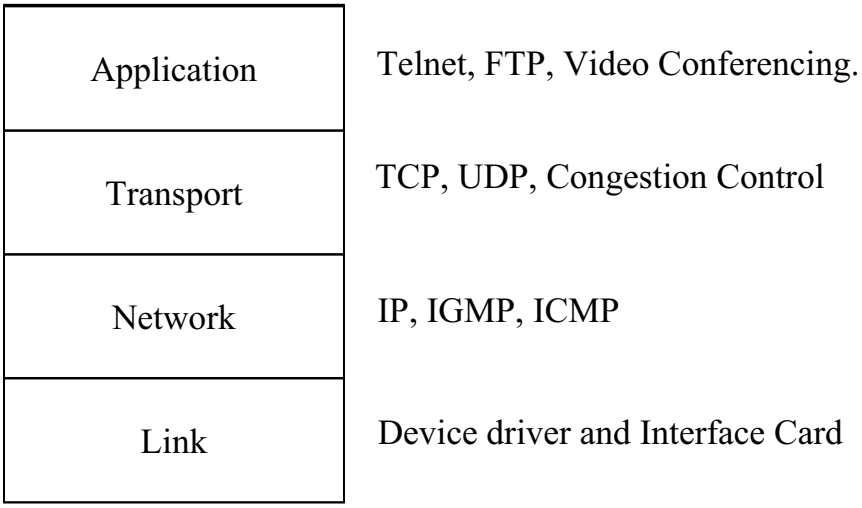

Figure 2.1: The four layers of the TCP/IP protocol suite

IP protocol does not cover all the communication standards. Instead of having a single, giant protocol that specifies complete details for all possible forms of communications, designers have chosen to divide the communication problem into sub-pieces and to design a separate protocol for each sub-piece. Doing so makes each protocol easier to design, analyze, implement and test. In this layered architecture, Internet protocol works on top of the link layer, which provides connectivity. Since, IP is connectionless and unreliable, transport mechanisms (protocols) are developed to be able to transfer the data safely. These mechanisms make use of IP. Applications sending and receiving data use the transport protocols. This hierarchical organization of the protocols leads to a layered architecture. This architecture is depicted in Figure 2.1

Each layer adds a header for the information that they need. The final form of the packet may be seen through Figure 2.2.

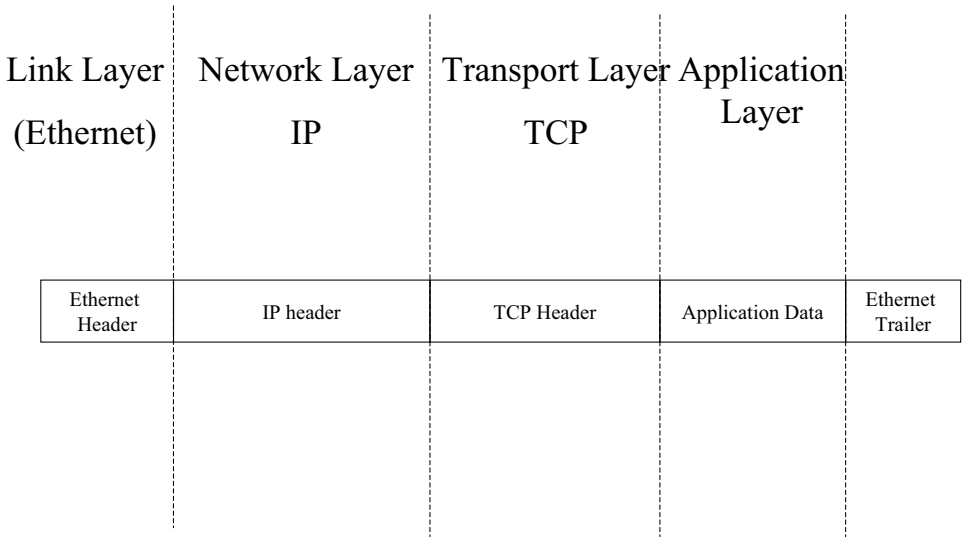

Figure 2.2: Header structure for an Ethernet Frame

2.1.1 IP Data Plane

This section describes how routers process and forward the packets, and some of the mechanisms employed by routers in order to provide better services.

A router forwards each packet from one network node to another. A source host creates a packet and places the destination address in the packet header, and sends the packet to a nearby router. When a router receives a packet, the router uses the destination address to select the next router on the path to the destination, and then transmits the packet. Eventually, the packet reaches a router that can directly deliver the packet to its final destination. The format of the packet in the Internet is unique, since a router may be connecting heterogenous networks.

Datagrams traverse the Internet by following a path from their source to their final destination. Each router along the path receives the datagram, extracts the destination address from the header, and uses the destination address to determine a next hop to which the datagram should be send. The router forwards the datagram to the next hop, either the final destination or another router. To

select the next hop efficiently and to make it possible that humans understand the computation, each router keeps information in a *routing table*. A routing table must be initialized when the router boots, and must be updated if the topology changes.

The delivery of the packet will be a *best effort* delivery, since IP does not provide any guarantee that it will handle the packet.

Buffer Management

The buffers are the waiting room for IP packets to handle the case when the outgoing rate is less than the incoming rate. The most popular queue management techniques are Drop-tail, Random Early Detection (RED), Weighted Random Early Detection (WRED). For a drop-tail queue, the packet is simply not admitted to the buffer, if the buffer is full. However the management of RED and WRED is more complex.

A RED queueing mechanism basically uses the average queue occupancy as input to a random function that decides whether there is a possibility of congestion or not. If it decides that there is congestion, it may discard some packets or mark them [10]. By dropping or the marking of the packets, the congestion control mechanisms are informed of a congestion along the path of transmission.

For RED, there are two important parameters [11], min_th and max_th that control the dropping process. Below min_th, the packets are flowing through the router by being untouched. If the buffer occupancy is above min_th and below the max_th, then packets get statistically dropped, with an increasing probability according to the average buffer occupancy. Above max_th, packets will be dropped with probability 1. The increasing probability is defined by a linear function with a constant slope which can also be set by the network administrator. Weighted Random Early Detection, is a variant of RED, that enables applying multiple policies to different flows using the same queue. Throughout this thesis, the network nodes that are considered use RED queueing mechanisms.

2.1.2 The Transmission Control Protocol

In this section, the Transmission Control Protocol (TCP) will be introduced [12]. Although TCP is a part of the TCP/IP protocol suite, it is an independent,

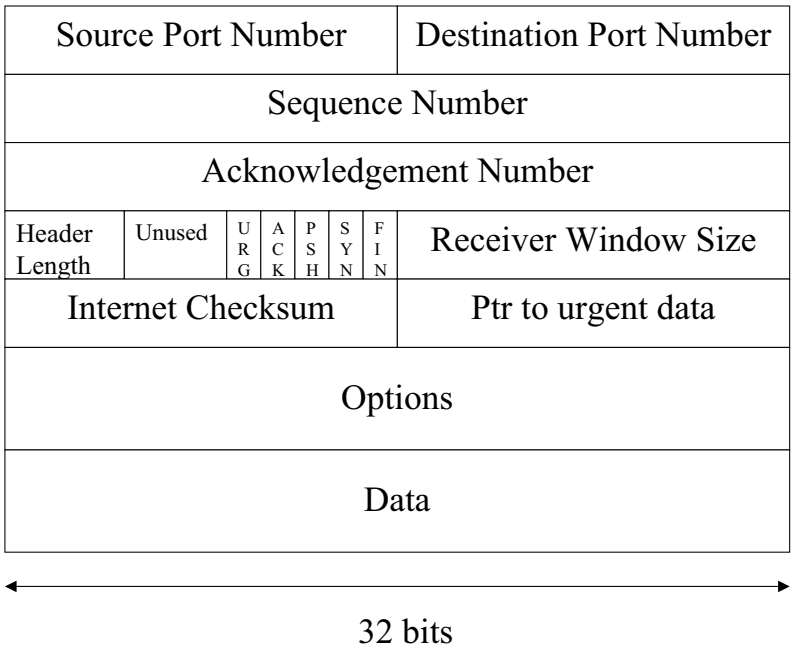

Figure 2.3: Header structure for a TCP Segment

general purpose protocol that could also be adopted for use with other delivery systems. The header structure of the TCP is as in Figure 2.3.

As already mentioned at the lowest level, computer communication networks provide unreliable packet delivery. Packets can be lost or destroyed when transmission errors interfere with the data, when network hardware fails, or when networks become too heavily loaded to accommodate the present load. Networks that route packets dynamically can deliver them out of order, deliver them after a substantial delay, or may deliver duplicates. Furthermore, underlying network technologies may dictate an optimal packet size or pose other constraints needed to achieve efficient transfer rates.

At the highest level, application programs often need to send large volumes of data from one computer to another. Using an unreliable connectionless delivery system for large volume transfers becomes tedious and annoying, and it requires every program to have error detection and recovery by its own. Because it is difficult to design, understand, or modify software that correctly provides reliability, few people can implement these functionalities successfully. As a consequence, one goal of the network protocol research has been to find general purpose solutions for the problems of providing reliable stream delivery, making

it possible for experts to build a single instance of stream protocol software that all application programs use. Having a single general purpose protocol helps to isolate application programs from the details of networking, and makes it possible to define a uniform interface for the stream transfer service.

Application programs send a data stream across the network by repeatedly passing data octets to the protocol software. When transferring data, each application uses convenient sized pieces, which can be as small as a single octet. At the receiving end, the protocol software delivers octets from the data stream in exactly the same order they were sent, making them available to the receiving application program as soon as they have been received and verified. The protocol software is free to divide the stream into packets independent of pieces the application program transfers. To make transfer more efficient and minimize the network traffic, implementations usually collect data from a stream to fill a reasonably large datagram before transmitting it across the Internet. Thus, even if the application program generates the stream one octet at a time, transfer across the Internet may be quite efficient. Similarly, if the application program chooses to generate extremely large blocks of data, protocol software can choose to divide each block into smaller pieces for transmission. This property of the TCP is also used as an underlying architecture for implementing the Binomial Congestion Control Mechanisms and is also an important issue. Combining small pieces of data means delaying the generated data, in order to combine them with a new generated data. For a video application, since the generated data consists of the frames that have a deadline to be met, delaying them may result in their failure to catch their respective play-out times. However, a large frame can be sent in smaller pieces. In this case, the possibility of missing their play-out times is only determined by the network conditions, or the available rate information. For the system proposed in this thesis, large sized frames are divided into smaller sizes, and smaller sized ones are sent without further delaying them.

Connections provided by the TCP/IP stream service allow concurrent transfer in both directions. Such connections are called *full duplex*. From the point of view of an application process, a full duplex connection consists of two independent streams flowing in opposite directions with no apparent interaction. The stream service allows an application process to terminate flow in one direction, while data continues to flow in other direction, making the connection *half duplex*. The advantage of a full duplex connection is that the underlying protocol software can send control information for one stream back to the source in datagrams carrying data in the opposite direction. Such *piggybacking* reduces network traffic.

Providing Reliability

It was mentioned that a reliable stream delivery service guarantees to deliver a stream of data sent from one machine to another without duplication or data loss. Most reliable delivery protocols use a single fundamental technique called *positive acknowledgement with transmission*. The technique requires a recipient to communicate with source, sending back an acknowledgement (ACK) message as it receives data. The sender keeps a record of each packet it send and waits for an acknowledgement before sending the next packet. The sender also starts a timer. Sender retransmits a packet if the timer expires before an acknowledgement arrives.

The problems caused by duplicate packets are handled by assigning each packet a sequence number and requiring receiver to remember which sequence numbers it has received. Acknowledgements contain these numbers, so the sender correctly associate packets with acknowledgements. TCP/IP acknowledgements are *cumulative* because they report how much of the stream has been accumulated at the receiver.

Window Based Congestion Control

Window based schemes use a technique called sliding window. This technique is more complex for positive acknowledgement and retransmissions than the simple method discussed before. Sliding window techniques utilize the network much effectively than the previous one since they allow the sender to transmit multiple packets before waiting for an acknowledgement. The easiest way to envision the window operation is to think of a sequence of packets to be transmitted as in Figure 2.4. Here, a fixed sized window is used. The protocol sends all the packets inside the window. We say that the packet is *unacknowledged* if it has been transmitted but no acknowledgement has been received. Technically, the number of packets that can be unacknowledged at a given time is constrained by the window size and is limited to a small fixed number. For example, in a sliding window protocol with window size 6, the sender is permitted to transmit 6 packets before it receives an acknowledgement. Once the sender receives an acknowledgement for the first packet inside the window, it slides the window along and sends the next packet. The window continues to slide as long as acknowledgements arrive.

In practice, the congestion window is not fixed, and TCP reacts to congestion. The congestion situation is a condition of severe delay caused by an overload of

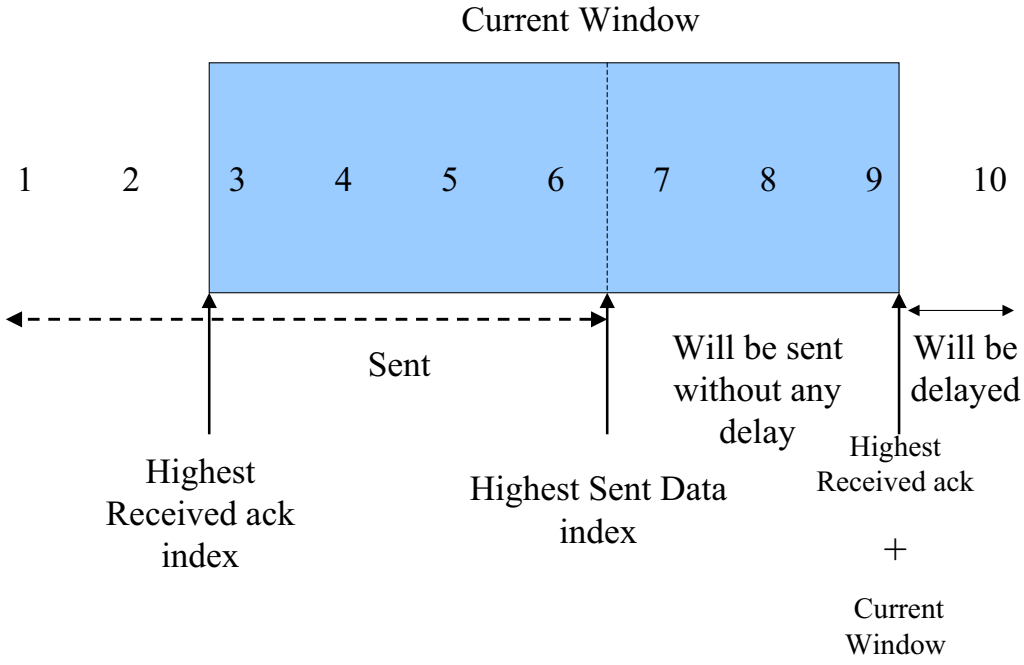

Figure 2.4: Sliding Window

datagrams at one or more switching points. When congestion occurs, router queue starts to build up, eventually leading to loss.

The end point does not know the details about where the congestion has occurred or why. To them, congestion simply means increased delay or lost packets. Most transport protocols use timeout and retransmission, so they respond to increased delay by retransmitting datagrams. Retransmissions aggravate congestion instead of alleviating it. If unchecked, the increased traffic will produce increased delay, leading to increased traffic, and so on, until the network becomes useless. This condition is known as congestion collapse [2].

To avoid such a situation, a congestion control mechanism must reduce its rate when congestion occurs. To avoid congestion, TCP standards suggest different techniques. These are slowstart, fast recovery and fast retransmit, and congestion avoidance.

During the slow start, the TCP window increases exponentially. It is named as it starts to transmit packets in a slow manner but accelerates rapidly. The initial value of the window is one or two packets. Once the window exceeds the threshold called the slow start threshold (*ssthresh*, which determines an upper

bound for the window size that can be incremented exponentially, the Congestion Avoidance (CA) phase starts and congestion window grows linearly rather than exponentially. Congestion window is updated as in equation (2.1) during this phase. The window is increased by a single packet size per round trip time, actually the time it takes to receive an acknowledgement after the packet is transmitted. This window increasing phase continues until a loss occurs.

TCP algorithms differ in terms of the way they react to congestion. The Tahoe type congestion control algorithm detects the loss by waiting a long period for the retransmission timer to timeout and sets its congestion window to a single packet. Reno, a variant of Tahoe, also sets its congestion control window to one packet as a result of timeout, however, react by employing a fast retransmit and fast recovery algorithm, upon the arrival of three duplicate acks. Vegas tries to avoid congestion while providing good throughput. It tries to detect congestion based on round trip time (RTT) estimates. Longer RTT will mean higher probability of congestion. So the algorithm lowers the rate by lowering the rate linearly when a possible loss is predicted [13].

As mentioned in the above paragraph RENO algorithm (RFC 2581) detects the lost packets by the arrival of three duplicate acknowledgements, which are generated by the receiver immediately after an out of order sequence has arrived. Since there are other reasons of a duplicate acknowledgement, the sender should wait for the same acknowledgement four times, and after that it recovers from the loss by employing fast recovery and fast retransmit algorithms.

The most generally deployed algorithm today is the TCP Reno algorithm [13]. The Binomial Congestion Control (BCC) [4] is a smoothed version of Reno in its responses to congestion. TCP-Reno responds to a lost segment by halving its congestion window, and if there is no loss it increases its congestion window one segment per RTT. BCC avoids to increase its window by one packet per round trip time, and does not lower its window by half. Instead, it uses some parameters that will result in the same average throughput but in a less oscillatory manner. Binomial Congestion Control is a major part of this thesis and is based on TCP-RENO.

Both RENO and Binomial Congestion Control mechanisms share their response and method of detecting a loss. Fast Recovery and Fast Retransmit algorithm is the key element of the lost detection and the recovery. This algorithm functions as follows; as the third duplicate acknowledgement is received, *ssthresh* is set to maximum of $2*SMSS$ and the $FlightSize/2$, where the *FlightSize* is the

amount of outstanding data in the network. After the *ssthresh* is set, the lost segment is retransmitted, the window is set to *sstresh* plus $3*SMSS$ at most, where $SMSS$ is the senders maximum segment size. Extra 3 segments are for the packets acked and therefore they are not in the network but are safely at the receiver. This artificially inflates the window. For each additional duplicate ack, the window is also inflated by an $SMSS$. This also artificially inflates the window. A new segment is transmitted as soon as the window allows. This mechanism increments the number of packets that are unacked. However, transmission rate will not exceed the transmission rate for the new value of $window = ssthresh$. As the next ack, acknowledging new data is received, cwnd is set to *ssthresh*. Algorithm stays active around 1 round trip time.

2.1.3 The User Datagram Protocol

UDP uses the underlying Internet Protocol to transport a message from one machine to another, and provides the same unreliable connectionless datagram delivery semantics as IP. It does not use acknowledgements to guarantee messages arrive, does not order incoming messages, and does not provide feedback to control the rate at which information flows between the machines. Thus, UDP messages can be lost, duplicated, or arrive out of order. Furthermore, packets may arrive faster than the recipient can process them.

The User Datagram Protocol(UDP) provides an unreliable connectionless delivery service using IP to transport message between machines. It uses IP to carry messages, but adds the ability to distinguish among multiple destinations within a given host computer.

An application that uses UDP, fully accepts the responsibility for handling the problem of reliability, including message loss, duplication, delay, out-of-order delivery, and loss of connectivity.

Each UDP message is called *user datagram*. Conceptually, a user datagram consists of two parts: a UDP Header and a UDP data area. As Figure 2.5 shows, UDP header is divided into 16 bit fields that specify the port from which it has been originated, the port which it has been destined, message length, and UDP checksum. The UDP checksum provides the only way to guarantee that the packet has arrived intact, since IP header does not provide a checksum for the data part it carries. A user datagram, that is going to be transmitted over the

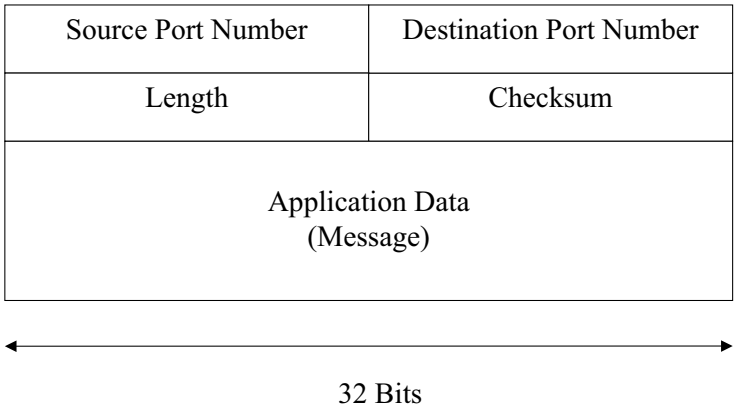

Figure 2.5: UDP Header

Internet, is encapsulated in an IP header that contains the necessary information for the packet to travel along the Internet and reach to its destination. This IP header is encapsulated into the link layer headers as it is transmitted over the links.

2.1.4 Congestion Control

The unresponsive flows that do not use end to end congestion control may lead to both unfairness and congestion collapse of the Internet.

Unfairness caused by the absence of end to end congestion control, is mainly from the interaction of TCP with unresponsive UDP flows. TCP flows reduce their sending rates in response to a congestion. Since TCP constantly reduces its rate in response to a packet drop, the UDP flows use the most of the available bandwidth.

For two different users employing TCP that are similarly situated, TCP provides roughly the same bandwidth to both. However, TCP congestion control

mechanisms produce rapidly varying transmission rates. While several applications can tolerate these oscillations, streaming applications such as video and audio do perform better with congestion control mechanisms that respond more smoothly to a loss and have smoother bandwidth profile.

Since uniformity is necessary for fairness, and provide better solutions for the multimedia applications **TCP Friendliness** is proposed. A congestion control mechanism is TCP friendly if its bandwidth usage, for a constant loss rate, is same as that of TCP [14]. In other words, the throughput of a TCP friendly algorithm on long term basis should be similar or less to that of TCP. An algorithm that is TCP friendly can achieve smoothness which streaming requires, while interacting fairly with the main data transmission protocol TCP.

On the other hand **congestion collapse** occurs when an increase in the network load results in a decrease in the useful work done by the network [2]. The first congestion collapse was caused by the unnecessary retransmissions of the TCP connections. However, the problems that are caused by this type of collapse have been corrected by improvements on timers and congestion control mechanisms.

Another cause of congestion collapse is the "undelivered packets". This arises when, at a node, the packets that will not be able to reach to their destination are forwarded. Main cause for such a situation is the increasing deployment of open loop applications that do not have congestion control.

In order to prevent the congestion collapse scenarios, and provide the fair interaction between different flows, TCP friendly congestion control mechanisms are proposed. For better performance of video applications, delay and bandwidth requirements should also be met. In order not to annoy audience by constantly oscillating quality, the rate adaptation should be made smoothly on longer time scales.

Recently Proposed Congestion Control Algorithms

There are various types of rate adaptation and congestion control schemes proposed in the literature. They all claim to be TCP-Friendly. These methods differentiate from each other based on methods of adapting their rates. Some schemes use window based methods, where as some perform rate based adaptations. The ones employing rate based techniques adapt their rate according to the TCP throughput model or an additive increase, multiplicative decrease

(AIMD) based method. For a scheme using window based mechanisms, window increment and decrement govern the rate control and the TCP friendliness can be achieved by suitably choosing parameters of the window adjustment algorithm.

Rate based Schemes are Rate Adaptation Protocol (RAP) [5], TCP-Friendly Rate Control (TFRC) [3], a model based TCP-friendly rate control protocol (TFRCP) [15], the loss-delay based adjustment algorithm (LDA) [16], Smooth and Fast Rate Adaptation Mechanism (SFRAM) [17], Direct Adjustment Algorithm (DAA) [18].

RAP [5] adjusts its rate by adapting the transmission times of the packets in an AIMD manner. It has two mechanisms for adapting the transmission rate. The coarse granular one works as follows: If there is no congestion, it shrinks the transmission times in an additive manner. If there is a loss of packet, they double the transmission timeout resulting in halving the rate. They also have fine granular rate adaptation that emulates the rate change of TCP because of RTT variation. They also consider multiple losses in one round trip time as a single one. This is very important since the fast recovery and retransmit algorithm of the TCP RENO is known not to recover well from the loss of multiple packets in a single round trip time.

TFRC [3] is a protocol based on the TCP response function. However, its not as aggressive as TCP. The receiver calculates the loss rate and RTT and informs the sender. The sender adjusts the rate according to a TCP throughput equation using these estimates. The smoother estimation of the parameters result in smoother rate adaptations.

TFRCP [15] is also a model based approach, however a different model for this protocol has been used. LDA [16] relies on Real Time Control Protocol (RTCP) feedback information. If no loss has occured, the rate is adjusted in an additive incremental manner. If loss has occurred, rate is decremented proportional with the loss. SFRAM [17] smoothly adjusts its rate when there is not a distinct bandwidth change. If there exists large variations, it adapts in a rapid manner. It averages the measurements in an adaptive way. DAA [18] also relies on the RTCP feedback mechanism. DAA employs both TCP-style AIMD and TCP throughput model.

As mentioned before, the window based algorithms use their window for determining the number of packets that can be transmitted, and yet not confirmed to be at the receiver. These packets are accepted as “in flight”. Changes in

the window size will effect the number of packets transmitted. Among the window based algorithms there are linear and non linear generalizations of TCP algorithm.

The linear generalization, Generalized AIMD (GAIMD) [19], develops a rule for using the α and β parameters in Equation 2.1 and 2.3. In this equation w is the window size, R is used for one round trip time period and δ_t is used to indicate an immediate or a relatively short term change.

$$\omega_{t+R} = \omega_t + \alpha\omega_t^k; \qquad \alpha > 0 \tag{2.1}$$

$$\omega_{t+\delta t} = \omega_t - \beta\omega_t^k; \quad 0 < \beta < 1 \tag{2.2}$$

Binomial Congestion Control [4] proposes a class of nonlinear generalization of TCP. These algorithms are motivated in part by the needs of streaming audio and video applications for which a drastic reduction in transmission rate upon each congestion indication (or loss) is problematic. Binomial algorithms generalize TCP-style additive-increase by increasing inversely proportional to a power k of the current window (for TCP, $k = 0$); they generalize TCP-style multiplicative-decrease by decreasing proportional to a power l of the current window(for TCP, $l = 1$). We show that there are an infinite number of deployable TCP-compatible binomial algorithms, those which satisfy $k + l = 1$, and that all binomial algorithms converge to fairness under synchronised-feedback assumption provided $k + l > 0$, $l \geq 0$.

Developers of BCC assumes, a TCP friendly algorithm has a throughput proportional with $\lambda\alpha S/(R\sqrt{p})$, where the λ is the throughput, S is the packet size, R is the round trip time, and p is the packet loss rate. For binomial algorithms throughput is around $\lambda\alpha 1/p^{\frac{1}{k+l+1}}$, and a binomial congestion control algorithm is TCP compatible if only $k + l = 1$ for suitable α, β. There are two types of binomial algorithms that are widely deployed. One of them is Inverse Increase Additive Decrease (INV) with parameters ($k = 1, l = 0$), and the other is SQRT called after that its parameters ($k = 1/2, l = 1/2$). For more information please refer to [4].

Even though a fair interaction is envisioned for similiar loss rates, for the cases with drop-tail queues some competing flows may experience different loss rates, therefore they experience different rates. By the implementation of the RED queues at the node's interface that is connected to the bottleneck link, this problem may be solved. There is also a study on the fairness of the binomial congestion control mechanisms. This study has been reported in the [20], which is a comparative study of binomial congestion control mechanisms. By the results

of this paper, the algorithms found to be converging to a fair allocation of the bottleneck bandwidth for the k is in the range of $[0, 0.2]$.

2.2 Video Transmission over IP Networks

2.2.1 Video Standards

The digital representation of a sequence of images requires a very large number of bits. However, video signals naturally contain a number of redundancies that could be exploited in the digital compression process. These redundancies are either statistical due to the likelihood of occurrence of intensity levels within the video sequence, spatial due to similarities of luminance and chrominance values within the same frame, or temporal due to similarities encountered amongst consecutive video frames. Video compression is the process of removing the redundancies in the video and representing the video with less amount of bits for reducing the size of its digital representation. Extensive research has been conducted since the mid eighties to produce efficient techniques for image and video compression.

The standards organizations ITU (International Telecommunication Union) and ISO (International Standards Organization) both released standards for still image and video coding algorithms. After the release of first still image standard, namely JPEG (alternatively known as ITU T.81) in 1991, ITU recommended the standardization of its first video compression algorithm, namely ITU H.261 for low bit rate communications over ISDN at rates multiple of $64kbits/s$, in 1993. The MPEG-1 standards for audiovisual data storage on CD ROM (1991), MPEG-2 (ITU-T H.262, 1995) for broadcasting applications have been released. ITU H.263 (1998) was released for very low bit rate communications over PSTN networks; then the first content-based object-oriented audiovisual compression algorithm was developed, namely MPEG-4(1999). By means of research on the video technology, scalable coding techniques such as two layer MPEG-2 and the multi-layer MPEG-4 standards are developed. There are also switch-mode techniques that have been developed, which can accommodate more than one coding algorithm in the same encoding process to result in an optimal compression of a given video signal. Some newly developed techniques employ joint source and

channel coding techniques to adapt the generated bit rate and hence the compression ratio of the coder to the time varying conditions of the communication medium.

Throughout this section, MPEG-1, MPEG-2, H.263 compression schemes are introduced. The system proposed in this thesis is working on the MPEG encoded videos. However, it may be possibly used with an H.263 coded video on very low bit rates.

2.2.2 MPEG-1, MPEG-2 and Concepts

MPEG stands for the Moving Pictures Expert Group, which is a committee under the Joint Technical Committee of ISO. To focus the design of the system around a practical objective, certain parameter constraints are defined. These parameter values represent boundaries; a bit stream with any parameters outside these boundaries is not accepted as an MPEG-1 stream. Therefore an MPEG-1 decoder is not required to decode it. MPEG standard describes various tools that may be used to perform compression, and gives some hints of how these might be implemented.

MPEG-1 and MPEG-2 achieve both spatial and temporal compression of the image sequence, and all known techniques of this types of analysis are computationally complex. However, MPEG-1 and MPEG-2 are both designed as asymmetric systems; the complexity of the encoder is much higher than the decoder.

The top level definition in MPEG-1 is a *sequence of pictures*. A sequence can be arbitrary in length and can represent a video clip, a complete program item, or a concatenation of programs. Within the sequence, the next lower definition is the *group of pictures* (GOP). In the simplest form of encoding without temporal compression, the GOP can be a single picture. However, in typical MPEG application the GOP will include pictures coded in three different ways and arranged in a repetitive structure most commonly between 10 and 30 pictures long. A picture or a frame consists of *slice*s and *macroblock*s. A macroblock contains all the information required for an area of the picture representing 16×16 luminance pixels. Macroblocks are numbered in scan order (top left to bottom right). In MPEG-1, a slice is any number of sequential macroblocks. The main significance of a slice is that, it is encoded without any reference to any other slice; this means

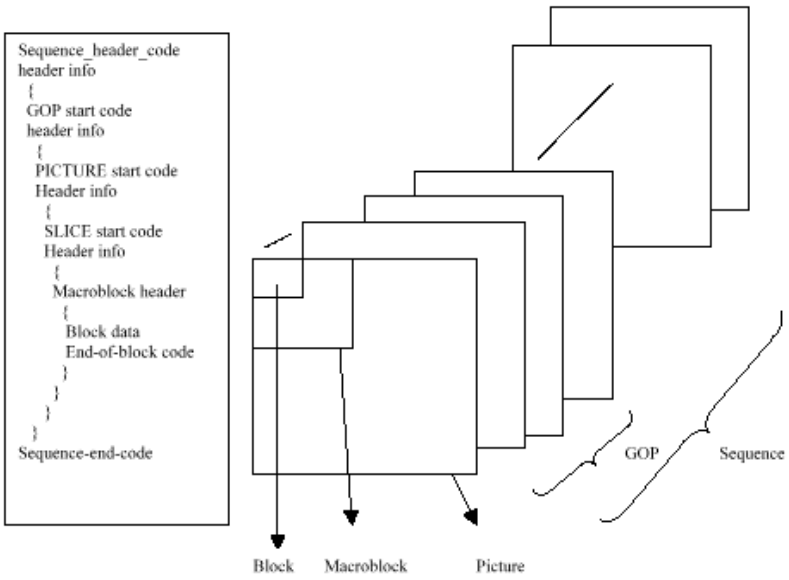

Figure 2.6: Slices in MPEG-1

that if data is lost or corrupted, decoding and recovery can usually commence at the beginning of the next slice. The hierarchical organization of MPEP video sequence is given in Figure 2.6.

There are three types of frames. They are I, P and B type frames. I (Intra) frames are the frames that are encoded using only the information within that frame. In other words it is spatially coded. The non intra frames use information from outside the current frame, from frames that have already been encoded. For a non intra frame, motion compensated information is used for that macroblock. This compansation results in less amount of total data. As in Figure 2.7, a region in frame N is searched in the frame N+1 in a limited "Search Area". After the best matching part is found a motion vector is generated that contains the necessary information for the prediction process.

The I-frames are coded solely on its own information. The P frames are predicted unidirectionally from I frames or a preceding P frame, and the B frames are predicted from proceeding P frames and preceding I or P frames bidirectionally. The I frames and the P frames are called anchor frames, because they will be used as references in coding of other frames using motion compensation.

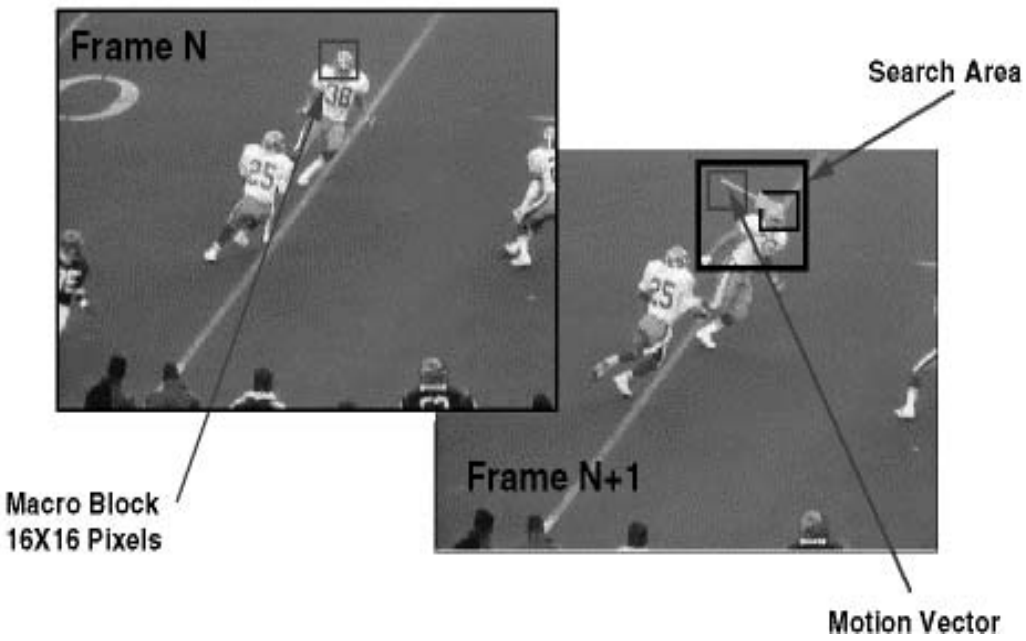

Figure 2.7: Motion Estimation

B-frames, however, are not anchor frames, since they are never used as a reference. The GOP starts with an I-frame. It is possible to place couple of B frames preceding the I frame. The first P frame is encoded using the previous I frame as a reference for temporal encoding. Each subsequent P-frame uses the previous P frame as its reference. Therefore an error occurred in the previous frame will propagate as the P frame becomes the reference of others. The B frames use the previous anchor (I or P) frame as a reference for *forward prediction*, and the following anchor as a reference for *backward prediction*. B frames are never used as a reference for prediction.

As a summary, the encoding order of the frames may not be similar to the order of the pictures needed to be shown. Therefore the transmission order of the frames may not also be the same as their display order.

The MPEG-2 has similar basic principles. It is possible to express MPEG-2 as an MPEG-1 with improvements such as, tools for interlace, scalable syntax, a range of profiles and levels accommodating wide range of applications, plus a system layer to handle multiple program streams. While the MPEG-2 standards are accepted as more complex, MPEG-1 provides basics.

The scalability techniques are introduced in order to make it possible for a part of a video sequence to be decoded at a desired quality. The minimum decodable subset of the bitstream is called the base layer. All other layers are enhancement layers, which will improve the quality of the video. There are three types of scalability; Spatial, SNR, Temporal Scalability. Spatial (pixel resolution) scalability provides the ability to decode video at different frame sizes. By adding them on to each other it is possible to end up with a picture size equal to the original video. SNR scalability offers versions of the video, coded with different quantizer step size for the quantizer. Therefore, resulting in coarser to gradually improved quality. Temporal scalability refers to decodability at different frame rates without first decoding every single frame. There may be a composite use of the above techniques or they may be used alone.

The system proposed in this thesis, makes use of the MPEG-1 streams that are temporal scalable encoded. Our mechanism selectively discards some of the frames in order to adapt the rate of the video available bandwidth in the internet. The video stream used has a bit rate around 2Mbits/s.

2.2.3 Video Streaming over IP

It is possible to transmit a stored video in two different modes. They are the download and the streaming modes. In a download mode user downloads the video file, and plays back the video file after download has been completed. However, full file transfer usually takes long and sometimes unacceptable transfer time. On the contrary, in the streaming mode, the video is played out while parts of the video are still being transmitted. Since it has a real-time nature, video streaming applications have some requirements on the transmission medium, namely bandwidth, delay and loss requirements. However, today Internet does not have any QoS (Quality of Service) support to guarantee that the packets will be delivered within the requirements. Furthermore, for multicast, it might be hard to efficiently meet different requirements of different users.

The General Architecture

Figure 2.8 presents a general architecture of video streaming. The raw audio and video data are compressed by compression methods before a request is made and stored into the storage devices. As a client requests video data, streaming server retrieves compressed video/audio data from storage devices and then

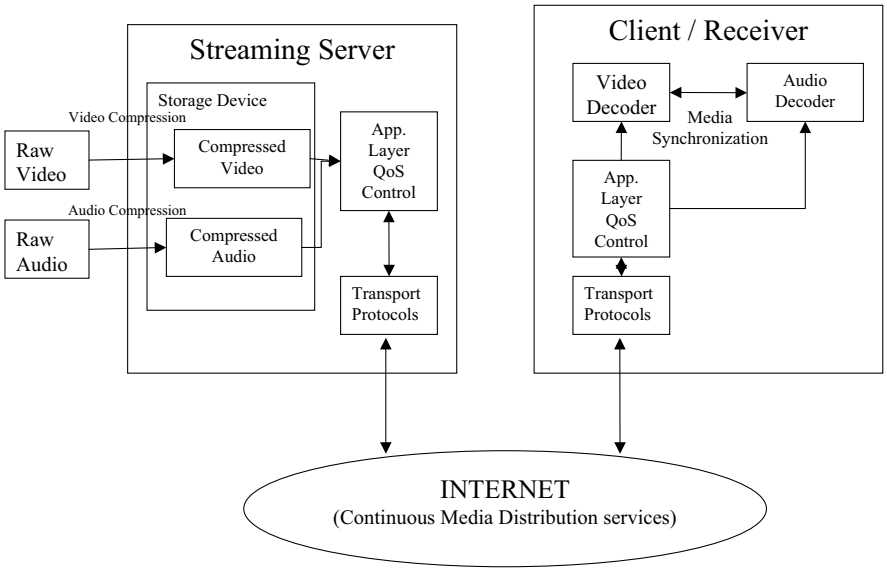

Figure 2.8: A general video streaming architecture [1]

the application layer QoS control module adapts the video/audio bit-streams according to the network status and QoS requirements. After the adaptation, the transport protocols packetize the compressed bit-streams and send the packets through the Internet. Packets may be dropped or experience excessive delay inside the network due to a congestion. To improve the quality of video transmission, continuous media distribution services are deployed in the Internet (e.g. caching). To achieve synchronization between the video and the audio decoder, video synchronization mechanisms are required. The above six areas are closely related and they are the elementary granules of the video streaming architecture.

Video compression of a raw video is required to prevent the inefficient usage of the bandwidth. The video coding can be further classified into two classes, they are scalable and non-scalable coding. The available types of scalable video encoding are SNR, spatial, temporal scalabilities, and newly developed FGS, PFGS, namely fine granular and progressive fine granular scalability techniques.

The requirements of the streaming video such as bandwidth, delay, loss, VCR like functionality, decoding complexity imposed on the video encoder and decoder, and techniques addressing these issues should be emphasized at this point. *Bandwidth* cannot be set to a certain level on today's Internet even though the

video has a minimum bandwidth requirement. It is also desirable for a streaming server to employ congestion control to avoid the congestion that it may cause inside the Internet. The streaming servers using UDP do not consider the fact that the network may be overloaded with several of these streams. Therefore the data flow and finally the streams themselves will suffer randomized and excessive losses. The situation may lead to a worse situation, a congestion collapse. In this case, even if the network resources are fully used, the successfully transmitted packets are very few. *Delay* is another requirement. Streaming video requires bounded delay on an end to end basis. The play-out will pause if there are missing packets, which is displeasing to humans. Play-out buffering is required in order to suppress the time-varying delay that the Internet introduces . *Loss* is a fact of Internet. In order to prevent the loss of information as a whole the multiple description coding might be implemented. VCR like functionality provides user a tool to command the stream to start, pause, and fast forward. Decoding complexity is an issue that is mostly related to mobile applications. Since they have limited amount of battery, applications running on these devices must be simple.

Application layer quality of service control tries to adapt video quality, to changing network conditions by switching through different quality levels. These techniques include congestion and error control. The congestion control aims to prevent the packet loss and reduce delay. Error control has the obligation of improving quality in the presence of packet loss. Error control mechanisms include forward error correction (FEC), retransmission, error resilient encoding, and error concealment.

Congestion control is necessary to prevent packet loss and delay. Bursty loss and excessive delays are two facts that have devastating effects on the quality of the video, and they are generally caused by network congestion. For a normal video streaming application, *rate control* attempts to minimize the possibility of network congestion by matching the rate of the video stream to the available network bandwidth.

For a pre-compressed video, *rate shaping* is the mechanism to match the rate of the video sequence to the target rate constraint. The main contribution made in this thesis is the proposal of a rate shaper. However there are different types of rate shapers. They are codec filters, frame dropping filters, layer dropping filters, frequency filters and re-quantization filters. A *codec filter* decodes the sequence and encodes it according to the available rate over the network. A *frame dropping*

filter can distinguish frames and drop frames according to their importance. The dropping order would be first B frames, later P frames and at last I frames. The frame dropping filter is used to reduce the data rate of a video stream by discarding the necessary amount of frames and transmitting the remaining ones at a lower rate. This filter can be both used at the source or in the network. *Layer dropping filter* can distinguish between layers of a scalable coded video and drop them according to their importance. The *frequency filter* works in the frequency domain, and discards some of the frequency domain coefficients. It may employ low pass filtering, color reduction, and color to monochrome filtering. The *requantization filter* performs its operations on the DCT coefficients, and changes the quantization step size. Our proposed system, employs a frame dropping filter at the server.

Continuous media distribution services implemented in the Internet, provide the adequate network support to decrease the delay, and packet loss ratio. Built on top of the IP protocol, continuous media distribution services are able to achieve QoS and efficiency for streaming video over the best effort Internet. Continuous media distribution services are network filtering, application level multicast, and content replication.

The streaming servers are the key components for providing streaming services. To offer quality streaming services, the servers are required to process multimedia data under timing constraints. Furthermore, they are required to support interactive functionalities such as rewinding, fast forwarding. The fundamental components of a server are a communicator, an operating system, and a storage system.

Media synchronization is a characteristic functionality of a video streaming application. By use of the media synchronization mechanisms, the video content can be displayed at the receiver as it was originally recorded. Best known example of synchronization is lip movements of a speaker and the speech.

The protocols for streaming media delivery are standardized for the communication between clients and streaming servers. Protocols for streaming servers provide, addressing, transport, and session control. They can be classified into three groups, network layer, transport protocols, session control protocols.

Network layer protocols provide basic network service support such as network addressing. The IP serves as the network layer protocol for Internet streaming protocol.

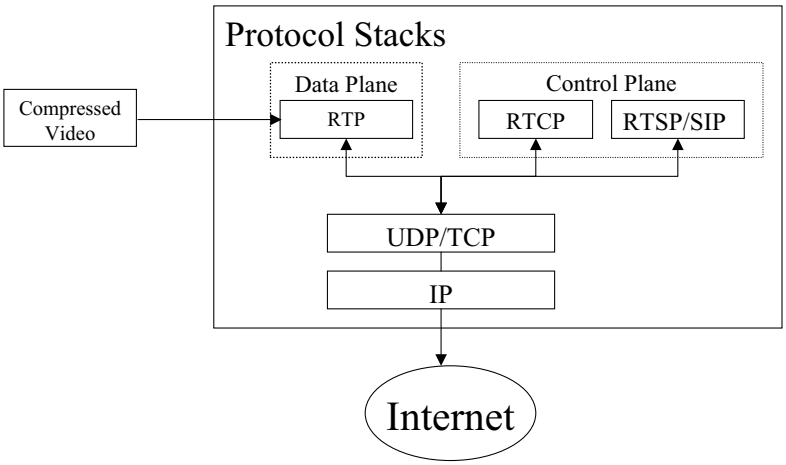

Figure 2.9: Protocol stacks for streaming media [1]

Transport protocols provide end to end network transport functions for streaming applications. Transport protocols include UDP, TCP, real-time transport protocol (RTP), and real time control protocol (RTCP). UDP and TCP are layer 3 protocols, however RTP and RTCP are layer 4 transport protocols, which are implemented on top of the TCP and UDP.

Session control protocol defines the messages and procedures to control the delivery of the multimedia data during an established session. The RTSP (Real Time Streaming Protocol) and the session initiation protocol (SIP) are such session control protocols.

In Figure 2.9, the relationship between these three types of protocols can be observed. On the data plane, the compressed video/audio data is retrieved and packetized at the RTP layer. The RTP provides timing and synchronization information. The RTP packetized streams are then passed through the TCP/UDP layer and the IP Layer. The latest research on video streaming techniques provides alternatives to the use of TCP or UDP. There are newly proposed techniques which have been emphasized in the IP networks section. At the receiver side, the media streams are processed in the backwards order, first IP

layer, then UDP/TCP layer, and the control plane. The control signals are also encapsulated in TCP, and IP headers.

The transport protocols for media streaming include UDP, TCP, RTCP, RTP. UDP and TCP provide the basic transportation functionality, RTP and RTCP run on top of the UDP/TCP.

UDP and TCP protocols provides functionalities such as multiplexing, error control, congestion control, or flow control. The port numbers of the UDP and TCP headers make it possible to multiplex different applications running on the same machine with the same IP address. For error control, TCP and UDP implementations employ a checksum to detect bit errors. If bit errors are detected on any packet, that packet is discarded. TCP uses retransmissions to recover from lost packets, therefore provides a reliable connection. However the retransmissions of TCP may not be suitable for time stringent applications. However, for a unidirectional streaming, the timing is not very stringent. TCP also employs a congestion control mechanism which prevents the streaming application from sending too much data, and overloading the network. TCP also has a mechanism to prevent the receiver buffer from overflowing while UDP does not have. For the simulation purposes the receiver buffer has been assumed sufficiently large so it never overflows in the system proposed. The actual TCP implementation naturally handles this problem. If a receiver buffer overflow occurs, TCP will lower the transmission rate. Therefore the rate shaping will take place according to this new rate.

UDP is much more generally employed, since TCP has a oscillatory behavior and may have excessive delay. UDP does not provide any guarantee on the delivery of the packet, therefore receiver will need to rely on the RTP system.

RTP is an international standard protocol designed to improve end-to-end transport functions for supporting real-time applications, where RTCP is a complementary protocol that provides QoS feedback to the participants of an RTP session. Indeed RTP is the data transfer protocol while RTCP is a control protocol.

RTP does not guarantee QoS or reliable delivery. Its functionalities are time-stamping, sequence numbering, payload type identification, source identification. The time stamping provides marking for application to be able to synchronize different media streams. Sequence numbering provides a way to detect out of order delivered packets. Payload's type identification indicates the type of data

that has been carried. It indicates whether the content has an encoding such as MPEG1/2 or audio. Source identification indicates the source of each packet.

The function of the RTP header is providing the necessary mechanisms that the UDP does not employ. The TCP however already has these types of fields. It has its own sequence numbers, synchronization markers. The source identification can be a challenge, however IP header also provides the necessary information. Still the selection of headers is a research issue by itself, some modification of TCP header will provide the necessary functionalities. In order to mention, the selection and implementation of a header structure is an open issue and is out of the scope of the research presented. The necessary sequence numbers, timing functionalities are used as the ones of the video itself, and through the changes of the TCP header without disturbing the functionality of the TCP itself.

RTCP is the control protocol designed to work together with the RTP. RTCP provides QoS feedback, participant identification, control packet scaling, inter-media synchronization, minimal session control information. The QoS feedback includes statistics about the reached packets, fraction of the lost packets, delay, packet inter-arrival jitter. Based on the feedback, the sender can adjust its transmission rate. TCP again has all of these information, through mechanisms it already has. The rate is also dictated by TCP. The participant identification provides a mechanism to identify the source of a packet. Control packet scaling, scale the RTCP control packet transmission with the number of participants. Among the control packets, 25% are allocated to the sender reports and 75% to the receiver reports. To prevent the control packet starvation, at least one control packet is sent within 5 seconds at the sender or receiver. Inter-media synchronization mechanism is the indication of the realtime and the corresponding RTP time stamp. Minimal session control information is used to provide a mechanism to transport the session information such as session names.

RTSP and SIP are two session control protocols. They provide the basic initiation, VCR like functionalities, and the session termination functions.

2.2.4 Quality Adaptation

The quality adaptation is the method of adapting the rate of a video through changing its quality, so that the required rate matches the rate determined by the congestion control mechanism. The system that we have proposed also aims

to adapt the rate of the video according to the rate determined by the congestion control mechanism.

Most of the work done in this field tries to achieve a constant quality over long periods while performing a TCP-friendly transmission. [21], [22] employ TCP-Reno to satisfy the concerns on TCP friendliness. [23], [24] chose to adapt the video rate to smoother bandwidths, so they use TCP-friendly mechanisms like TFRC and TFRCP. The systems proposed in [25], [26] aim to minimize the number of packets that miss their play-out times by employing adaptive play-out mechanisms.

Some mechanisms are designed for both encoding the video and providing the transmission of the video [27], [28], [29]. The work presented in [30], [8], [31], [32], [33], [34] are the examples of systems that adapt the pre-encoded or stored video to a available bandwidth based on either a feedback from the congestion control mechanism or a dynamic mechanism.

The encoding of video adaptive to the available bandwidth is called *rate control* [35]. [36], [37], [38] dictate some improvements to the underlying network architecture in order to be able to meet the demands of the video streaming applications.

Layered quality adaptation is a pre-encoding of video that makes it possible to stream the video with different rates from a discrete set. [39], [40], [6] are some proposed mechanisms that have implemented such layered quality adaptation.

There are video rate shaping mechanisms implemented in the network [41]. There are also some techniques that were employed for the constant bit rate applications. However, these adapt a rate of a pre-encoded video to a given rate [42], [43].

[22] is actually an overview of the TCP streaming. Their point on retransmissions is as follows: since video on demand has interactivity limited to the control commands such as start, pause, fast-forward, it may be possible to make retransmissions without great deal of problems. However, they mention that, retransmission is an issue that should be seriously considered. Their other point is that quality adaptation should be made based on the long term oscillations of the bandwidth, otherwise audience may find the video very annoying. Their vision is that an ECN capable network would be really suitable for the increase

of the performance. In an ECN capable network it is possible to adjust the rate almost without doing any retransmissions and losing any packets.

[21] employs dynamic rate shaping and a TCP congestion control mechanism but in a semi-reliable way. Their rate shaping filter employs two techniques, re-quantization and elimination of some transform coefficients. These eliminations are based on the minimization of the distortion caused by the rate change. It is left to a decision mechanism to retransmit the lost data.

[25] and [44] are proposing a mechanism that adapts play-out speed according to the network conditions. The first paper proposes a mechanism for wireless error-prone channels. The second paper is a variant of the system for real-time media streaming. [26] is offering an advance over adaptive play-out mechanisms by adaptively encoding the video.

Some of the mechanisms, in order to achieve best possible quality, both encode and also further shape the video according to the available bandwidth [27]. Their methodology toggles between modes of transmission, which are sending I frames, I and P frames, and sending all frames. The condition of the network is estimated by the penalty assigning to the lost packets in a differential manner.

[32] is one of the most similar systems to our mechanism in the sense of the method employed for rate shaping. Even though the system implemented works on the TCP output buffer and uses Binomial Congestion Control parameters, their implementation is related to the exceeding of a threshold. If this threshold on the TCP buffer is exceeded, packets are gradually rejected with an increasing probability. The video streamed is also MPEG-4 video coded with a transformation based on wavelet transformation coding techniques.

[30] is actually developed for wireless channels. The algorithm is based on the play-out deadlines and the cost assigned to each of the frames. The cost of frames is assigned according to their position in the GOP which actually define their importance. I frames have smaller cost than the others. A frame is sent according to the combined consideration of the network delay, their cost and the play-out deadlines. According to this mechanism, the video packets are scheduled with respect to their importance instead of their original playback order. Their similarity to our system is the usage of the delay as a constraint on the decisions.

[8] makes use of TFRC and the MPEG-4 Fine Granular Scalability (FGS) that has been developed by the Microsoft Research China. The MPEG-4 FGS

provides enough mechanisms to match the rate of the video to the available bandwidth. Matching may not be perfect however, on the long run, it does satisfy the limiting rate that has been dictated by the congestion control mechanism.

[31] re-quantizes the DCT coefficient to adapt the video rate to the available bandwidth information. The congestion control mechanism used is a TCP based mechanism with limited retransmissions.

[33] is also a technique used for the wireless channels. They implement an algorithm that reduces the effect of the packet losses by selectively sending the packets. They prioritize packets according to their relative importance in stream. They transmit a packet in the slot if it will not prevent the following higher priority ones being transmitted in the next slot.

[34] employs a priority based technique which will ensure the minimum frame rate delivered before transmitting the enhancement layers. The technique defines a window for the transmission and forwards the base layer data, if there is still room for upper layers it forwards them. This mechanism also works on the output buffer of the congestion control mechanism, which is similar to the system we propose.

[36] proposes some mechanisms to optimize the video streaming most efficiently in a loss or delay differentiated network. However, as it dictates constraints on the network, it is harder to be implemented. [37] investigates the possible improvements via employing just simple prioritization on the flight, while the packets are trans-passing through the network. [38] also implies differentiation based on delay and loss.

Layered video is a basic method for adapting to the available bandwidth. [6] employs layered video streaming techniques. In this scheme, lower and higher layers are both streamed when there is excess bandwidth, however during the times when there is not enough bandwidth only the higher layers are streamed. The lower layers are drained from the buffers, accumulated at the client, during low bandwidth situations. By employing this type of mechanism, any layer that is buffered is mostly guaranteed to be used at the client. No buffered data will be dropped since the arrival of the lower layers are not sufficient. The used congestion control mechanism is RAP. [40] is an implementation of this method to Binomial Congestion Control Mechanism.

[41] is a technique that employs MPEG filtering in the network which also discards the frames considering their dependencies.

[43] suggests one of the systems developed for the given constant bit rate. It considers the client buffer constraints as well as QoS metrics at the client side. This information is used to decide on whether to discard the frames or not. The question they are trying to answer is, "What percentage of the frames can be transmitted so that the transmitted ones will make their play-out deadlines". They evaluate some number of optimal selective frame discarding algorithms by employing dynamic programming techniques and heuristics.

Chapter 3

SELECTIVE FRAME DISCARDING (SFD)

Selective frame discarding is the name for the frame discarding filter that we have proposed. The frames are arriving into the quality adaptation buffer of the server, which is the term that we are using interchangeably with the term server output buffer. The frames are arriving at the frame rate of the video. For every frame arriving into this buffer, a decision is made whether to admit this frame, or to discard it. In order to make the decision the main concern is, if the frame is admitted, will it be able to reach the client before its play-out time expires? Furthermore, there is a hierarchical architecture for the video sequences. Some of the frames are more important than the other ones. Each frame's decision should also be effected by its importance in the video sequence. For this reason there needs to be a differentiation between frames. I and P frames are containing information necessary for the decoding of the frames that are dependent on them, so it is essential to protect such frames. Therefore, while B frames can be sacrificed easily, I and P frames should be admitted at the cost of losing some more B frames. In order to achieve differentiation, we assign priorities for each of the frames. A high priority frame is admitted even if there is a probability that it may not make its deadline. However, a low priority frame is not admitted if it will be delayed at the server more than a fraction of its left play-out time. Once the frame is admitted to the quality adaptation buffer, it is the transport protocol's responsibility to deliver this frame to the client. The client also buffers the content that arrives. It keeps an estimate for the duration of the content arrived and informs the server about this estimate. Therefore, the

server has an estimate of the time left for a frame to reach its destination before it is admitted.

3.1 Priority Assignment For MPEG Video Frames

An encoded video with standard MPEG-1 codec, has I, P and B frames existing in the stream. 'I' (intra) frames consist of the JPEG like encoded blocks and macro blocks. The 'P' frames are, in the most part, predicted frames from the preceeding I and the P frames. Up to some level, they also have Intra coded blocks. These blocks are generated because prediction procedure would not give out a clear result, or the prediction vectors will be so large that no gain from the bandwidth will be supplied. This situation arises when there is a scene change. The P frames corresponding to such an event will have higher size respect to other P frames. Since they carry more information and their sizes are larger they should be assigned as high priority. P frames contain both the Intra coded and P coded blocks. B frames, which are generated from both preceding and proceeding I and P frames, generally have predicted blocks. Since no frames depend on them, any error on a B frame will not propagate. However for P frames, the errors will propagate through the GOP until a new GOP starts and a new intra frame is displayed.

It is generally accepted that [45] Intra frames are the frames that are biggest in size. P frames which are next important, also having some intra blocks, has the next largest sizes. B frames have the smallest size among them.

Therefore, in order to determine the importance of a frame, it is possible to check the size of the frame. Even though this method may not be the best way to assign priorities, its implementation simplicity makes such a priority assignment mechanism a suitable choice. The frames having more than some number of Intra blocks will have larger sizes and since they are the important ones, we can define a threshold, for which the frames having larger sizes than this threshold, can be assumed to be important and marked as high priority. This is the method that we propose on the classification of the frames.

The packets that can survive by themselves become more important than the ones that are dependent on the success of the others in the case of insufficient

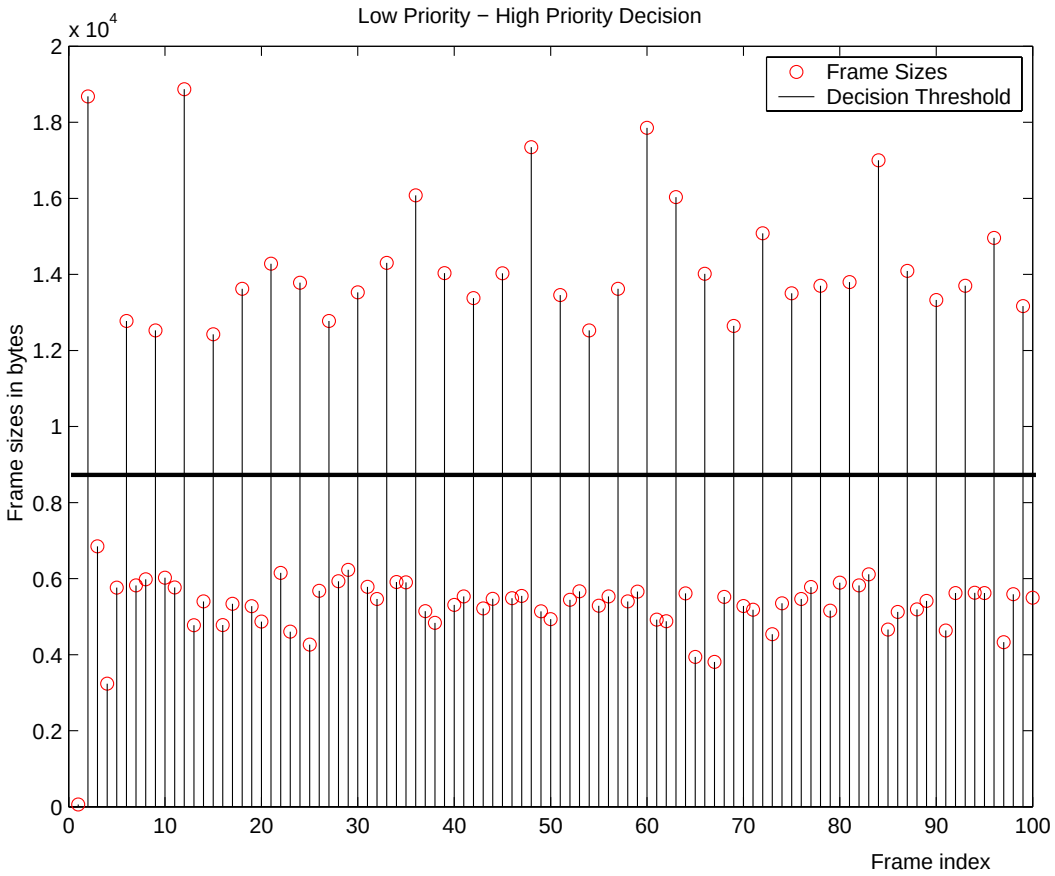

Figure 3.1: The frame sizes of a video stream

bandwidth. The frames that have smaller sizes contain less independent information. The Intra frames that are not very dependent on the success of any other frame have larger sizes.

In Figure 3.1, we can see that most of the frames are small sized, but the self dependent ones are in small numbers and in larger sizes. Here a threshold is assigned, and the ones above that threshold are marked as high priority. There could be more than one layer of separation, according to the sizes of the frames. In that case we could mark the largest ones as, priority 0, and the smaller ones as priority1, priority2, respective to their sizes. In our binary case, we mark the smaller ones as low priority.

3.2 Server Output Buffer

The server output buffer is actually the standard TCP output buffer, as BCC uses the underlying TCP architecture. Since TCP parameters are no longer used,

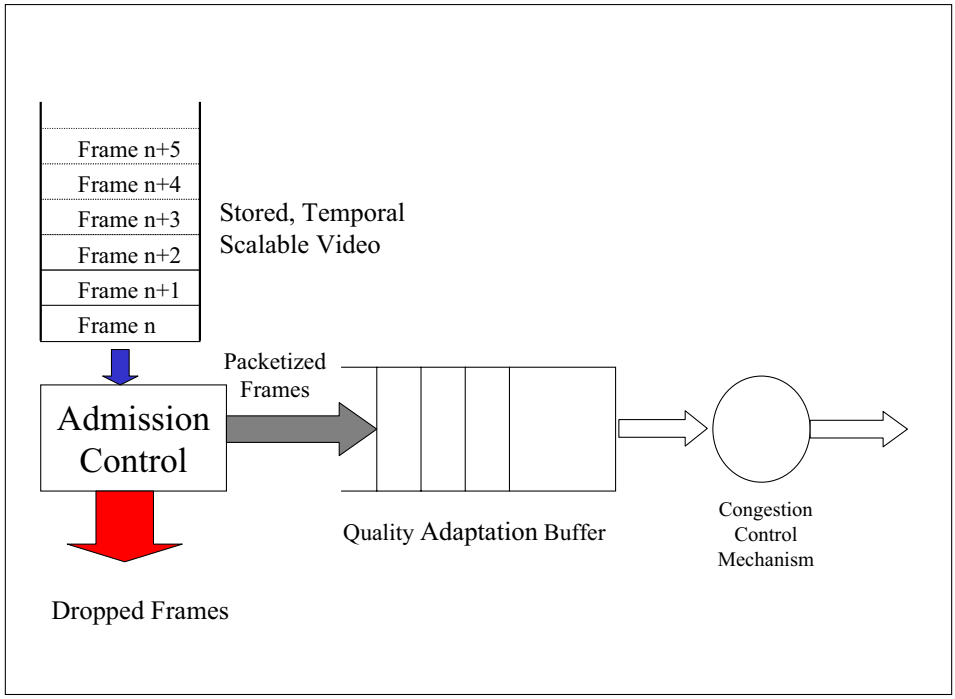

Figure 3.2: The server architecture and the output buffer

this buffer can be accepted as the server output buffer. For the system that is developed, this buffer is of finite length D seconds.

The congestion control mechanism governs the transmission of the packets residing in this buffer. The packets are not dropped out of the queue, once they are admitted. As the packets are transmitted copies of them are kept until they receive an acknowledgement. This buffer is fed by the output of the priority assignment sub-module and only drained by the Congestion Control Mechanism that is forwarding the TCP packets.

The frames, that are at the input of the buffer, are separated into smaller sizes if they are larger than the size of the Maximum Transmission Unit (MTU). These smaller packets are formed at the size of MTU. The remaining part which is less than the MTU, is sent as it is. In Figure 3.2, the situation is depicted. The frames having a size more than an MTU are chopped into packets at the size of MTU without actually changing their priority. The frames arrive into this buffer at the rate of the video itself. Exceeding this rate may be beneficial in the short term. However, injecting packets at a rate more than their drainage rate out of the system will cause the play-out buffer to overflow and unnecessary loss of packets.

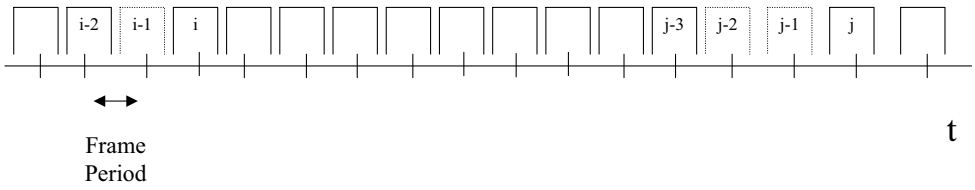

Figure 3.3: The frame sizes of a video stream

3.3 Client Play-Out Buffer

Client play-out buffer stores the necessary amount of content to keep the video uninterrupted during the periods that the transmission rate is very low. During the periods that the transmission rate is sufficient, this buffer is built-up again. It also suppresses the early or a little late arrival of the packets. Since the video frames are fragmented into smaller units in our system, they are reconstructed in the play-out buffer. Packets that belong to the same frame are combined according to their TCP sequence numbers. The drainage of the buffer is stopped if the number of packets in the buffer fall below a certain threshold. By employing a lower threshold, display of the frames still having packets on the flight will be delayed. As soon as the buffer occupancy exceeds this lower threshold, play-out will be resumed. This may provide more time for each frame to be constructed.

The frames are drained at the standard rate of the video. The missing frames are also assumed to be displayed and the next existing frame is displayed at its own deadline. In other words, the period for the missing frame is not filled with the next existing frame. The scene will be frozen during the time of the missing frames. The situation is given in Figure 3.3 where the frames i and $i-2$ are the ones that are in the play-out buffer. Instead of missing $i-1^{st}$ frame, the $i-2^{nd}$ frame is continued to be shown on the screen. A similar picture which will be less disturbing will be shown. For the case of j^{th} and $j-3^{rd}$ frames on the same stream, $j-3^{rd}$ will be displayed instead of missing $j-1^{st}$ and $j-2^{nd}$ frames.

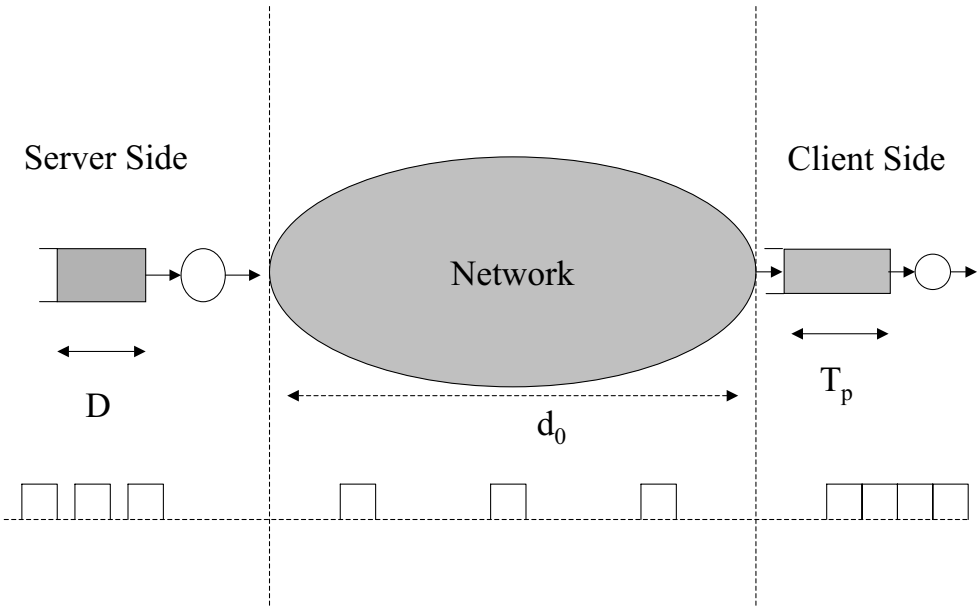

Figure 3.4: The end to end visualization of the system

3.3.1 Estimating the Latencies

A video streaming environment might be modelled with the queueing structure given in Figure 3.4. D is the delay estimate in the output buffer of the server. It is estimated with running average of the actual waiting time in the output queue of the server. The running average filter is $D^{k+1} = \beta \times D^k + (1 - \beta) \times D_{last}$, where D_{last} is the delay experienced in the buffer by the last packet drained from the buffer.

T_p is the estimated length of the play-out buffer in seconds. It is calculated as a running average $T_p^{k+1} = \alpha \times T_p^k + (1-\alpha) \times T_i$, where T_i is the instantaneous length of the play-out buffer in seconds. Actually, T_i is the period of time that will elapse from now until current frame at the end of the play-out buffer will be displayed. We assume that the information about T_p is available to the server. The value of T_p is updated every time a frame is drained from the buffer. This information is then fed back to the server, for the decision process. The feedback is provided by the piggy backing of this information on the TCP acknowledgments. The client instructs the server to start the algorithm. This instruction may also be carried by the acknowledgements. When this signal is set 'ON', the algorithms become

effective. The implementation details are out of the scope for the research that we have conducted.

The very first packet will be at the client without any delay at the server since the server is initially empty. It will be delayed in the network by d_0 seconds. Since there is an initial waiting time T' for building up the play-out buffer, the time that the first frame will be displayed is at $d_0 + T'$. Assuming that the delay in the network is small compared to T' in order for a frame to be played-out successfully, the following inequality should be satisfied ($D < T_p$). This inequality, in case it is satisfied, shows that the frame can be delayed up to T_p seconds at the server without violating its play-out at the client play-out buffer. But we note that only the estimates are available to the admission control mechanism.

3.4 Selective Frame Discard, A Heuristic Based Technique

First of all, we need to mention some of the motivating facts in order to explain the proposed algorithm more clearly. One of the facts is that there may be a mismatch between the video rate and the available bandwidth. This situation becomes problematic when the available bandwidth in the network is less than the rate of the video. In this case, it will take more time to transmit the whole video than the play-out duration of the video. If the video is sent at a rate more than the available bandwidth, there may be unpredictable losses. There is also an evident possibility that the network may become highly congested by such a behavior.

As a second fact, some frames of the video sequence are much more important than the other frames. The important frames should be treated as high priority while others are treated as low priority. To prevent the unpredictable loss of data during an insufficient bandwidth condition in the network, relatively unimportant frames should be discarded. Otherwise, the video might experience extensive delay. This discarding methodology may provide delivery of the more important frames of the video on time by sacrificing less important frames. SFDA (Selective Frame Discarding Algorithm) given in Table 3.1 addresses the listed concerns.

Then given a new high priority frame, it is admitted if $D < \gamma_{HP} \times T_p$ equation is satisfied. Following the lines of the previous section it may appear that γ_{HP}

Upon the arrival of each frame

if n ϵ High-Priority,

if $D < \gamma_{HP} \times T_p$
admit;
else
reject;

else if n ϵ Low-Priority,

if $D < \gamma_{LP} \times T_p$
admit;
else
reject;

Table 3.1: Selective Frame Discarding Algorithm (SFDA)

should be set to one. However, in this thesis, we set $\gamma_{HP} = \infty$ for two reasons. Firstly, the choice of γ_{HP} may unnecessarily discard certain high priority frames just because delay estimates are not sufficiently accurate. Moreover in our simulation studies, the available bandwidth to a single connection is larger than the bandwidth required for the base layer. We leave the discussion for the choice of γ_{HP} in the case of more general scenario for future research. On the other hand, we suggest to use a parameter $0 < \gamma_{LP} < 1$ in order to minimize the probability of discard for high priority frames.

3.4.1 Retransmissions

A packet will be retransmitted if it is not acknowledged. If retransmitted packet reaches the client before its due play-out time, it would not pose a problem for the continuity of the video. In the system being proposed, retransmissions are 'ON' since the TCP Reno is the underlying concept of Binomial Congestion Control and retransmissions are an integral part of the operation of TCP Reno. There are two types of retransmissions for the Reno; first type is triggered by the arrival of duplicate acks, second type is triggered by the timeouts.

The retransmissions triggered by the timeouts are generally caused by the loss of multiple packets in one round trip time. Caused by such a fact, they

indicate that our transmitted packets have been lost on large scales. So the decision to be made is; should one leave a gap in the video information and send the new data or re-send the missing part? Our system re-sends the data at the cost of increasing latency in output buffer of the server. However, we chose not to significantly modify the TCP-Reno code we have and in our proposed system, retransmissions are 'ON' as in the original TCP-Reno.

3.4.2 Implementation

The system is simulated using the ns v2 network simulator [9]. The simulator runs on a linux platform. The exact version that our implementation evaluated is the ns release 2.1b8a. The TCP implementation on this version of the ns includes various types of TCP and also BCC. However, the TCP implementations are generally based on the assumption that all the packet sizes are fixed. Therefore instead of using sequence numbers based on the bytes, the original implementors have chosen to use the packet counts as sequence numbers and window sizes. This property of the window and the sequence numbers are not appropriate for the video transmission simulations. Because a frame may be smaller than the size of an MTU, a window implementation based on the packet counts rather than bytes will treat this packet as a full MTU size packet. Even though there is a possibility of transmitting more packets, we will transmit a single packet for this case. Also the window increment for transmission of a smaller size packet will be the same as the window increment for a full size segment. In order to avoid this, we changed the ns TCP code so that the window updates are based on byte increments and decrements.

The trace files for the respective video files are created in a separate 'C' program. The ns simulator also creates a trace file that indicates which packets have arrived before their play-out time expire. By the use of the output trace the video is reconstructed by a separate 'C' program. The video file is then separately viewed by a media player.

Chapter 4

Numerical Results

Already existing streaming video protocols use UDP as a transport protocol. UDP has some obvious advantages such as stable rate, simplicity of usage and low protocol overhead. However, UDP does not employ congestion control. This causes the major data transmission protocol TCP to suffer from the unfair share of the bandwidth (in favor of UDP) in IP networks. As a major motivation for this thesis, we will present this unfair interaction at first place. Smoothness, which is a key requirement for video applications for various BCC parameters, will be evaluated. After suggesting the most suitable parameters, the TCP friendliness of such a scheme will also be verified.

Performance evaluation of our selective frame discarding mechanism and existing UDP type streaming will be evaluated on similar network conditions. The comparisons are based on the streaming of a temporal scalable MPEG encoded video. Since lesser waiting times to play video content is desirable, our system's performance for different buffering delays will also be reported.

4.1 Interactions Between Transport Protocols

In order to present the effect of unresponsive UDP flows on the ongoing TCP data traffic, we set a single bottleneck scenario as in Figure 4.1. Here 20 TCP connections from different sources share the same transmission path with UDP flows. Along the transmission path of each flow, all the links have sufficient bandwidth to carry video at its own bit rate except the bottleneck link. All of the traffic flows must pass through this bottleneck link to reach their final

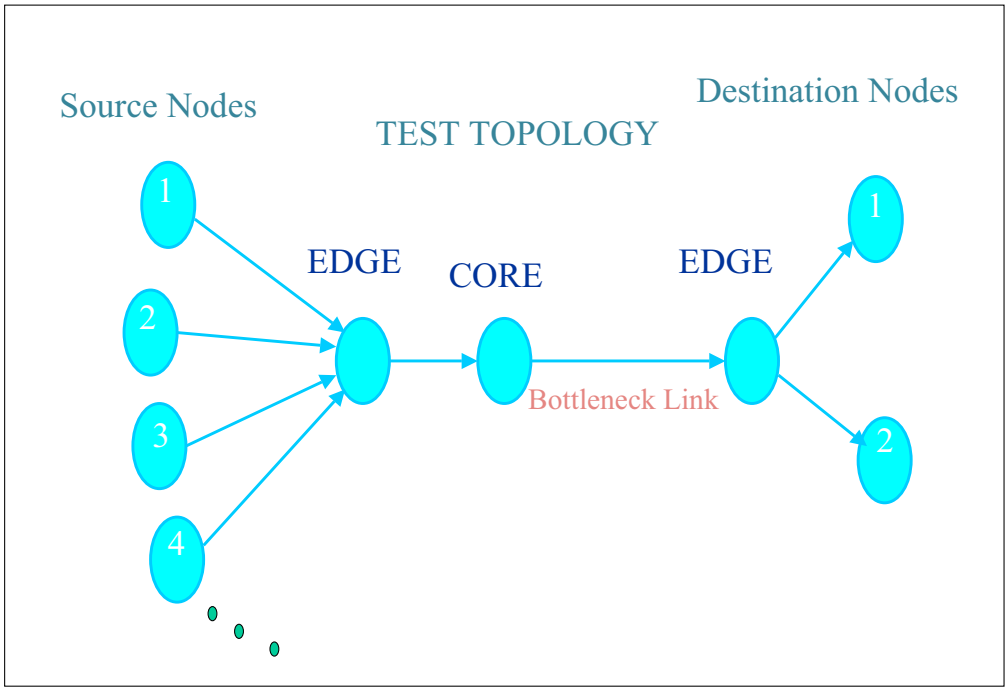

Figure 4.1: Topology of the network "Dumbbell"

destinations. By adjusting the bandwidth of this link, it is possible to simulate different loss levels over the Internet and different available bandwidth scenarios. For a lossless transmission, each connection should be provided with a rate at least matching to the rate of the video being streamed. The video that we are streaming has a rate of 2.0 Mbits/sec. By setting the capacity of the bottleneck link 2.0 Mbits times the number of sources there will be enough capacity for each of the flows, so a lossless transmission may be achieved.

A single bottleneck link network will simulate the interaction of similarly conditioned flows. Such a topology is called "dumbbell topology" in the literature [14]. Since the capacity of the links that connect each of the sources to their destinations is much higher than the bottleneck share of each flow, the loss events will take place only on the bottleneck link. The buffer management of this link is important since all loss events take place at the output interface of the core node. The buffer management method of this node is chosen to be Random Early Discard (RED). Since there is a single class of network traffic, in other words the network is best effort, no scheduling mechanisms are necessary or employed. The RED parameters for this node are chosen with the aim to prevent extensive losses of a single flow. The queue size is 200 packets, and the RED parameters are

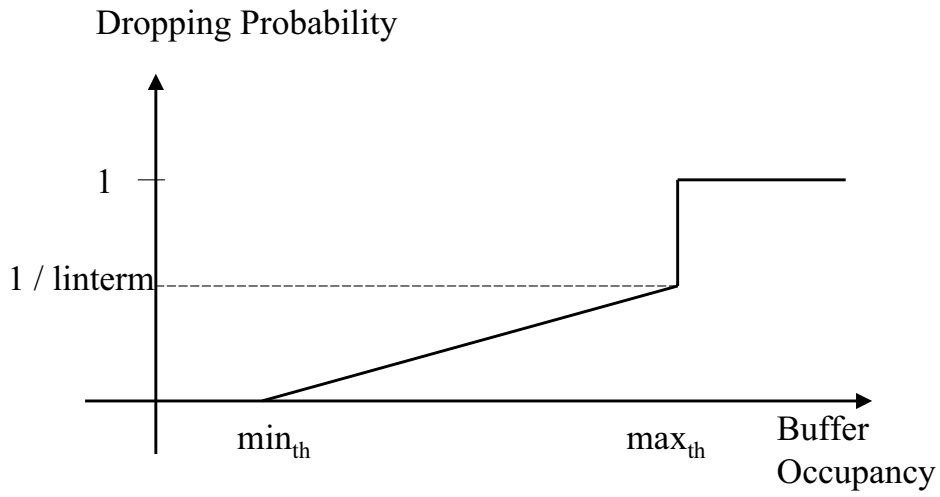

Figure 4.2: The RED dropping probabilities

chosen minimum threshold as 40 packets and maximum threshold as 195 packets. The linear term that determines the maximum dropping probability is 180, which means maximum dropping probability is 1/180. The dropping probabilities are given in Figure 4.2. By setting the bottleneck link capacity to 12 Mbits on this topology, we will investigate the interaction of TCP with UDP. In order to present the conjectured unfairness, 20 TCP sources and 9 UDP sources each having 1 Mbits/s rate are sequentially started. First, all TCP connections start their transmission. Twenty seconds after TCP sources start their transmission, the first UDP source starts streaming at 1 Mbits/s. After 30 seconds have passed, the second UDP source starts transmission. All the remaining UDP sources start their transmissions after 30 seconds following each other sequentially. However, the last source begins its transmission after 10 seconds that the previous one has started. The total successfully received bit-rates by the clients of the UDP sources are presented in Figure 4.3. The share of the network capacity that UDP uses should be 1/21 of the available rate if the protocols were interacting fairly. However, it successfully accomplishes a rate around 1 Mbits/s. The excess rate is presented to the TCP flows, which reduce their rates in order to prevent congestion. However, UDP does not take any action against congestion, and continues to stream its content with its dictated rate. As new UDP sources

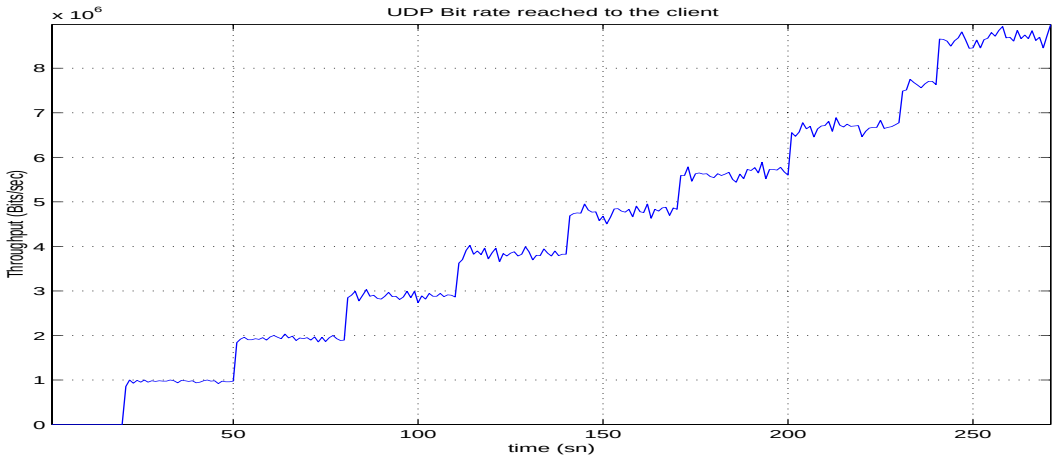

Figure 4.3: Total successfully received bit-rate of UDP flows

arrive, unfair sharing of the network resources becomes more evident. After the arrival of all 9 UDP sources, they use roughly the 3/4 of the available bandwidth, while the remaining 20 TCP sources try to achieve their transmission through the remaining 3 Mbits/s bandwidth. For each TCP source, the available rate becomes 150 Kbits/s. Under these conditions, throughput of a single TCP source is given in Figure 4.4.

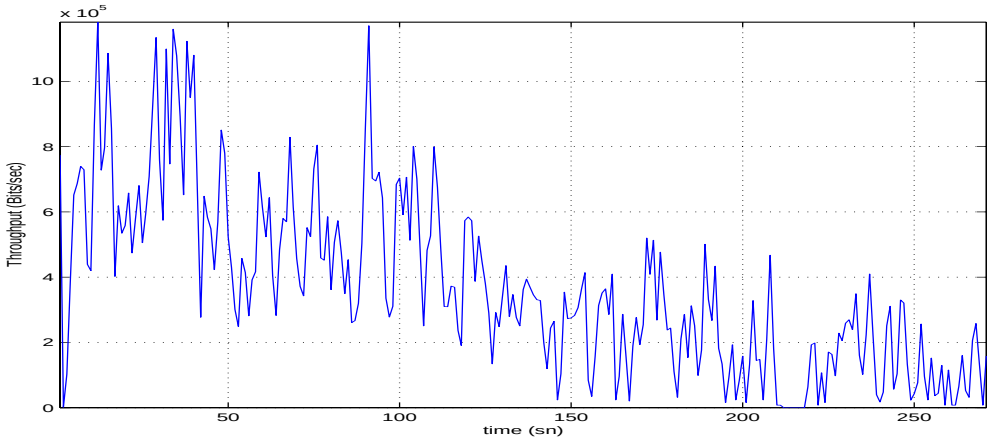

Figure 4.4: Total successfully received bit-rate of a TCP flow

Such a situation leads to the extreme suffering of data connection over the Internet. As a solution, all the sources can be enforced to use TCP. But such an enforcement will lead to the suffering of the video streaming applications because of the oscillatory behavior of TCP.

4.2 Smoothness

In this thesis, we use the term "smoothness" to describe throughput behavior of a transmission. A smooth algorithm provides transmission rate with a certain mean and its variance around this mean is minimal. In this section, smoothness of different parameter settings for the binomial congestion control mechanism are compared through simulations. We observe the congestion windows of the algorithms, and their respective throughputs. There are two proposed parameter sets for BCC [4]. These are called SQRT and INV. k and l are the window update parameters as used in Equation 2.2 and 2.3. The square root (SQRT) set has values $k = l = 0.5$. The inverse (INV) set has the values $k = 1$ and $l = 0$. Since TCP friendliness of the algorithm depends on the value of sum $k + l$, any value of k and l that satisfies equation $k + l = 1$ is accepted as TCP-friendly. However, α and β values are left to be freely chosen. The choice of α parameter dictates the increment of the window size after a window amount of data has been acknowledged to be transferred. The β is the decrement constant and we chose that constant same as TCP.

We will consider the values $k = 1$, $k = 0.5$, $k = 0.2$ and $k = 0$. We call the set $k = 0.2$ as BCC-02. These values correspond to INV, SQRT, BCC-02 and TCP, respectively. In scenario we have considered, 20 sources are streamed over the same dumbbell topology. The capacity of the bottleneck link is set as 32Mbits/sec. According to a fair sharing of this link each source should get a fair share of 1.6 Mbits/sec of a link capacity. We measure the throughput of the connection by adding up the amount of data arrived in a second period. The presented results are for a single connection, except the comparison of the throughput of the BCC and TCP sources. In this comparison total throughput of all the sources are given.

4.2.1 Inverse BCC Parameters

The first parameter set that is going to be investigated is the inverse parameter set, which has the inverse characteristics of TCP value set. It has window increment parameters as $k = 1$ and $\alpha = 1 \times MSS$, where MSS is the maximum segment size. The window decrement parameters are $l = 0$ and $\beta = 0.5$.

In such a situation, the window size is decremented slowly in response to congestion and it is also incremented slowly for probing the available bandwidth.

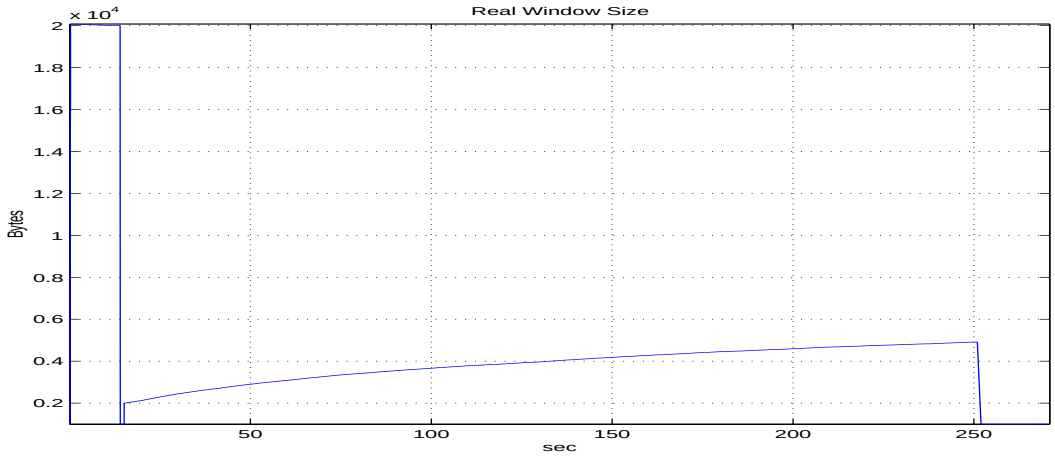

Figure 4.5: Window size variations of Inverse BCC flow

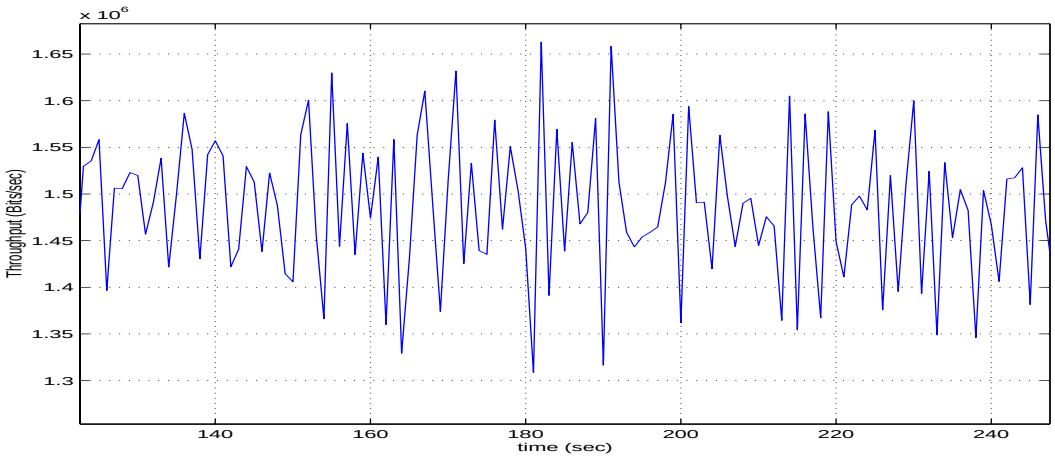

Figure 4.6: The actual throughput of single Inverse BCC flow

Therefore, the adaptability of this scheme is sacrificed for the smoothness of the rate. Its lack of adaptability causes this algorithm to experience timeouts. The window size variation over the transmission period is given in Figure 4.5. Around 15^{th} second of the simulation, it suffers a timeout. After this period it never achieves its previous rate of transmission. Its throughput is presented in the Figure 4.6.

4.2.2 Square Root BCC Parameters

We will consider two different versions of this parameter set. The first set uses the size of one packet increment for the successful transmission of a window full data if k was set to be 0. However the value of k is still 0.5 for the SQRT parameter. The second uses a quarter full packet size increment. In this section, after we

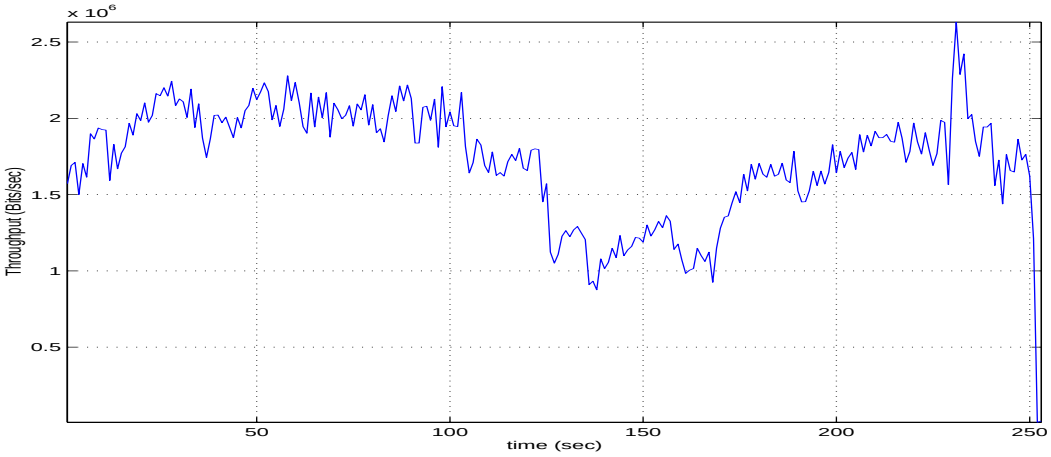

Figure 4.7: Window variations of SQRT BCC

present and discuss the results of the one packet increase, we will conclude that it may be too aggressively probing for the available rate. Therefore, we propose that alternative quarter increments may give better results.

For the one packet increment scheme, as the window size is presented in Figure 4.7, there are many timeouts. A timeout is caused for a BCC scheme as a result of multiple dropped packets in single round-trip time. These timeouts may result in, delaying of the content over one or more seconds. During this period, aggressive probing mechanisms may occupy all the bandwidth that was previously used by the SQRT BCC. During the period between 120^{th} and 140^{th} seconds there is such a problem, the scheme experiences a series of timeouts. Its throughput is seriously affected, which can be observed through Figure 4.8. In this case all 20 sources are using the same mechanisms. The aim is to investigate their interaction with each other, which is not yet a well explored area. As all the sources use the same slowly responsive algorithm, regaining the old rate requires some time. After a timeout, slow start threshold is also lowered, therefore, it is not possible to regain the same available bandwidth using the slow start mechanisms.

Perhaps, if the algorithm attacks for more rate less aggressively, the congestion that is caused by the total probing experiments may not become very serious, and congestion may result in dropping of less packets in a single RTT. Therefore, we may try to decrement the α parameter which will end up in less aggressive probing for the available bandwidth. However, since the $k + l = 1$ condition is still satisfied, we don't need to be concerned about the TCP friendliness. Actually, probing experiments are done less aggressively, which will obviously lead to better TCP friendliness.

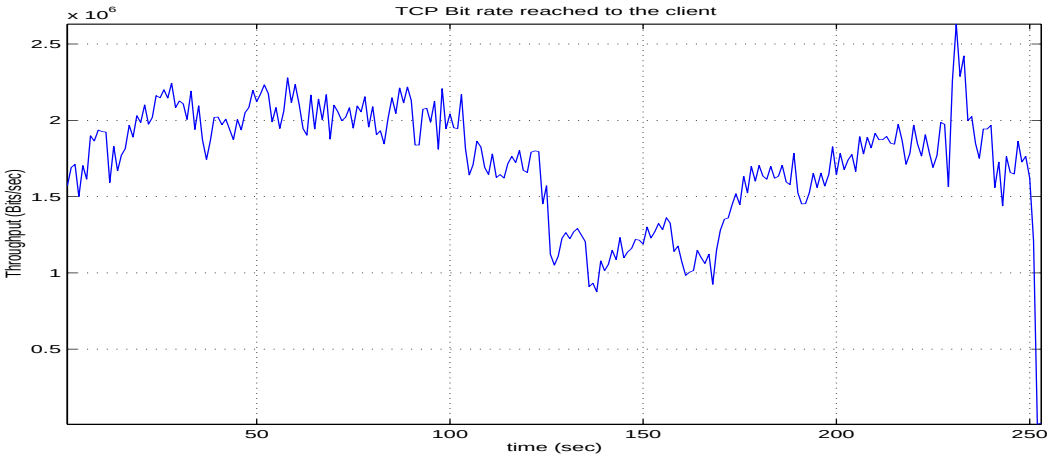

Figure 4.8: Throughput of SQRT BCC

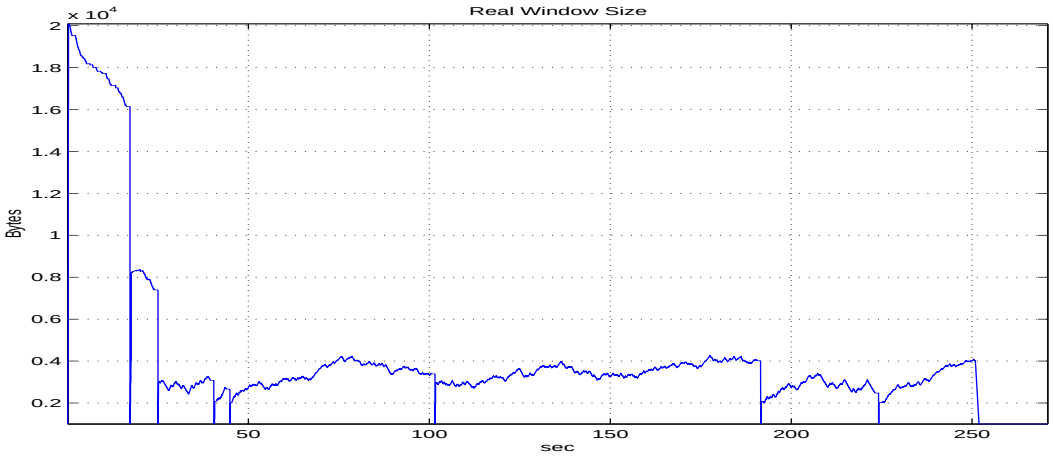

Figure 4.9: Window size variations of SQRT BCC for quarter packet increments

The value that has been chosen is $\alpha = 0.25$, which corresponds to a window increment of a quarter full size packet, after a successful window-full transmission if k was set to be 0. The resulting window variation can be observed in Figure 4.9, and its respective throughput is visible through Figure 4.10. Even though the number of timeouts are decreased, there are still timeouts. These may result in unfair sharing of bandwidth. While some sources are not transmitting, others will increase their rates more than the fair share they would take. On the worst case, such a situation will lead to unfair allocation of sources.

4.2.3 $k = 0.2, l = 0.8$ BCC Set (BCC-02)

For the sake of a healthy transmission, it is necessary to prevent time outs as much as possible. In the literature [20], the parameter set that the BCC is at

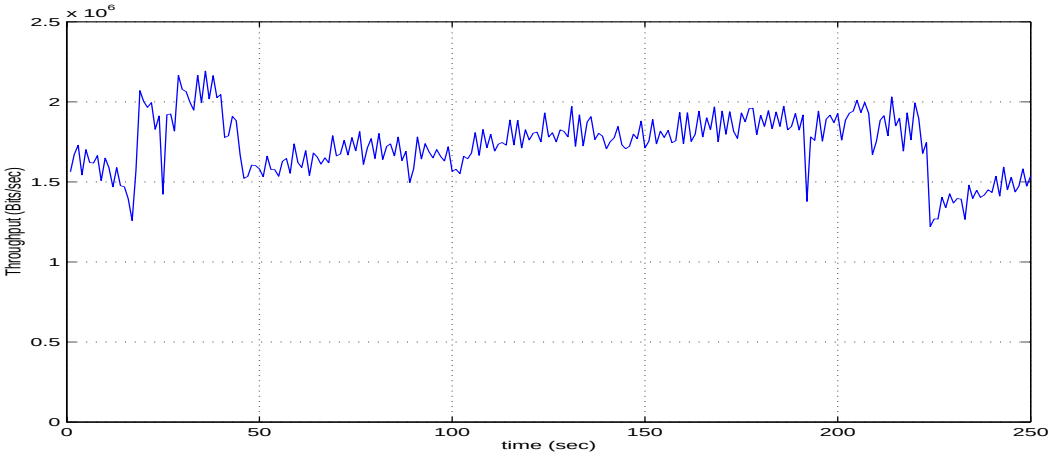

Figure 4.10: Throughput of SQRT BCC for quarter packet increments

most fairly interacting with each other is found to be in the set $k = [0, 0.2]$. Since both smoothness and fairness are important for a streaming protocol, values in this set should also be considered.

To be as fair as possible, while adapting rate as smooth as possible, the $k = 0.2$ should be considered as it is the extreme smooth parameter that is reported to be fair. The value of α is chosen to be 0.25, which results in more smoothness as also been observed in the SQRT. The window size variations for one of the 20 sources making transmission on a "dumbbell" topology can be seen in Figure 4.11. As we can observe, there are no timeouts occurred in this parameter setting. Actually, we can conclude that this scheme has relatively better adaptability for the network conditions. Even though it is adaptable, the throughput is provided on a smoother basis, this can be observed from Figure 4.12. As it is evident, it is able to smoothly adjust its rate over the transmission period.

4.2.4 The TCP Set

Is the TCP really oscillatory? We can observe TCP over the same network scenario. The window size variations are observable in Figure 4.13, and its respective throughput can be seen in Figure 4.14.

As it can be observed, the window size variations are so dense that it is not possible to distinguish any one of them. As we consider the throughput, it oscillates between 0.3 Mbits/sec and 2.5 Mbit/sec. So such an oscillatory

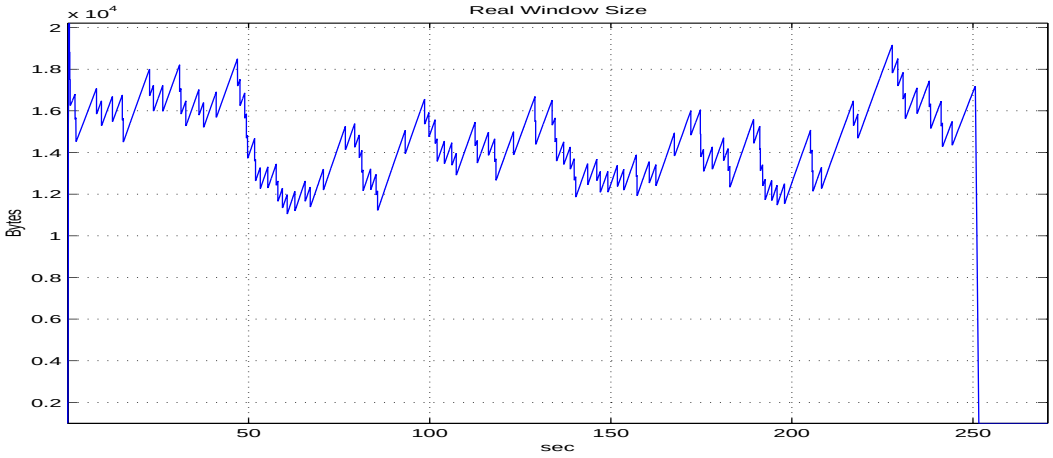

Figure 4.11: Window size variations of k=0.2 BCC for quarter packet increments

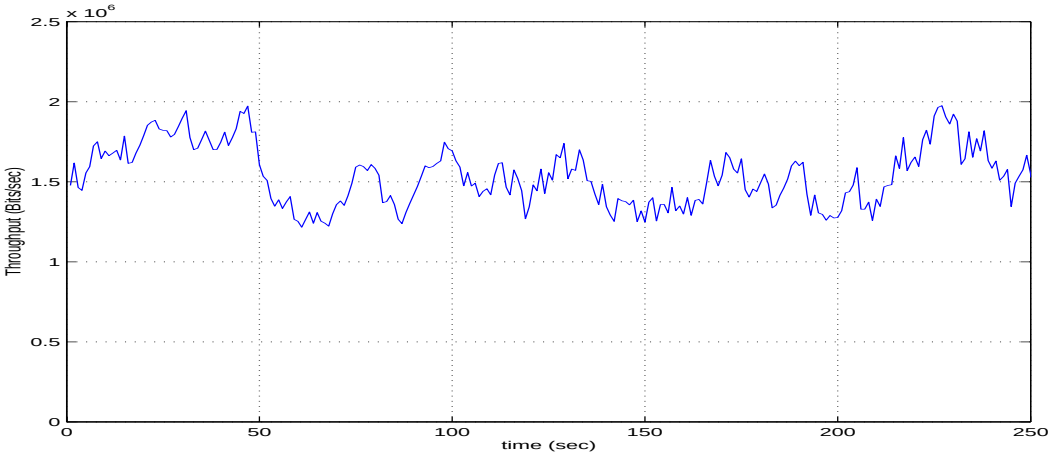

Figure 4.12: Throughput of k=0.2 BCC for quarter packet increments

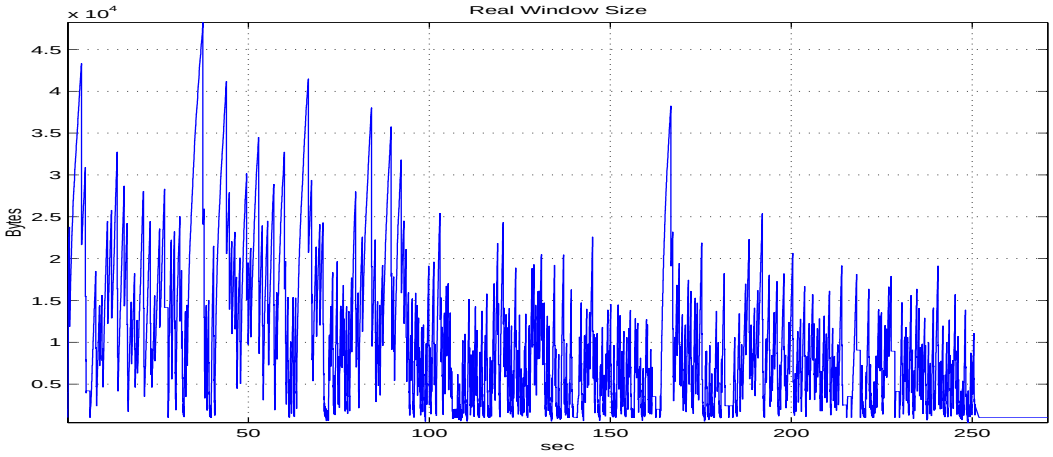

Figure 4.13: Window size variations of TCP

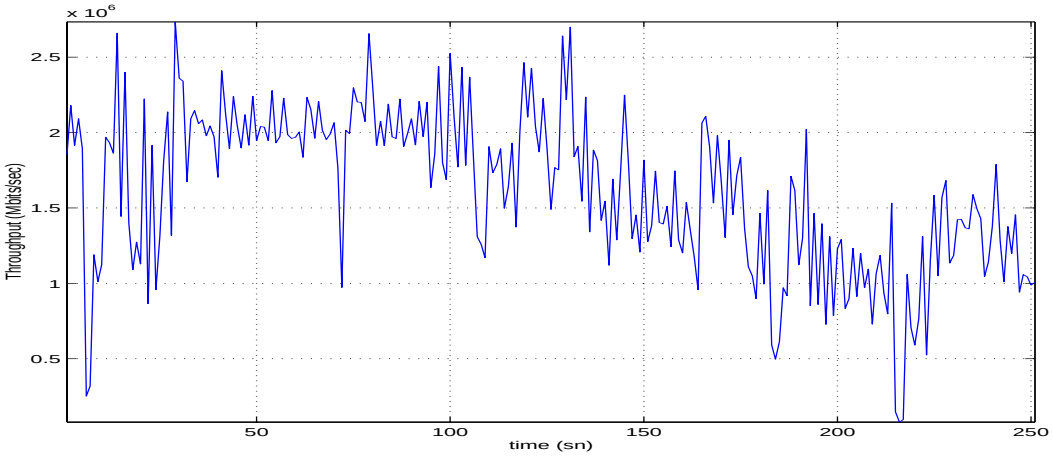

Figure 4.14: Throughput of TCP

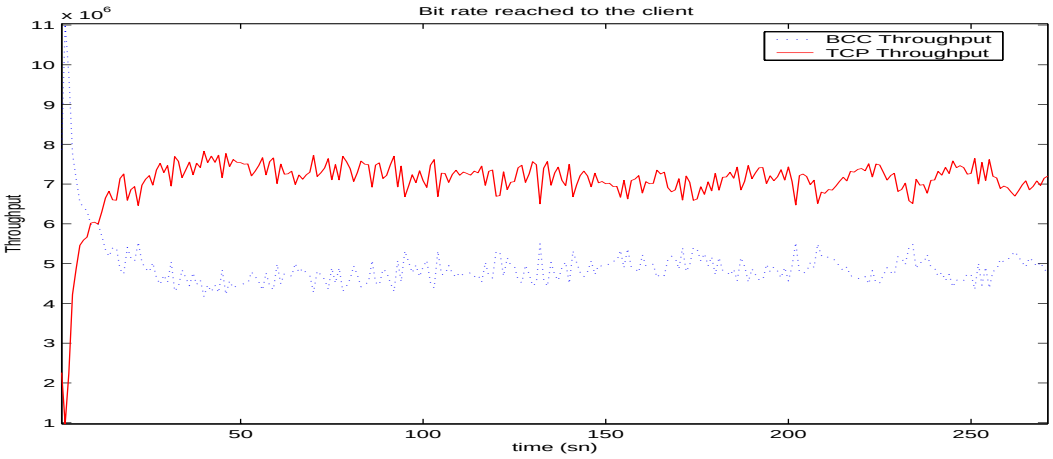

Figure 4.15: Total throughput of BCC k=0.2 and TCP sources

behavior of TCP will not result in desired smooth variations of a video streaming application.

4.2.5 Interactions with TCP

Since TCP is the major data carrier, its interaction with the parameter set, which we have found to be best among the others, is a question that needs to be answered. Again for the dumbbell topology of Figure 4.1, we consider the interaction of 10 TCP sources with 10 BCC k=0.2 sources. The result presented is the sum of all TCP's throughput in solid, and sum of all BCC's throughput in dashed lines. From Figure 4.15 we can observe that the total of throughput of BCC flows is less than that of TCP. As this figure indicates, our algorithm is TCP friendly. The BCC flows seem to be taking about 10% less than the fair

share of the bandwidth, however this is the price that needs to be paid for the smoothness. The results in the previous work done are also in this direction. Work done by Bansal et. al. [14] concludes that such a loss of throughput does not prevent them to be deployed since they do not take throughput away from the TCP connections.

4.3 Video Streaming Using the BCC

Throughout this section we will be presenting the results of our video streaming system that employs prioritization of packets according to their relative importance. We will present two different selective frame discarding systems, which are delay based and the length based schemes. Delay based system has been presented in the section 3.4. The length based system constraint is simply the quality adaptation buffer length in packets. A frame is not accepted if the buffer length is above some threshold. This threshold is set to be 100 packets. Since mean frame size is 7638 bytes, which corresponds to 8 packets, this buffer can take around 12 frames. 12 frames corresponds to a half seconds of video content. For the delay based system, if the packets already in the quality adaptation buffer are waiting more than a fraction of their left time towards their play-out deadlines, no more packet is admitted to the buffer. Furthermore, we will also consider the streaming case using UDP.

We will present the results of the above schemes for various network conditions, namely for the cases of extremely insufficient to sufficient bandwidth situation. For this purpose for the topology in Figure 4.1, we will consider the cases of 20 BCC sources, sharing the bottleneck of 24 Mbits/sec, 32 Mbits/sec and 40Mbits/sec. Also a multi-bottleneck scenario will be presented.

4.3.1 Sharing the 24 Mbits/sec bottleneck

In this scenario, each source will receive a fair share of the bandwidth, which is 1.2 Mbits/sec. Such a bandwidth share is above the HP layer rate requirement. At this phase, we will present results for delay based scheme. First for this network scenario, the server buffer latency and the effective threshold for admission will be given. This threshold we propose is $0.1 \times T_p$ i.e. $\gamma_{LP} = 0.1$. The T_p is the feedback of the estimated play-out buffer length. In Figure 4.16, the comparison

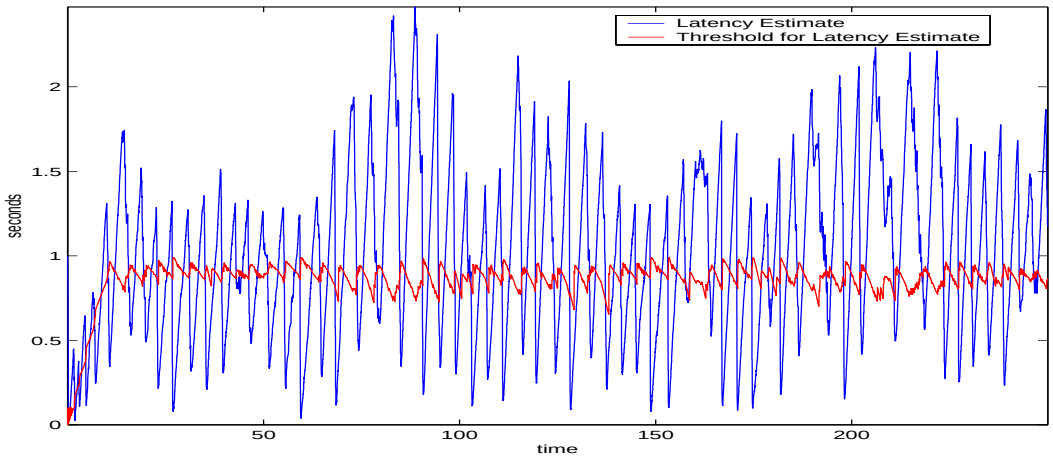

Figure 4.16: Comparison of the latency in server buffer vs admission threshold

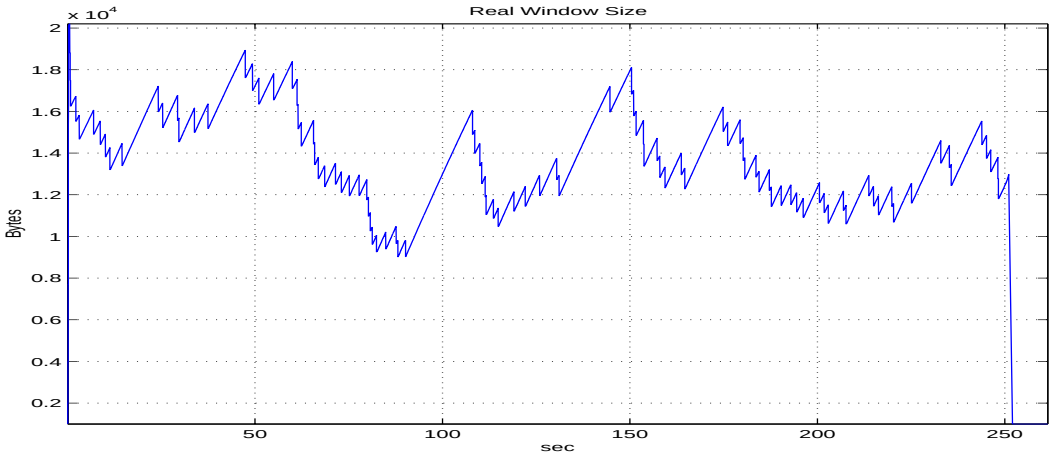

Figure 4.17: Window size variation of BCC k=0.2 over 1.2 Mbits/sec fair share bottleneck

of the latency in server buffer with the admission threshold is given. From this figure, we observe that the threshold value limits the latency delay to an upper bound, that does not exceed or even get close to the play-out buffer length in time. Therefore, system will not be further delaying the packets that will not be able to make their play-out time and will be discarding them, whenever the threshold is exceeded. Since the Intra frames and some important P frames will also be protected, low layer of the temporal scalable video content will be preserved. As the rate is low, latency continues to increase since the clock is ticking. After already highly delayed packets are sent the latency starts decreasing. As it goes under the threshold dictated by play-out deadlines all the frames are again admitted. The advantage of such a scheme is that, it responds to the changing conditions rapidly. The window size variation for this transmission is as in Figure 4.17.

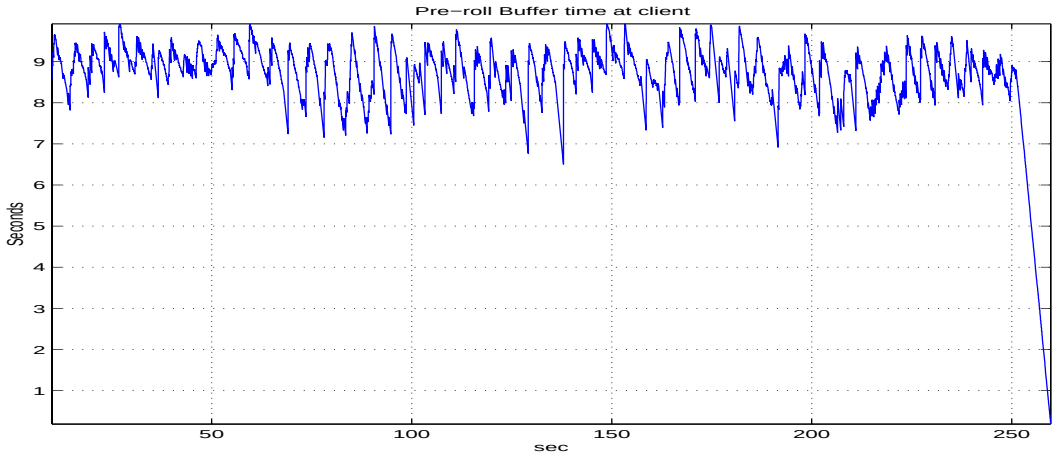

Figure 4.18: Play-out duration of BCC k=0.2 over 1.2 Mbits/sec fair share bottleneck

The resulting play-out duration is presented in Figure 4.18. The play-out duration of the video has some oscillations, however these oscillations are not critically high. Play-out duration is maintained at the level of initial buffering period. This period is chosen to be 10 seconds, different values of this may also be considered according to the conditions of the network that the video will be streamed over.

An alternative to delay based approach might be the length based approach, which indeed relies on the same principle. As the transmission rate decreases, output buffer length will increase. The server output buffer occupancy in such a case for the same topology will be of the form in Figure 4.19 for the window size variations in Figure 4.20. This methodology may experience a weakness in maintaining play-out buffer duration when the network has highly oscillatory behavior. However, for the cases presented here we do not examine such a situation. Even though the level of the play-out buffer is much more steady, the resultant video form is not as pleasable as the one that has been achieved with the video form of delay based system. Figure 4.21 presents the resultant play-out duration over the transmission period.

In such a bandwidth sharing situation, UDP suffers unpredictable losses in the network. These losses are generally because of the bigger sized frames. This information is based on the observations of the visual simulation interface of the ns simulator. They are divided into more number of packets and also they are streamed in a bursty manner. These cause sudden bursts in the network traffic, the packets belonging to higher sized I and P frames are more likely to be dropped. The compared results of all the mechanisms are presented in Table 4.1.

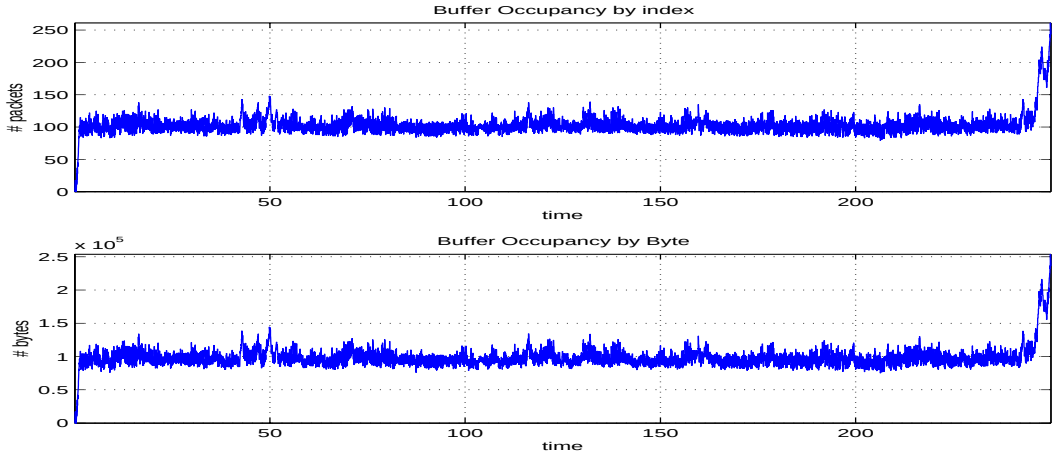

Figure 4.19: Length based SFD output buffer occupancy of BCC k=0.2 over 1.2 Mbits/sec fair share bottleneck

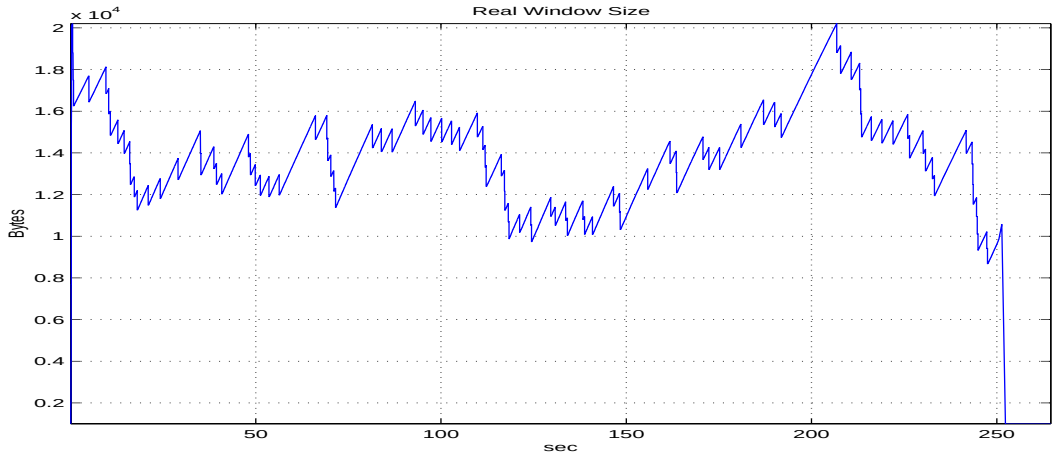

Figure 4.20: Window variations of BCC that length based SFD is employed k=0.2 over 1.2 Mbits/sec fair share bottleneck

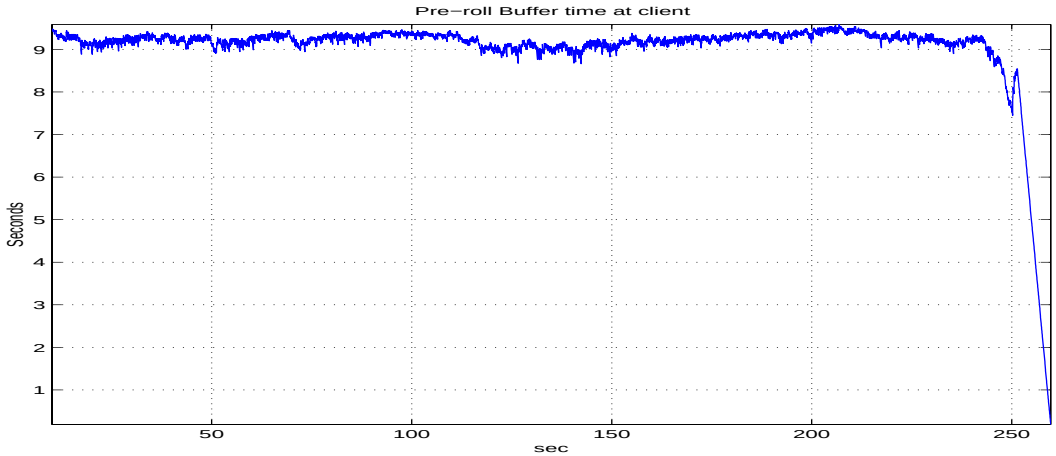

Figure 4.21: Play-out buffer duration of BCC that length based SFD is employed k=0.2 over 1.2 Mbits/sec fair share bottleneck

Total HP frames of the Video	597						
Total LP frames of the video	6069	E2 24 Mbit Shared by 20 Connections					
Total HP packets of the video	15442	Length Based		Delay Based		UDP	
Total LP packets of the video	52246	count	percent	count	percent	count	percent
HP frames Dropped at the Server		0	100%	0	100%	0	0
LP frames Dropped at the Server		29334	56%	29653	56%	0	0
HP packets lost during transmission		0	0	0	0	5252	34.01%
LP packets lost during transmission		5	0	6	0	18149	34.74%
HP packets rejected at the client		0	0	0	0	0	0
LP frames rejected at the client		0	0	0	0	0	0
Actual rate taken by each connection		1.162 Mbps		1.150 Mbps		1.200 Mbps	

Table 4.1: Loss based statistics of different schemes over 1.2Mbits/sec available bandwidth channels

As the table indicates UDP suffers randomized loss over the transmission path while the BCC streaming schemes streams the content by dropping the frames according to their relative importance in the video sequence. Furthermore UDP stream's loss rate seems higher than the ones occurring for the SFD schemes. This is also because of the bursty behavior of UDP traffic. Even though the TCP traffic is known to be creating bursty traffic, since its bandwidth does not exceed the available rate extensively, losses due to burstiness do not seem to be evident.

4.3.2 Sharing the 32 Mbits/sec bottleneck

For the 32 Mbits/sec bottleneck, each source will have its share of 1.6 Mbits/sec link capacity. The loss conditions will not be that severe for the video. Again as in the 24 Mbits bottleneck scenario, the delay based scheme will be presented at first. Later on the results for the length based scheme will also be presented.

The delay of the server output buffer can be observed in Figure 4.22.

Even though the buffer latency is oscillatory, it is bounded by 1.6 seconds. Therefore the packets in the buffer are forwarded after at most 1.6 seconds. This will prevent the extensive delaying of admitted packets. The window size variation in this situation is visible through Figure 4.23. The resultant play-out buffer length is presented in Figure 4.24.

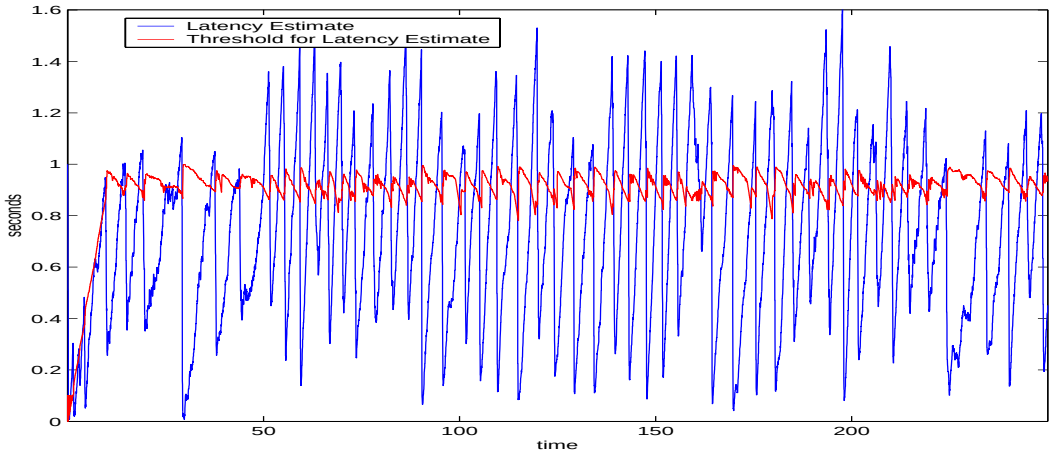

Figure 4.22: Comparison of the latency in server buffer vs admission threshold

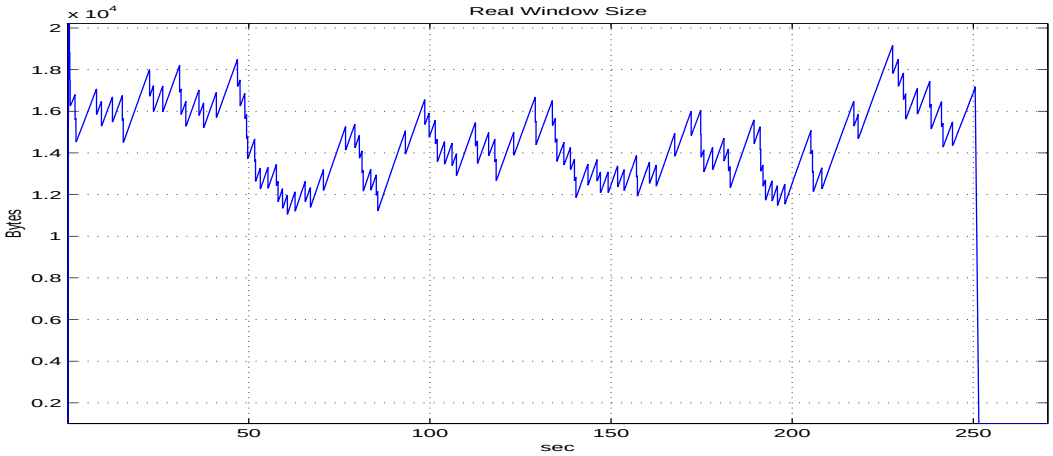

Figure 4.23: Window size variations of BCC that delay based SFD is employed k=0.2 over 1.6 Mbits/sec fair share bottleneck

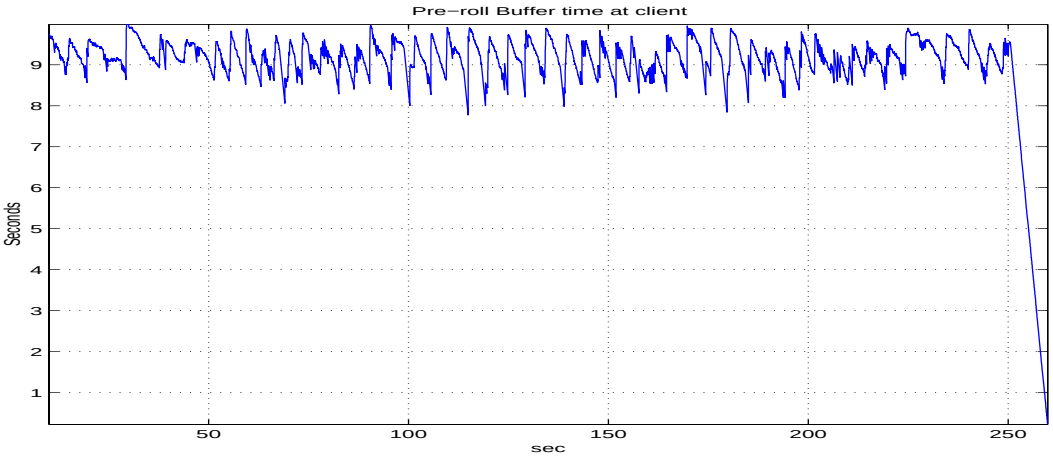

Figure 4.24: Play-out duration of BCC delay based SFD k=0.2 over 1.6 Mbits/sec fair share bottleneck

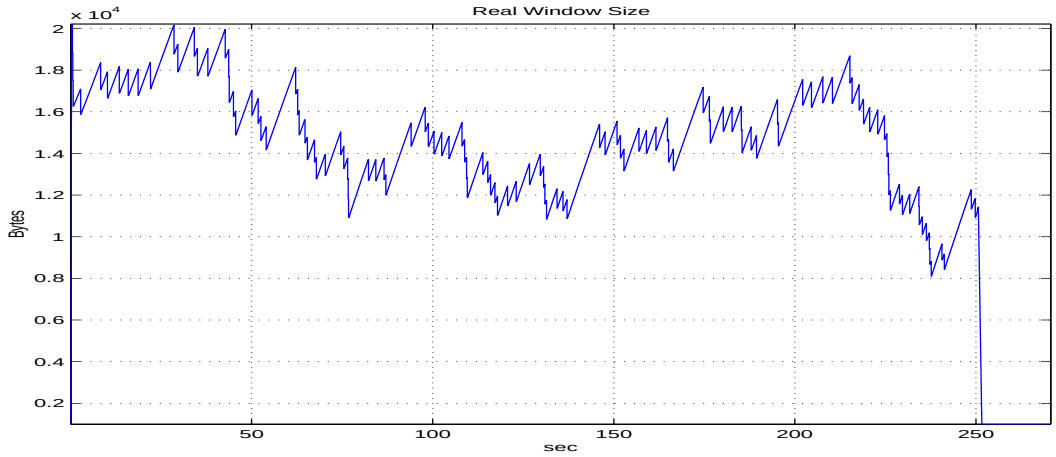

Figure 4.25: Window size variations of BCC that length based SFD is employed k=0.2 over 1.6 Mbits/sec fair share bottleneck

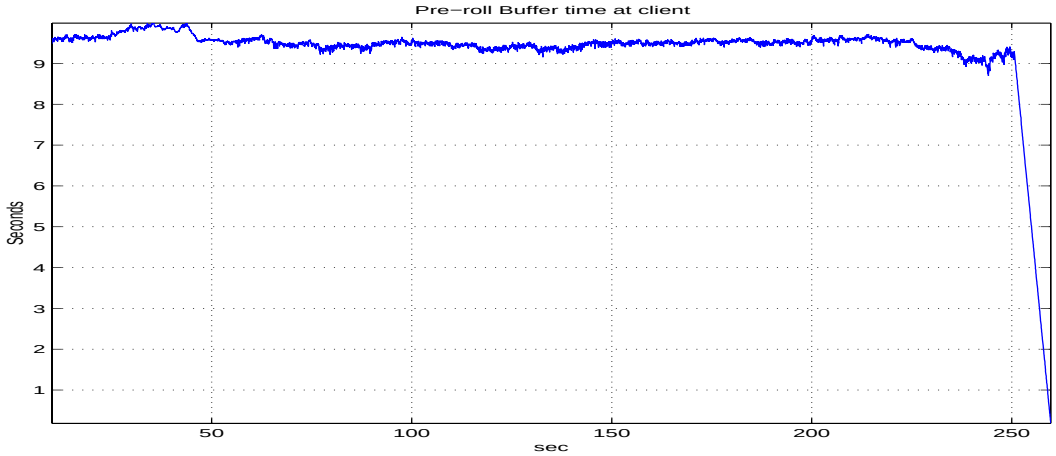

Figure 4.26: Play-out buffer duration of BCC that length based SFD is employed k=0.2 over 1.6 Mbits/sec fair share bottleneck

The length based SFD, for the transmission that has the window size variations as in Figure 4.25, achieves the play-out duration as in Figure 4.26. The output buffer length variation is as in Figure 4.27.

As the play-out buffer durations indicate, both the length based SFD and the delay based SFD achieve a steady buffering level. They keep around 9 seconds of content in reserve that will help them in case of a transmission off period. When two approaches are compared, they differ from each other by the content of the video they provide. These two videos are evaluated with a subjective video grading, namely Mean Opinion Score (MOS), in which the audience has been asked to grade the video over a scale of 6, as they just watched it and after they watched whole bunch of the results obtained with different schemes. Content has been viewed with Windows Media Player version 7.01.00.3055. The

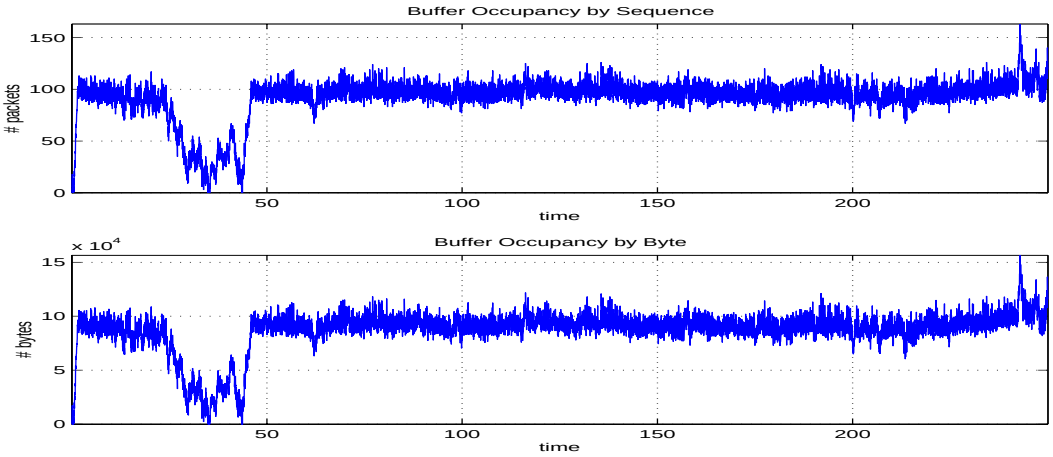

Figure 4.27: Length based SFD output buffer occupancy of BCC k=0.2 over 1.6 Mbits/sec fair share bottleneck

video content that has been achieved with the delay based SFD received a MOS of 2.725 out of 6, whereas the length based video received a MOS of 2.125 out of 6. The video streamed using INV parameter in combination with the delay based SFD in the same survey received the MOS value 2.025 out of 6. In order to verify the validity of the results, delay based SFD with BCC-02 under similiar network conditions has been re-evaluated. The result obtained was again 2.425 out of 6. In this survey, video simulated to be lossless transferred through our system received a MOS of 4.6 out of 6.

The statistics about both videos are reported in Table 4.2. For the loss rates reported in the table, the UDP video streaming received a MOS score of 1.5714/6. UDP performs poorly because it does not determine which packet will be lost. So the loss is suffered on a probabilistic basis, and most important frames will be lost most likely because of their sizes, and burst of traffic they cause. As we can observe from the table even though the bandwidth that has been taken by the length based algorithm is roughly a 0.05 Mbits higher, the achieved quality is found to be better for the delay based SFD by the audience. The UDP was the worst performing, since there was only one or two frames found viewable by the player so the frame rate was pretty low.

4.3.3 Sharing the 40 Mbits/sec bottleneck

In this scenario we again observe the video content that has been streamed over a bottleneck link shared by 20 identical BCC sources. This time the bottleneck link capacity is set as 40 Mbits/sec allowing a fair share of 2.0 Mbits/sec for each

Total HP frames of the Video	597						
Total LP frames of the video	6069	32 Mbit Shared by 20 Connections					
Total HP packets of the video	15442	Length Based		Delay Based		UDP	
Total LP packets of the video	52246	count	percent	count	percent	count	percent
HP frames Dropped at the Server		0	0	0	0	0	0
LP frames Dropped at the Server		14997	28.7	16609	31.7	0	0
HP packets lost during transmission		1	0	1	0	2670	17.30%
LP packets lost during transmission		8	0	3	0	10134	19.40%
HP packets rejected at the client		0	0	0	0	0	0
LP frames rejected at the client		0	0	0	0	0	0
Actual rate taken by each connection		1.5918 Mbps		1.54282 Mbps		1.600 Mbps	

Table 4.2: Loss based statistics of different schemes over 1.6 Mbits/sec available bandwidth channels

source. We again compare the same video content that has been streamed over the channel using delay based SFD, length based SFD, and UDP.

For this network conditions, we will evaluate the situation of the delay based SFD first, next we will evaluate the length based SFD, and finally compare their performances using their loss rates.

The quality adaptation buffer or the output buffer latency is presented in Figure 4.28. The window size variation throughout this transmission is presented in Figure 4.29. As we can observe from the latency sketch, since the line rate is very much close to the video rate, the arriving frames are sent without delaying them, therefore no frames have been discarded. Thus the delay based system also achieves the objective of keeping the content intact by not creating a burst that the network cannot withstand. Since the content is not delayed and discarded, the play-out buffer is kept at a certain level. The duration of the play-out buffer can be visualized over the Figure 4.30.

The conditions were not as good as delay based SFD for the length based scheme. It slightly suffers from the rate that the video was encoded. However, by discarding some of the low priority content it manages to match the rate of the video to the available bandwidth. The window size variation of the transmission is available in Figure 4.31. Quality adaptation buffer occupancy of the video is available in Figure 4.32, and finally the play-out buffer duration is given in Figure 4.33.

As the rate of channel falls below the video rate, the occupancy of the buffer increases and over the threshold of 100 packets, frames start to be discarded.

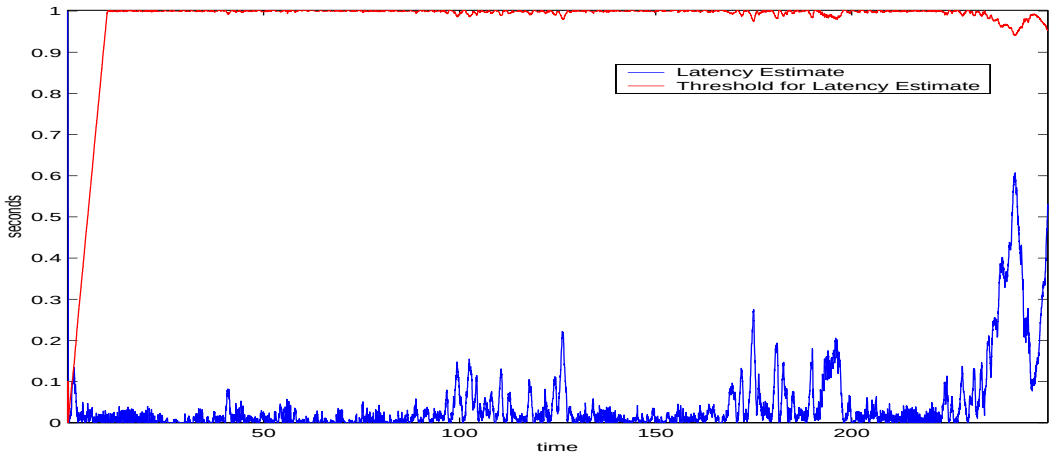

Figure 4.28: Comparison of the latency in server buffer vs. admission threshold

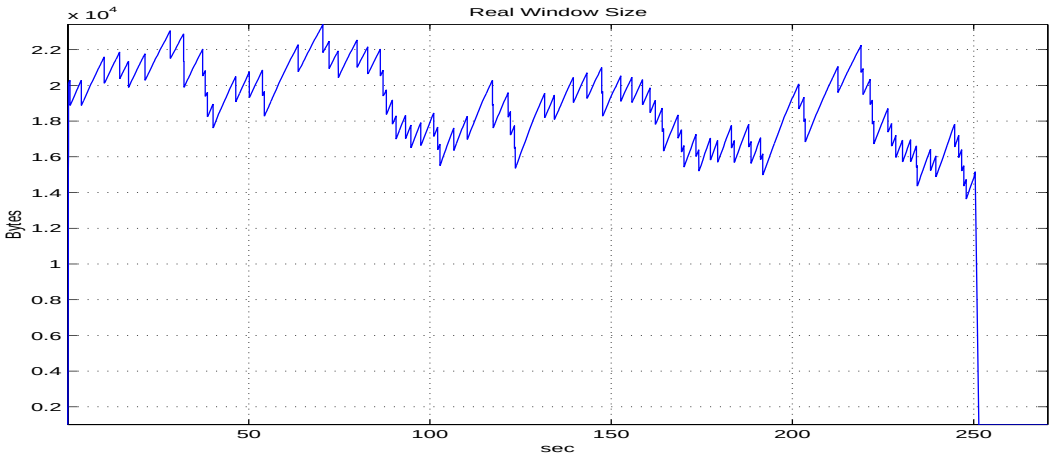

Figure 4.29: Window variations of BCC that delay based SFD is employed, k=0.2 over 2.0 Mbits/sec fair share bottleneck

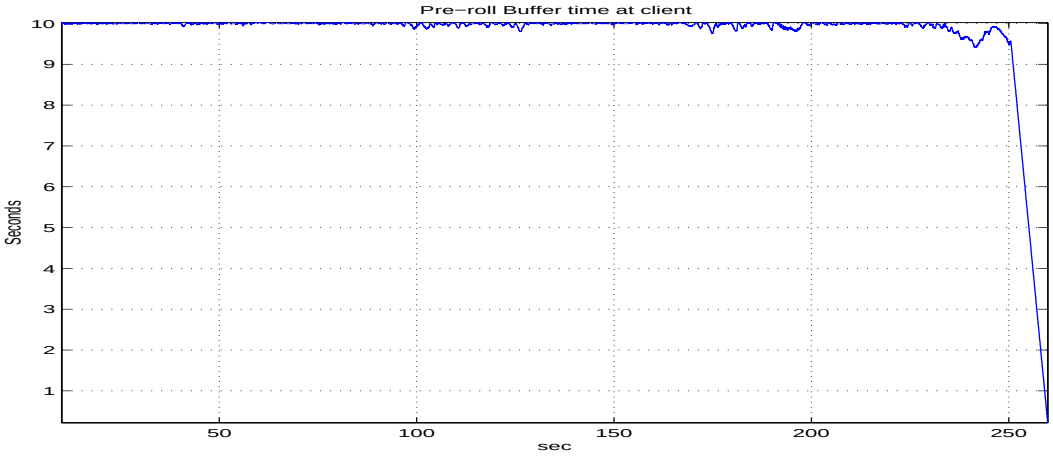

Figure 4.30: Play-out duration of BCC delay based SFD, k=0.2 over 2.0 Mbits/sec fair share bottleneck

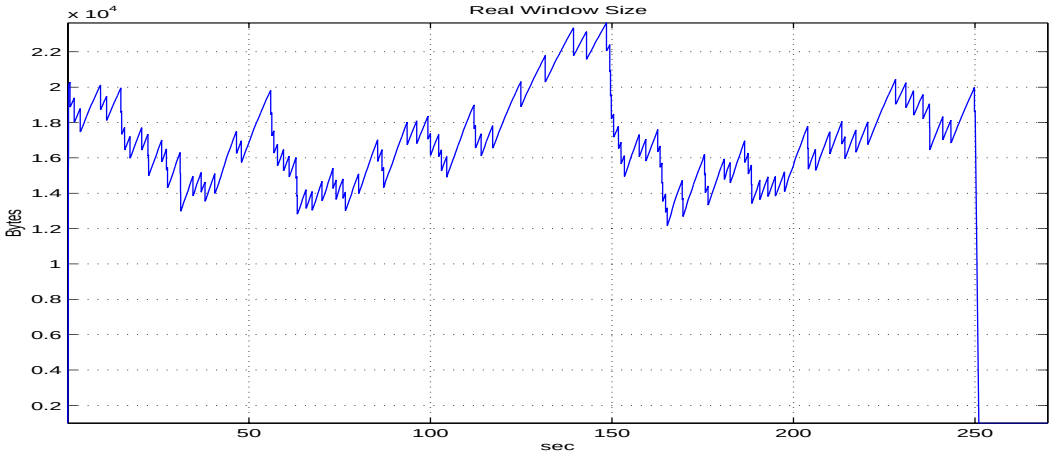

Figure 4.31: Window variations of BCC that length based SFD is employed, k=0.2 over 2.0 Mbits/sec fair share bottleneck

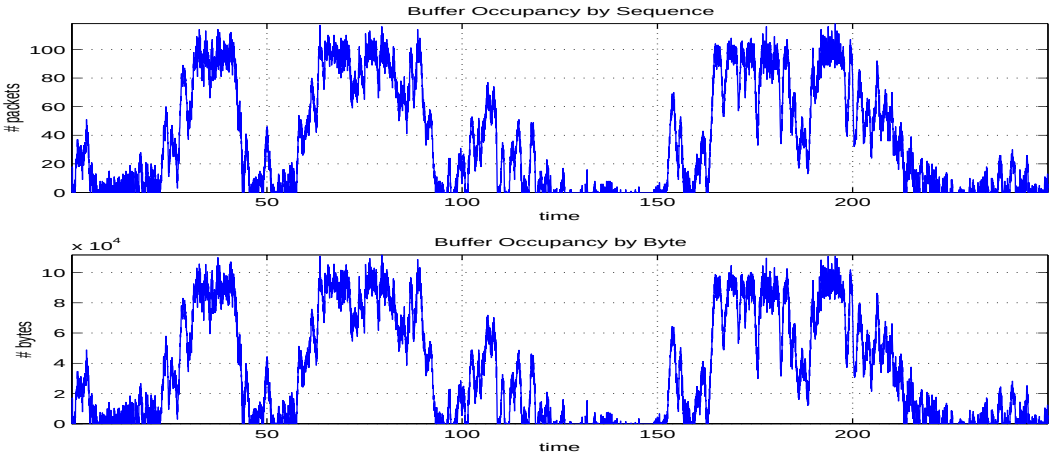

Figure 4.32: Length based SFD output buffer occupancy of BCC k=0.2 over 2.0 Mbits/sec fair share bottleneck

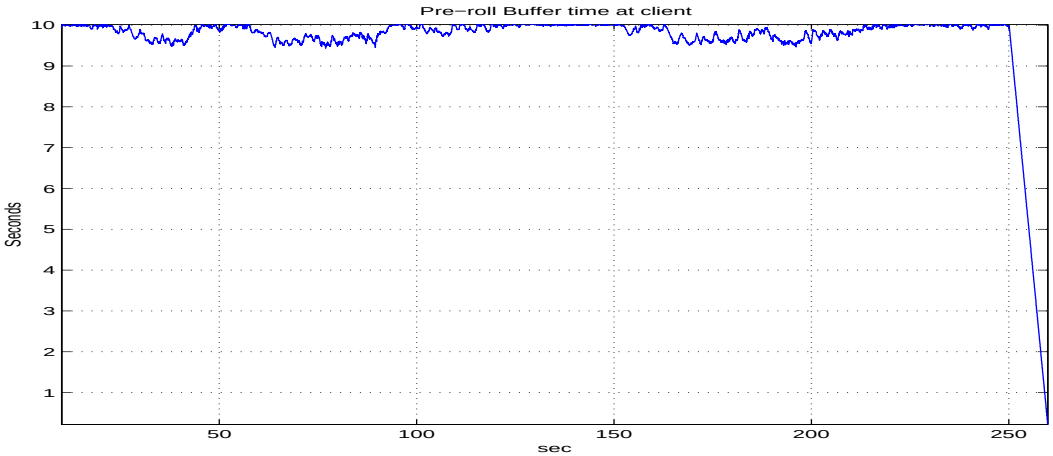

Figure 4.33: Play-out buffer duration of BCC that length based SFD is employed, k=0.2 over 2.0 Mbits/sec fair share bottleneck

Total HP frames of the Video	597						
Total LP frames of the video	6069	40 Mbit Shared by 20 Connections					
Total HP packets of the video	15442	Length Based		Delay Based		UDP	
Total LP packets of the video	52246	count	percent	count	percent	count	percent
HP frames Dropped at the Server		0	0	0	0	0	0
LP frames Dropped at the Server		1719	3	8	0	0	0
HP packets lost during transmission		1	0	1	0	118	0.10%
LP packets lost during transmission		8	0	8	0	1038	1.00%
HP packets rejected at the client		0	0	0	0	0	0
LP frames rejected at the client		0	0	0	0	0	0
Actual rate taken by each connection		1.98682Mbps		2.037 Mbps		2.000 Mbps	

Table 4.3: Loss based statistics of different schemes over 2.0 Mbits/sec available bandwidth channels

The loss statistics of the streaming results with techniques: delay based, length based SFD and finally UDP is given in Table 4.3.

As we can see, UDP stream suffers some losses even though it is streamed at the video rate. These losses are due to the burst of the packets created by the video frames that are of larger sizes. A single frame is streamed at its output line rate. Therefore at the bottleneck link as the packets face no other congestion, they arrive instantly at higher rates than the bottleneck link can carry. However their overall bit rate is matching to the line rate, some loss can be suffered over the network.

4.3.4 Multi-bottleneck Network Scenario

For the multi-bottleneck scenario we will consider a dual-bottleneck network, as in Figure 4.34. The first bottleneck link is shared with a total of 10 connections, where the second bottleneck as it is also being shared by the 10 same connections, and 10 more connections participating from this point on. This scenario will provide us some insights on the effects of distance. The first bottleneck that has the capacity of 16 Mbits per second, providing 1.6 Mbits/s capacity for each connection. Second has 32 Mbits/sec capacity which also provides the connection of 1.6 Mbits/sec for each.

The window size variation of a source that traverses the both bottlenecks is given in Figure 4.35. As we can observe from the window size variations, the source has never fallen into a timeout, this shows us that our parameter set is

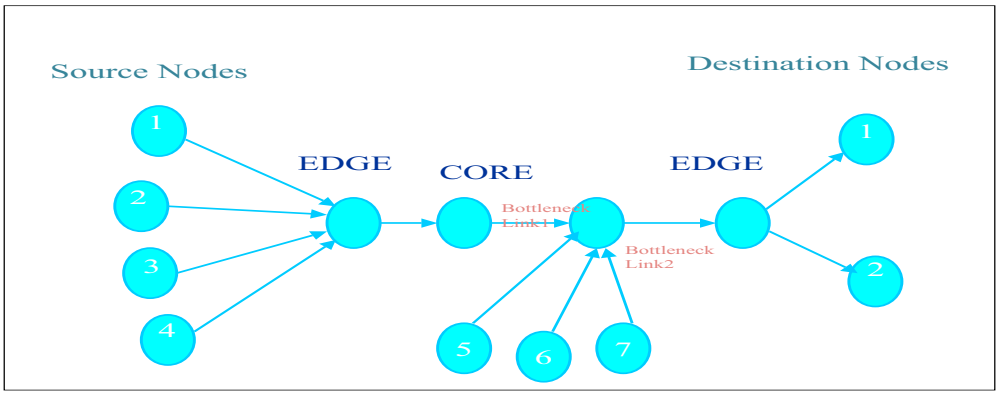

Figure 4.34: The Multi-bottleneck network topology

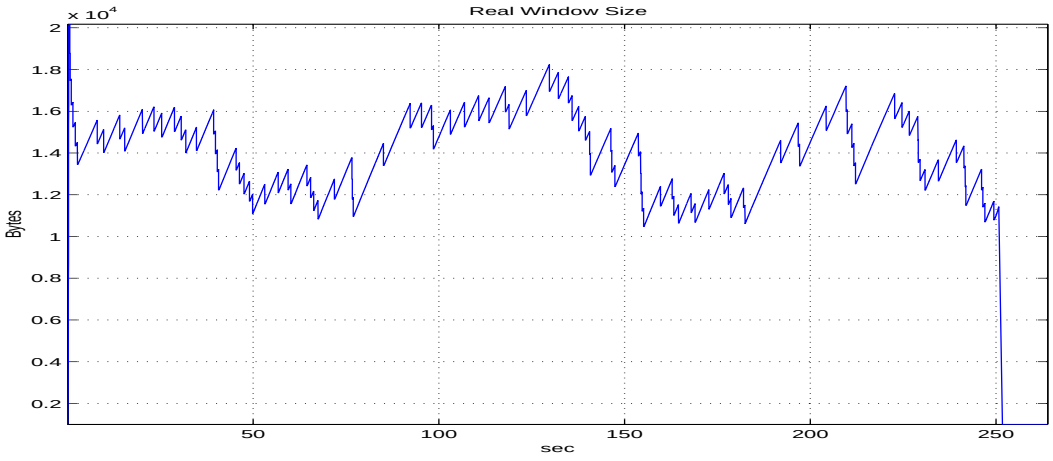

Figure 4.35: The window variations for multi-bottleneck topology

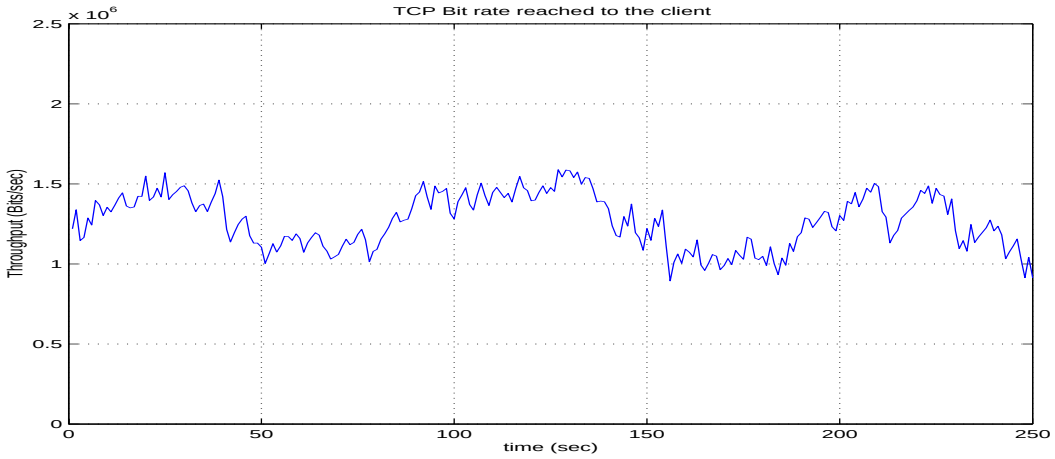

Figure 4.36: The resultant throughput for multi-bottleneck network topology

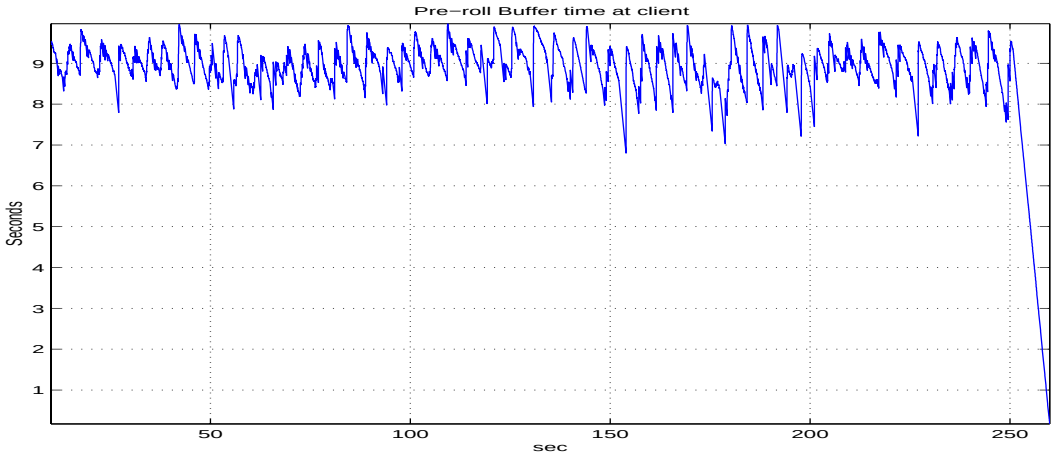

Figure 4.37: The resultant play-out duration for multi-bottleneck network topology for delay based discarding

not only smooth but at the same time flexible enough to adapt the change in the conditions. Therefore, it results in the necessary responsive behavior that is necessary for a healthy transmission. The throughput resulting from the same window size is visible through the Figure 4.36. The throughput is smoothly varying over time, however it is little less than its fair share of the both bottleneck links. This is because of the difference in the round trip times among the sources. The far sources need more time to understand the network conditions. However, closer sources take advantage of being close to the destination and can respond to the events in the network earlier. The resultant play-out buffer occupancy is given in Figure 4.37.

The resultant play-out buffer occupancy with the length based scheme is given in Figure 4.38, for the window size variations in Figure 4.39.

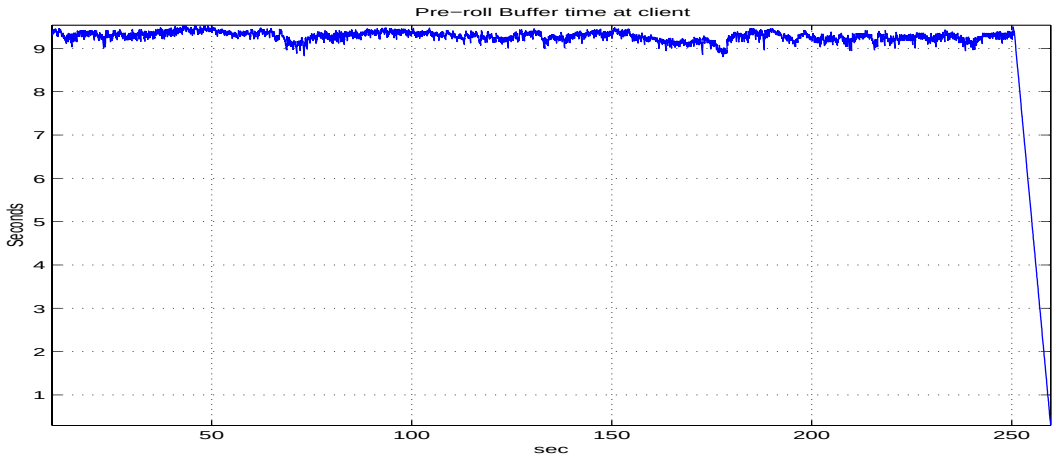

Figure 4.38: The resultant play-out duration for multi-bottleneck network topology of delay based system

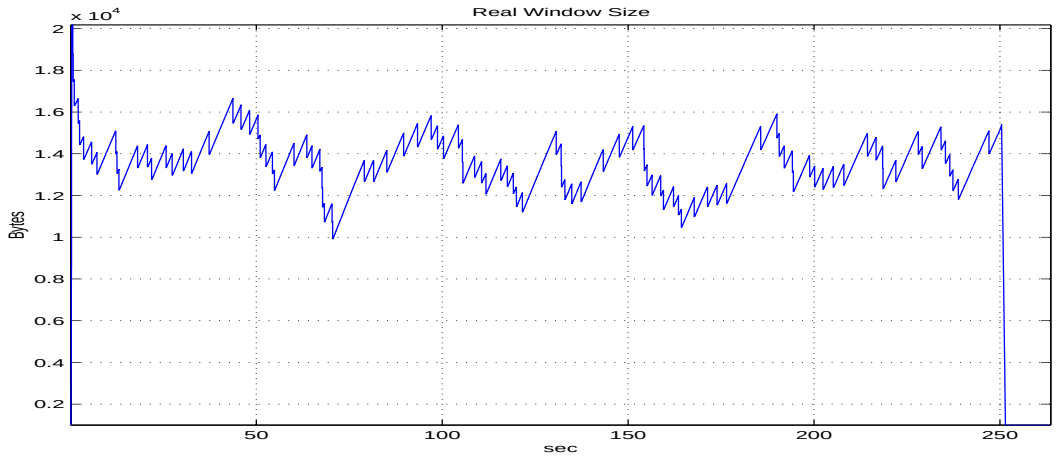

Figure 4.39: The window size variations for multi-bottleneck network topology of delay based system

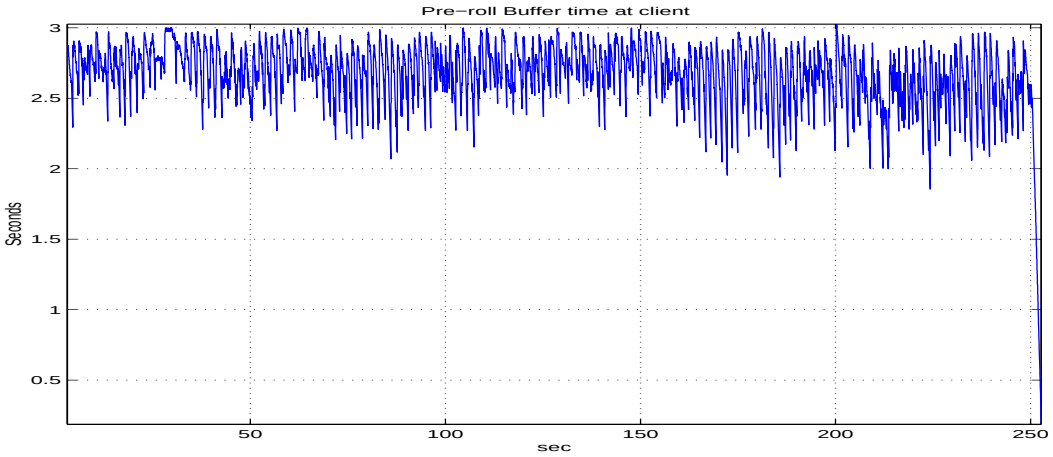

Figure 4.40: The resultant play-out duration for single-bottleneck network topology for delay based discarding

4.4 Effect of pre-buffering period

In this section we will consider the effect of the initial waiting period before starting the play-out. The network considered is the single bottleneck topology in Figure 4.1, for which 20 sources are sharing the 32 Mbits/sec bottleneck link. For all the above simulations, the initial waiting period before the play-out of the content to start was chosen to be 10 seconds. This duration is observed to be sufficiently large for the continuity of the content in the long timeout situations. However, lower period of waiting for the play-out may be achieved. First we chose 3 seconds as the alternative period. The resultant play-out duration of this scheme shown in Figure 4.40, for the window size variations in Figure 4.41. As we can see the play-out duration over the transmission period is kept at the level of 3 seconds.

The window size is continuously dropping which results in decreasing of the throughput, however as a merit of our SFD mechanism, the playout buffer duration stays around the 3 seconds all through the transmission.

Although the 3 seconds period is successful enough, some larger values may be more suitable in order to provide some extra caution for the video streaming. 6 second case is considered and the resultant play-out duration is shown in Figure 4.42. The window size variations for this transmission is shown in Figure 4.43.

As we can observe, even though there are some small variations on play-out duration, it is set to 6 seconds which is exactly the time that the client waits to play the video content. Also 10 seconds and larger waiting periods will have

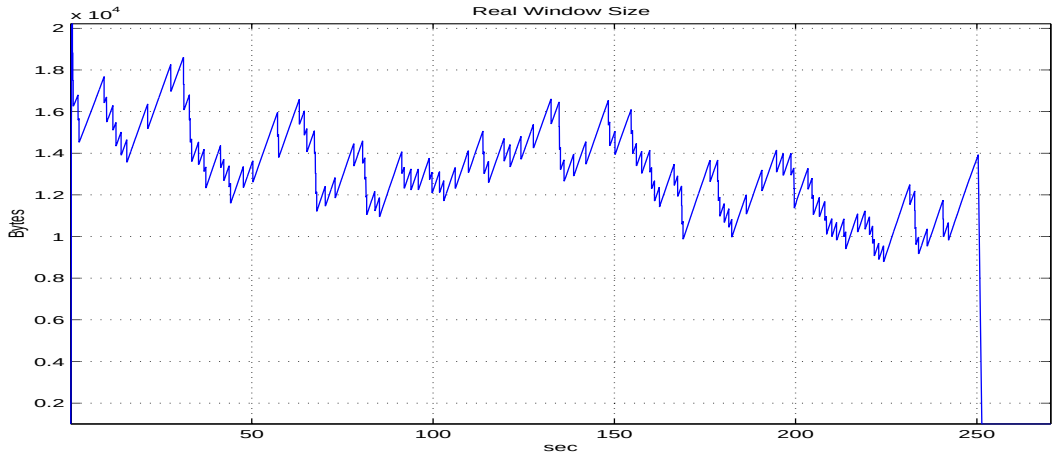

Figure 4.41: Window size variations for the transmission that the client waits 3 seconds in order to start the play-out of the video

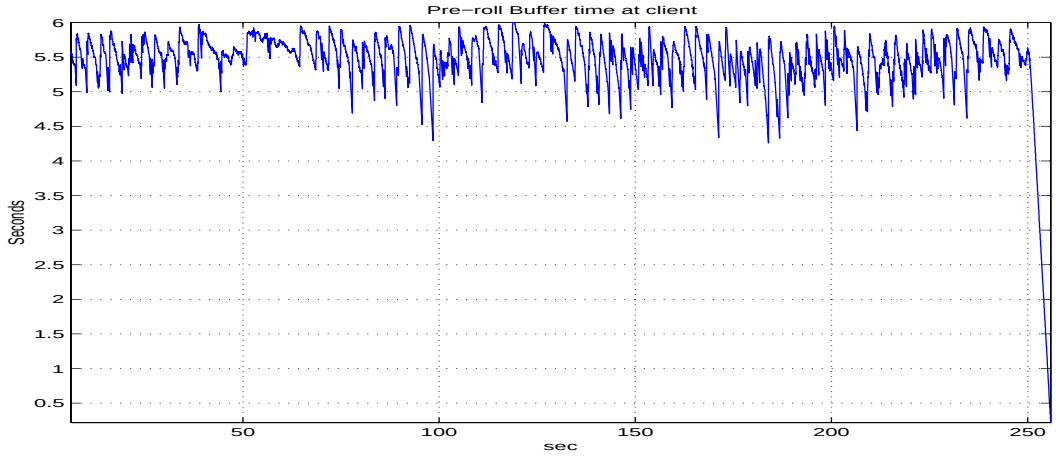

Figure 4.42: The resultant play-out duration for single-bottleneck network topology for delay based discarding

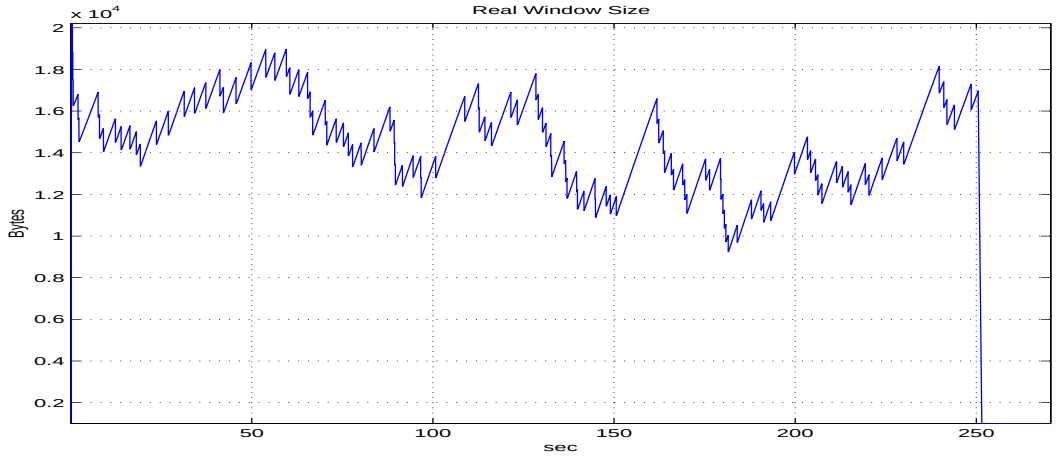

Figure 4.43: Window variations for the transmission that the client waits 6 seconds in order to start the play-out of the video

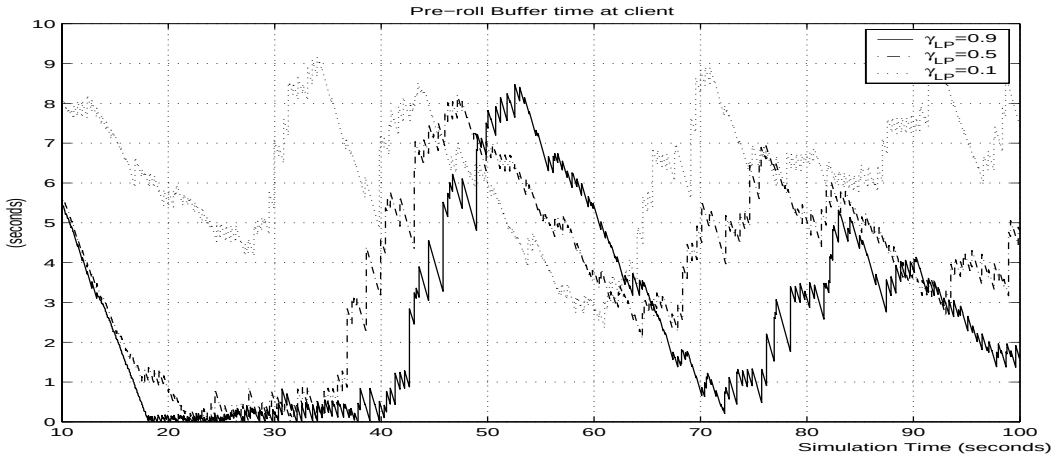

Figure 4.44: Play-out buffer lengths for bottleneck link of 10 Mbits and γ_{LP} = 0.9, 0.5, 0.1

similar results. Therefore whatever the network conditions are, it is possible to maintain a steady play-out duration, if there is a transmission going on.

4.5 Effect of γ_{LP} parameter

Under the assumption of sufficient bandwidth conditions for High priority frames, at this section we will investigate the effect of the γ_{LP}. According to this assumption γ_{HP} was selected to be infinite. The γ_{LP} parameter effects the bound on the delay at the quality adaptation buffer. Therefore, adaptation of stream under different conditions are effected by the value of this parameter.

If γ_{LP} parameter is set close to 1, the probability of packets missing their play-out times is high. In such a case the play-out buffer length in seconds will also decrease drastically and buffer will underflow. By setting smaller values to this parameter it is possible to control the delay at the server. The smaller values will perform even better. In order to observe the differences between these parameters, we have evaluated a very limited bandwidth scenario. We fixed the bottleneck link to 10 Mbits/sec. This bottleneck is again shared by 20 sources. Since the bandwidth requirement for High Priority frames is around 450 KBits/sec, there is enough bandwidth for this type of packets. However, there will be very little of available bandwidth for the low priority frames.

Under the same bandwidth conditions, we evaluated different γ_{LP} parameters. The obtained play-out buffer durations are given in Figure 4.44. The transmission

rates obtained by each evaluation of the γ_{LP} are smooth and similar. As we can observe in the figure, during the period between 20 and 30 seconds the buffer durations for the $\gamma_{LP} = 0.9$ $\gamma_{LP} = 0.5$ are exhausted. During this period play-out will pause due to the absence of the content that might be displayed. Many packets would miss their play-out deadline if we were not applying a lower threshold for the drainage of this buffer. By use of this mechanism already transmitted content is prevented from being lost. Since some level of frame rate changes are not noticable to humans [25] inefficient use of bandwidth is prevented. This method prevents the packets missing their play-out times by pausing the play-out. In the previous results this mechanism did not have any effect since the play-out buffer durations were high enough. As we can see $\gamma_{LP} = 0.5$ recovers more quickly from the buffer underflow situation and does not fall into such a situation during the simulation period. However, the $\gamma_{LP} = 0.9$ exhausts its buffer again after 70 seconds of the simulation. Also on the average its buffer duration is generally below the other two parameter set. The $\gamma_{LP} = 0.1$ does not lose many of its buffer length during the transmission period and recovers from the situations, by maintaining its buffer level at some level, that other two parameter values are performing worse.

Chapter 5

Conclusions

The inherent uncooperative behavior of UDP used currently as the transport protocol of choice for video networking applications, is known to be leading to congestion collapse of the Internet. Congestion collapse can be prevented by using mechanisms in networks that penalize uncooperative flows like UDP or employing end-to-end congestion control. Since today's vision for the Internet architecture is based on moving the complexity towards the edges of the networks, employing end-to-end congestion control for video applications has recently been a hot area of research. One alternative is to use a TCP-friendly end-to-end congestion control scheme. Such schemes, similar to TCP, probe the network for estimating the bandwidth available to the session they belong to. The average bandwidth available to a session using a TCP-friendly congestion control scheme has to be the same as that of a session using TCP. Some TCP-friendly congestion control schemes are highly responsive as TCP itself leading to undesired oscillations in the estimated bandwidth and thus fluctuating quality. Slowly responsive TCP-friendly congestion control schemes to prevent this type of behavior have recently been proposed in the literature.

However, throughout the previous work done, it was observed that the slowly responsive algorithms lose throughput against faster ones (like TCP) under dynamic network conditions. This reduction in throughput is the price that needs to be paid in return for the smoothness. Our experimental studies lead to similar results. BCC may still be deployed since it does not take any throughput away from the existing TCP connections. The interaction between TCP and slowly responding congestion control mechanisms are still an open issue that need to be considered and researched in great depth. Even though some parameter sets

for the BCC may result in unfair interactions in some network scenarios, the parameter set that we are using provides the necessary responsiveness and the smoothness.

We have proposed an architecture for video streaming in IP networks using slowly responding TCP-friendly end-to-end congestion control. In particular, the congestion control algorithm used is based on BCC. In this architecture, the video streaming device intelligently discards some of the video packets of lesser priority before injecting them in the network in order to match the incoming video rate to the estimated bandwidth using BCC and to ensure a high throughput for those video packets with higher priority.

In this thesis, major work done was the development of a simulation testbed using ns-2 that would be used for simulations of Internet video streaming. We have shown the efficacy of the proposed architecture using simulations in a variety of scenarios on the developed test-bed. However, the functionality of the system may not yet be optimal. Further investigation and analysis of the system will provide more insight for improving system performance.

Since pre-stored video is coded at a fixed rate, a rate shaping mechanism is required in order to match the rate of the video content to the dynamically determined rate by the congestion control mechanism. The scheme that we have developed provides necessary adaptation by discarding the parts of video in respective order of their importance. By employing such a mechanism, it is possible to stream a presentable video with lower quality but at rates that are much lower than the original bit rate requirement of the video. Our mechanism provides a good performance even in the conditions that normally streaming by UDP would not result in acceptable video play-out. The delay based scheme that we have developed has the best means of adaptation according to the network conditions. Although not analytically, this has been shown by the mean opinion scores that we have obtained for an audience of 10 people. The basic length based thresholding scheme also provides the adaptability but with a lower quality.

Bibliography

[1] D. Wu, Y. Hou, W. Zhu, Y. Zhang, and J. Peha, "Streaming Video over the Internet: Approaches and Directions," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 282–300, March, 2001.

[2] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, 1999.

[3] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based Congestion Control for Unicast Applications," in *SIGCOMM 2000*, (Stockholm, Sweden), pp. 43–56, August 2000.

[4] D. Bansal and H. Balakrishnan, "Binomial Congestion Control Algorithms," in *INFOCOM*, pp. 631–640, 2001.

[5] R. Rejaie, M. Handley, and D. Estrin, "RAP: An End-to-End Rate-Based Congestion Control Mechanism for Realtime Streams in the Internet," in *INFOCOM*, pp. 1337–1345, 1999.

[6] R. Rejaie, M. Handley, and D. Estrin, "Layered Quality Adaptation for Internet Video Streaming," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, pp. 2530–2543, 2000.

[7] N. G. Feamster, "Adaptive Delivery of Real-Time Streaming Video," *M. Eng. Thesis, Massachusetts Institute of Technology*, May 2001.

[8] T. Liu, W. Qi, H. Zhang, and F. Qi, "A Systematic Rate Controller for Mpeg-4 FGS Video Streaming," in *ICIP01*, p. Layered and Scalable Video, 2001.

[9] Network Simulator Version 2, "www.isi.edu/nsnam/ns,"

[10] K. Ramakrishnan and S. Floyd, "RFC 2481 A Proposal to add an Explicit Congestion Notification (ECN) to IP."

[11] G. Armitage, "Quality of Service in IP Networks, Foundations for a Multi-Service Internet," *Macmillan Technical Publishing,*, April 2000.

[12] W. Stevens, "TCP/IP Illustrated, Volume 1: The Protocols," *Addison-Wesley*, 1994.

[13] J. F. Kurose and K. W. Ross, "Computer Networking," *Addison-Wesley*, 2001.

[14] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker, "Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms," *Proceedings of ACM SIGCOMM '01*, 2001.

[15] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A Model Based TCP-friendly Rate Control Protocol," *UMass-CMPSCI Technical Report TR 98-04*, 1998.

[16] D. Sisalem and H. Schulzrinne, "The Loss-Delay Based Adjustment Algorithm: A TCP-friendly Adaptation Scheme," in *Proceedings of NOSSDAV*, (Cambridge, UK.), 1998.

[17] Y.-G. Kim, J. Kim, and C.-C. J. Kuo, "Smooth and Fast Rate Adaptation Mechanism (SFRAM) for TCP-friendly Internet Video," in *Packet Video Workshop*, 2000.

[18] J. Crowcroft, J. Roberts, and M. I. Smirnov, eds., *Quality of Future Internet Services, First COST 263 International Workshop, QofIS 2000, Berlin, Germany, September 25-26, 2000, Proceedings*, vol. 1922 of *Lecture Notes in Computer Science*, Springer, 2000.

[19] Y. Yang and S. Lam, "General AIMD Congestion Control," *Y. R. Yang and S. S. Lam. General AIMD Congestion Control. Technical Report TR-200009, Department of Computer Science, University of Texas at Austin, May*, 2000.

[20] K. Chandrayana, B. Sikdar, and S. Kalyanaraman, "Comparative Study of TCP Compatible Binomial Congestion Control Schemes," in *Proceedings of IEEE HPSR, Kobe, Japan, May*, 2002.

[21] S. Jacobs and A. Eleftheriadis, "Streaming Video Using Dynamic Rate Shaping and TCP Flow Control," *Journal of Visual Communication and Image Representation*, vol. 9, no. 3, pp. 211–222, 1998.

[22] C. Krasic, K. Li, and J. Walpole, "The Case for Streaming Multimedia with TCP," *Lecture Notes in Computer Science*, vol. 2158, pp. 213–.., 2001.

[23] M. Miyabayashi, N. Wakamiya, M. Murata, and H. Miyahara, "MPEG-TFRCP: Video Transfer with TCP-friendly Rate Control Protocol," *Proceedings of IEEE International Conference on Communications (ICC2001), Helsinki*, vol. 1, pp. 137–141, 2001.

[24] M. Miyabayashi, N. Wakamiya, M. Murata, and H. Miyahara, "Implementation of Video Transfer with TCP-friendly Rate Control," *Proceedings of International Technical Conference on Circuits/Systems, Computers and Communications 2000*, vol. 1, pp. 117–120, 2000.

[25] M. Kalman, E. Steinbach, and B. Girod, "Adaptive Media Playout for Low Delay Video Streaming over Error-prone Channels," *IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Wireless Video*, 2001.

[26] M. Kalman, E. Steinbach, and B. Girod, "Rate-Distortion Optimized Video Streaming with Adaptive Playout," in *International Conference on Image Processing, ICIP, Rochester, New York*, 2002.

[27] K. Sripanidkulchai and T. Chen, "Network-adaptive Video Coding and Transmission," in *Proc. Visual Communications and Image Processing, San Jose*, 1999.

[28] D. S. Turaga and T. Chen, "Classification Based Mode Decisions for Video over Networks," *IEEE Transtions on Multimedia*, vol. 3, no. 1, pp. 41–52, 2001.

[29] R. Puri, K.-W. Lee, K. Ramchandran, and V. Bharghavan, "An Integrated Source Transcoding and Congestion Control Paradigm for Video Streaming in the Internet," *IEEE Transactions on Multimedia*, vol. 3, no. 1, pp. 18–32, 2001.

[30] S. H. Kang and A. Zakhor, "Packet Scheduling Algorithm for Wireless Video Streaming," in *Proc. of PacketVideo '02*, 2002.

[31] S. Jacobs and A. Eleftheriadis, "Real-time Dynamic Rate Shaping and Control for Internet Video Applications," *Workshop on Multimedia Signal Processing*, pp. 23–25, June, 1997.

[32] A. Balan, O. Tickoo, I. Bajic, S. Kalyanaraman, and J. Woods, "Integrated Buffer Management and Congestion Control for Video Streaming," in *Proc. Globecom*, 2002.

[33] Z. Jiang and L. Kleinrock, "A Packet Selection Algorithm for Adaptive Transmission of Smoothed Video over a Wireless Channel," *Journal of Parallel and Distributed Computing*, vol. 60, no. 4, pp. 494–509, 2000.

[34] W. Feng, M. Liu, B. Krishnaswami, and A. Prabhudev, "A Priority-based Technique for the Best-effort Delivery of Stored Video," *Proc. of Multimedia Computing and Networking*, January 1999.

[35] A. Vetro, Y. Wang, and H. Sun, "Rate Distortion Optimized Video Coding Considering Frameskip," in *ICIP01*, pp. 534–537.

[36] W. Tan and A. Zakhor, "Packet Classication Schemes for Streaming MPEG Video over Delay and Loss Differentiated Networks," *In 11th International Packet Video Workshop, Kyongju, Korea, May*, 2001.

[37] U. J. Daniel Forsgren and P. Österberg, "Objective End-to-End QoS Gain from Packet Prioritization and Layering in MPEG-2 Streaming," in *Proc. 12th International Packet Video Workshop (PV-2002), Pittsburgh, USA, 24-26 April*, 2002.

[38] J. K. Jitae Shin and C.-C. J. Kuo, "Quality-of-Service Mapping Mechanism for Packet Video in Differentiated Services Network," *IEEE Transactions on Multimedia*, vol. 3, no. 2, pp. 219–231, 2001.

[39] Y.-G. Kim and C.-C. J. Kuo, "TCP-friendly Layered Video for Internet Multicast," in *VCIP*, 2002.

[40] N. Feamster, D. Bansal, and H. Balakrishnan, "The Interactions Between Layered Quality Adaptation and Congestion Control for Streaming Video," *11th International Packet Video Workshop*, 2001.

[41] M. Hemy, U. Hengartner, P. Steenkiste, and T. Gross, "MPEG System Streams in Best-Effort Networks," in *Proc. of PacketVideo '99, New York*, April 1999.

[42] T. P. Chen and T. Chen, "Adaptive Joint Source-Channel Coding Using Rate Shaping," *ICASSP*, 2002.

[43] Z.-L. Zhang, S. Nelakuditi, R. Aggarwal, and R. P. Tsang, "Efficient Selective Frame Discard Algorithms for Stored Video Delivery across Resource Constrained Networks," in *INFOCOM*, pp. 472–479, 1999.

[44] M. Kalman, E. Steinbach, and B. Girod, "Adaptive Playout for Real-time Media Streaming," *International Symposium on Circuits and Systems, ISCAS*, Scottsdale Arizona, May 2002.

[45] G. Bozdagi and T. Sencer, "Preprocessing Tool for Compressed Video Editing," *International Workshop on Multimedia Signal Processing MMSP99.*