University of Alberta

Efficient Congestion Control for Internet Video Streaming

By

Patrick Baker Ssesanga

A Thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science

Department of Electrical and Computer Engineering

Edmonton, Alberta

Fall 2005

Your file Votre référence
ISBN: 0-494-09294-7
Our file Notre référence
ISBN: 0-494-09294-7

University of Alberta

Library Release Form

Name of Author: Patrick Baker Ssesanga

Title of Thesis: Efficient Congestion Control for Internet Video Streaming

Degree: Master of Science

Year this Degree Granted: 2005

Sept 29, 2005 [illegible]

Abstract

Efficient adaptive Internet video streaming of scalable video requires suitable transport control algorithms to adapt the transmission rate to the fluctuating network conditions. The algorithms implemented in TCP, the most prevalent Internet transport control protocol, are not suitable for video streaming as they create undesirable rate and hence viewer quality fluctuations. In this thesis we propose a streaming congestion control mechanism that uses the receiver buffer to monitor the occurrence of congestion and to determine the degree of its effect on the video connection. This information is used in determining how much the connection should adjust its rate to adapt to the network condition, resulting in a more robust technique of predicting available bandwidth for the video session. The simulation results show that the scheme performs well under congestion and bandwidth probing, and is fair to TCP and other streams while maintaining a smooth throughput profile.

Acknowledgements

I would like to thank my supervisor, Dr. Mandal for the support and guidance he has extended to me during my Maters Program and more particularly in the research and preparation of this thesis. I would also like to thank my research group mates for being a very supportive team. Many thanks go to the Canadian Council for International Studies for their financial support that has enabled me to undertake this Masters program.

To my family who have been so patient during the years that this program separated me from them, I am very grateful.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ACK	Acknowledgement
AIAD	Additive Increase Additive Decrease
AIMD	Additive Increase Multiplicative Decrease
API	Application Programming Interface
BIFS	BInary Format for Scenes
DAI	DMIF Application Interface
DMIF	Delivery Multimedia Integration Framework
EDF	Early Deadline First
FEC	Forward Error Correction
FGS	Fine Granular Scalability
IP	Internet Protocol
ISO	International Standards Organization
ITU	International Telecommunications Union
MDC	Multiple Description Coding
MIMD	Multiplicative Increase Multiplicative Decrease
MPEG	Moving Picture Expert Group
NAS	Network Attached Storage
OD	Object Descriptor
QoS	Quality of Service

RTCP	Real Time Control Protocol
RTP	Real Time Transmission Protocol
RTSP	Real Time Streaming Protocol
RTT	Round Trip Time
SAN	Storage Area Network
SIP	Session Initiation Protocol
SSP	Stream Synchronization Protocol
TCP	Transmission Control Protocol
TFRC	TCP Friendly Rate Control
UDP	User Datagram Protocol
VLVB	Very Low Bitrate Video Coding

Chapter 1

Introduction

Over the last decade, the Internet has seen such tremendous growth that it has positioned itself as the major communication network providing critical services such as e-mail, web access, audio and video, telephony, and multimedia content distribution. Multimedia content may be live or recorded depending on the application. The live content is generally streamed from a central distribution server either as intra-organization content or as entertainment content for general public access.

The Internet has become a major vehicle for multimedia content distribution with one such application being video streaming. Streaming generally refers to the real-time rendering of content as it is being received from the source. For live content, this is the only access option. The timeliness required of the transmitted stream of packets places quality of service (QoS) constraints on the underlying network. The recipients of the audio and video data would naturally expect an error-free, uninterrupted reception. Therefore, the Internet being a best effort network with little or no QoS guarantees is a particularly challenging network to stream live data. There is no bandwidth guarantee in the public Internet, and hence when congestion is experienced some packets will be lost. In addition, some received packets are corrupted during transmission. It is up to the delivery system to make sure that packets are received in the correct order and within the expected play out time.

Given the dynamic bandwidth in the Internet, the streaming process should be able to adapt to the variations in bandwidth in order to play the audio or video data smoothly and continuously. A robust error control mechanism is also necessary to minimize the effects of error and/or loss of packets. Streaming models can be broadly divided into two categories: unicast and multicast. In unicast, all clients interact directly with the streaming server, while in multicast the server sends out only one stream to a group of clients congregated around a multicast routing device. Although the multicast streaming relieves the server of the heavy load, it is only useful in broadcast like applications. A particularly challenging situation is when a given stream is to be randomly accessed such as in video-on-demand applications where the users start sessions at random times and desire to use VCR-like trick modes. This is a unicast situation, which requires the server to be highly scalable to accommodate the heavy load. It also places a heavy bandwidth demand on the video distribution network [1]. The trick modes such as fast-forward and rewind are particularly challenging as they imply increased data rate and hence bandwidth [2].

1.1 CHALLENGES

Streaming video over the Internet is quite challenging because the Internet and underlying communication protocols were designed for data communication. These protocols are therefore more concerned with the integrity of the data being delivered which does not have, in most instances, strict timing constraints. The Transmission control protocol (TCP) is the most common transport protocol for Internet data and it was designed to enable retransmissions of any corrupt data to ensure data integrity. This renders the TCP unsuitable for live video transmission in the Internet. Suitable video transport protocols should be able to accommodate the timeliness delivery requirement of video packets from server to client while at the same time responding to network capacity fluctuations caused by congestion that is common in connectionless packet networks such as the Internet. TCP responds to congestion by halving the transmission rate and then increasing the rate when the condition is presumed to have subsided. This mechanism helps the Internet to avoid collapsing under heavy congestion. Although this mechanism is acceptable for data traffic, it is not suitable for video and other type of real time traffic. Real time traffic, therefore, mostly uses the UDP for transmission over the Internet because the UDP does not carry the retransmission and rate reduction overhead of the TCP. However, since all communicating sessions are supposed to respond to congestion by backing off to maintain network stability, streaming applications incorporate their own control protocols. This allows these applications to behave as expected of Internet applications while at the same time maintaining acceptable quality that usually suffers greatly during network fluctuations. This behavior of mimicking the TCP is called *TCP friendliness* and is required of all streaming control protocols used in a network with TCP traffic. *Smoothness* is another required quality of a streaming control protocol and this refers to how the protocol is able to minimize quality fluctuations when responding to congestion. A number of congestion control algorithms have been proposed in the literature for streaming applications. However, some of them do not respond well to the network congestion, and thus treat the TCP traffic unfairly. When they do respond to the congestion, they cause heavy undesirable quality fluctuations. This thesis aims at addressing some of the problems associated with the existing congestion control mechanisms.

1.2 Motivation and Objectives

Internet video streaming has been spreading fast for applications such as entertainment, information delivery, and education. Transport protocols, for delivery of content over the Internet, are typically data centric and therefore are not suitable for real time information delivery in video applications. A number of new protocols and modifications to the existing data transport protocols have been proposed for Internet video streaming. Because of the extensive use of the TCP as a transport protocol for Internet data, any new protocols are required to behave almost similar to a TCP stream in responding to network congestion. This maintains network stability and ensures that data traffic is not treated unfairly by the bandwidth hungry video traffic. A flow control algorithm for a video streaming application must therefore balance the need for limiting quality variation resulting from network fluctuations and TCP friendliness.

The most common congestion control algorithms in use today are based on the binomial family of algorithms similar to the TCP algorithms and equation based algorithms based on long term models of typical TCP sessions. These algorithms do not individually cater for the balanced requirement of video streaming applications. It is on this basis that we seek, in this thesis, to develop a new congestion control algorithm whose performance surpasses the existing algorithms for video streaming applications. We develop a congestion control algorithm that is based on the receiver buffer for the detection of congestion since congestion leads to a significant reduction in the buffer level as a result of increased queuing delays at the affected network nodes. The degree of response to congestion is also determined from the observed throughput as determined from the impact of congestion on the buffer level. We use both the temporal and actual data size in measuring the buffer length to take into consideration the timing requirements of the video application being supported. The client readily captures this information and periodically passes it on to the server that determines the most appropriate adjustment considering the network conditions in light of the timing and smoothness requirements of the video application.

1.3 ORGANIZATION OF THE THESIS

The organization of this thesis is as follows. Chapter 2 presents a review of the video compression technology for streaming applications and network congestion control protocols. In chapter 3, we present the proposed congestion control scheme. The performance of the proposed congestion control algorithm is presented in chapter 4. Chapter 5 presents the conclusion and future research direction.

Chapter 2

Review of Background Work

Video streaming has stringent requirements on bandwidth, delay and loss, and yet the Internet, being a best-effort network, does not inherently support QoS guarantees with respect to such demands. In this chapter we first present a general overview of video streaming; architecture, challenges and techniques to address some of the challenges. Extensive research has been conducted at all layers of the video streaming architecture to address the difficulties encountered in delivery of video over the Internet. Because of the fluctuating nature of Internet bandwidth, video streaming applications need to be adaptive to these conditions without significantly degrading the quality of the received video. Adaptive video streaming is realized through scalable video coding and efficient congestion control techniques. We review video coding technology for video streaming especially focussing on MPEG-4 and the features that make it the most competitive standard for this application. We then review a few selected congestion control techniques relevant to Internet video streaming.

2.1 VIDEO STREAMING ARCHITECTURE

The basic video streaming architecture is made up of six major layers namely; video compression, application layer QoS control, continuous media distribution services, streaming servers, media synchronization mechanisms, and streaming media transport protocols .

In Fig. 2.1 raw video and audio are compressed by the respective *compression algorithms* and then either stored or delivered directly to the application layer. In the case of stored video, the streaming server retrieves the compressed video/audio data from storage upon receiving a request from a streaming client. The *application layer QoS control* module adapts the video/audio bitstreams according to the network status and QoS requirements. The *transport protocols* packetize the adapted compressed bitstreams and transmit the media packets to the Internet. Continuous media distribution services (e.g. caching, multicasting, etc) are deployed in the Internet to offer load balancing and also to try and minimize the effects of Internet congestion, which result in packet loss or excessive delay. Peer application QoS control and transport layers exist in the client to provide complementary media processing for the client application. After the media streams have been decoded, the media are rendered with the help of *synchronization mechanisms* to preserve the media timing.

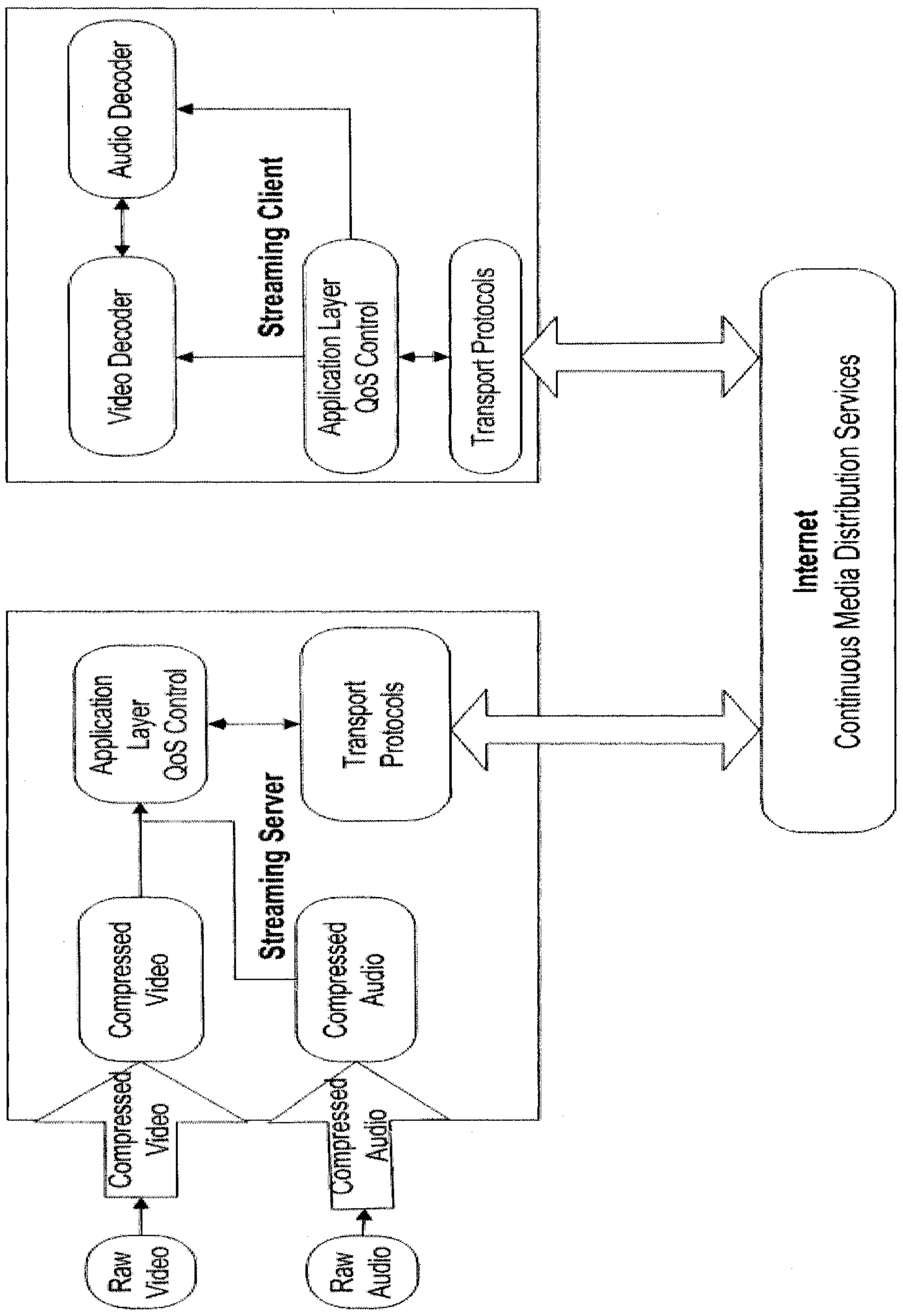

Figure 2.1: Video Streaming Architecture

2.1.1 Video Compression

The recent developments in digital video technology that have enabled unprecedented compression ratios have brought the possibility of delivering video over low bandwidth networks. Developments in error control with reduced overhead and the introduction of scalability have made a significant impact in low bit-rate multimedia communication. There are two main digital video standards organizations; the Moving Picture Expert Group (MPEG) under the International Standards Organisation (ISO) and the International Telecommunications Union (ITU). The MPEG has developed the MPEG series of standards while ITU has been responsible for the H26x series. ISO's MPEG-1 and MPEG-2 are mostly high bit-rate compression systems and are unsuitable for the low bit rate application (e.g. video transmission through the Internet). The ITU standards on the other hand have been developed for low bit applications. H.263 for example is one such standard that has been developed for video conferencing, but because the motion in conferencing applications is much lower than in other video applications this standard is not suitable for general video streaming purposes. The ITU has recently developed another high compression and scalable video coding standard known as H.264. On the other hand the ISO has come up with MPEG-4 which is also suitable for low and varying bandwidth applications. The scalability of these standards renders them suitable for streaming applications where network fluctuations dictate that some data be dropped to reduce the transmission rate. In this thesis we consider the MPEG-4 standard which is the more popular and loaded with several novel features.

2.1.2 Application Layer QoS Control

To cope with varying network conditions and different presentation quality requested by the users, various application layer QoS control techniques have been proposed [3], [4], [5], [6]. The application layer techniques, employed at the end systems, include congestion control and error control. Bursty loss and excessive delay, resulting from congestion, significantly affect the video presentation quality. Congestion control is therefore employed to minimize packet loss and reduce delay by adapting the transmission rate to the observed network conditions. This involves detecting the occurrence of congestion, reducing the transmission rate during congestion and then increasing the transmission rate when congestion subsides. Congestion control in video streaming applications is generally implemented as a rate control scheme [7]. Rate control techniques can be classified in three categories: source-based, receiver-based, and hybrid rate control.

In source-based rate control the sender is responsible for adapting the video transmission rate, based on the information it has gathered on the congestion state of the network. Source-based rate control schemes are implemented as either probe-based or model-based [7]. The probe-based approach is based on probing experiments aimed at estimating the available network bandwidth by monitoring session entities such as packet loss rate. The model-based approach is based on a throughput model of a TCP connection.

Under receiver-based rate control the receivers regulate the receiving rate of video streams by adding/dropping channels while the sender does not participate in rate control. Receiver-based rate control is used in multicasting scalable video, where there are several layers in the scalable video and each layer corresponds to one channel in the multicast tree. Like source-based rate control, receiver-based rate control is implemented through either probe-based or model based mechanisms. The advantage of receiver-based rate control is that the receiver can more accurately estimated the available network bandwidth from the observed throughput.

Hybrid rate control is implemented with the receiver and sender cooperating in detecting congestion and adjusting the transmission rate. This rate control method can be employed in both unicast and multicast video streaming applications.

Rate shaping is another function of the application layer QoS control plane and is aimed at matching the rate of the pre-compressed video bitstream to the target rate constraint [3]. Rate shaping is obviously implemented at the sender and involves the use of a filter, which may be a codec filter, a frame-dropping filter, layer-dropping filter, frequency filter, and re-quantization filter.

Error control is used to improve video presentation in the event of packet damage or loss. Error control mechanisms include forward error correction (FEC), retransmission, error-resilient encoding, and error concealment.

With FEC redundant information is added to the bitstream so it can be used in the recovery of lost information. Channel coding, source-coding and joint source/channel coding are the various categories of FEC. Channel coding for Internet applications is usually implemented in terms of block codes. Joint source/channel coding is employed for optimal rate allocation between source coding and channel coding.

Although retransmission is not very suitable for real-time applications, delay constrained selective retransmission techniques have been proposed in the literature [8]. Retransmission enables important packets that have been lost to be retransmitted if they can arrive at the receiver before their stipulated play out time.

Error-resilient encoding is employed to enhance robustness of compressed video to packet loss. Multiple description coding (MDC) [9] [10] is one such technique proposed for Internet video streaming applications. A major disadvantage of using the MDC and all the other preventive error control techniques is the overhead that comes with the extra information included with the video bitstream.

Error concealment is a reactive mechanism employed by the receiver to conceal the effects of lost data improving the video presentation [11}. Error concealment is implemented in two approaches namely, spatial and temporal interpolation. In the spatial interpolation, missing pixels values are reconstructed using neighbouring spatial information while in the temporal interpolation; lost data is reconstructed from the data in the previous frames. Other error concealment schemes include maximally smooth recovery [12], projection onto convex sets [13], and various motion vector and coding mode recovery methods such as motion compensated temporal prediction [14].

2.1.3 Continuous Media Distribution Services

Continuous media distribution services are designed to provide QoS and improve efficiency for streaming video/audio over the best effort Internet. These services include network filtering, application-level multicast, and content replication.

Network filtering is a congestion control technique that involves the employment of one or more filter nodes along a bottleneck path between the server and the receiver. These filter nodes transfer the media adaptation processing load from the server.

The multicast streaming architecture is highly scalable and therefore useful in continuous media distribution in multi-party setups. Whereas efforts have been made in defining the IP multicast protocols deployment has been on a limited scale. To exploit the advantages of multicast technology, application-level multicast techniques aimed at building multicast services on top of the Internet Protocol have been proposed [15]. In the media multicast networks, each multicast capable node, called a media bridge, performs routing at the application layer. The media bridges in a media multicast network employ a distributed the application-level multicast routing algorithm to determine the optimal virtual paths for propagating content throughout the network. When the underlying network fails or becomes overly congested, the media multicast network automatically and dynamically reroutes content via alternate paths according to application level routing policies. Media bridges also only subscribe to a media stream only when a client has requested for it.

Another important technique for improving scalability of the media delivery system is media replication. This takes two forms; caching [16] and mirroring [17]. In mirroring, distributed media servers are used to spread the media services closer to the clients and therefore reducing the load offered to the network. The benefits are realized as the number of clients increases enabling a streaming service to scale up efficiently with demand. Caching exploits the fact that different clients will demand much of the same content. Thus making copies available on a cache server upon the initial access enables subsequent requests to be serviced from the local copy without reloading from the source server.

2.1.4 Streaming Servers

The role of streaming servers is to manage storage and efficient processing of multimedia information under the strict timing constraints of a streaming application. In addition to supporting clean multimedia play back at the client, streaming servers are also expected to provide VCR-like control operations (e.g. stop, pause/resume, fast forward and fast rewind) for video-on-demand services. To realize these functions a streaming server is constituted by a real-time operating system, a communicator, and storage system. A real-time operating system, that employs real-time scheduling techniques, is essential for the streaming server to be able to fulfill the timing requirements in managing processing of continuous media. Two main scheduling algorithms have been proposed for multimedia applications namely; the earliest deadline first (EDF) [18] and rate monotonic scheduling [19].

In order to manage limited resources, efficient resource management is required. Resource management involves admission control and resource allocation. A streaming server must perform admission control tests to decide whether a new client connection can be admitted without violating performance guarantees already committed to the existing connections. If a connection is accepted, the resource manager allocates resources required to meet the QoS for the new connection.

Admission control algorithms can be classified as deterministic [20] or statistical [21]. Deterministic control algorithms provide hard guarantees to clients while statistical algorithms provide guarantees for only a percentage of media units. Correspondingly, resource (resources include processing power and bandwidth) allocation schemes can be either deterministic or statistical.

Storage systems for continuous media are other resources of the streaming server that need efficient design. The design of multimedia storage systems has to consider [22] high throughput, large capacity and fault tolerance requirements. High throughput can be realized with data scattering techniques such as data striping [23], while storage capacity can be increased by the use of multiple storage disks and large scale storage area networks (SAN) [24] [25]. The scalability of SAN's can be improved by implementing network attached storage (NAS) [26] techniques allowing stored content to be accessed directly using high speed connections. NAS also allows storage to be distributed geographically. File systems and data access protocols become significant in designing and building NAS systems. Fault tolerance enables servers to recover from data damage or loss. Error correcting techniques (parity encoding) [27] [28] [29] and disk mirroring [17] (having several copies of data in different locations or disks) are the major schemes used to realize fault tolerance in storage systems.

2.1.5 Media Synchronization

Media synchronization refers to maintaining the temporal relationships within one data stream and between various media streams. Three levels of synchronization can be identified as intra-stream, inter-stream, and inter-object synchronization. These layers of synchronization correspond to three semantic layers of multimedia data [30].

The lowest layer of continuous media is the media layer whose unit is logical data such a video/audio frame adhering to strict temporal constraints to ensure coherent user perception. Intra-stream synchronization is implemented at this layer to ensure the continuity of logical data units.

The second layer of continuous media is the stream layer whose basic unit is the whole stream. Inter-stream synchronization is employed to maintain temporal relationships among different media streams. One such case is the matching of audio with the corresponding talking-mouth video points.

The highest layer of a multimedia document is the object layer, which integrates streams and time-independent data such as text and still images. Synchronization at this layer is referred to as inter-object synchronization and it is aimed at starting and stopping the presentation of time-independent data within a tolerable time interval, when some previously defined of its presentation are reached.

As the media transit the Internet, they incur variable delays. These delay variations disrupt the embedded synchronization. Media synchronization mechanisms must be implemented at the client to ensure synchronized presentation of the media. These mechanisms can be proactive or reactive [31]. The proactive mechanisms, aimed at minimizing latencies and delay jitter, include versatile disk-reading scheduling algorithms, network transport protocols, operating systems and synchronization schedulers. Reactive mechanisms, implemented as compensation algorithms at the receiver, are designed to recover synchronization in the presence of synchronization errors. One such scheme is the stream synchronization protocol (SSP) [32] where presentation of some stream components is delayed in order to allow for recovery from network delay variations.

2.1.6 Protocols for Streaming Video

Streaming protocols include network-layer protocols, transport protocols and session control protocols. The Internet protocol (IP) is the backbone of the Internet at the network layer. Transport protocols for streaming applications include TCP [33], TCP friendly rate control (TFRC) [34], UDP [35], RTP (Real-time transport protocol) [36], and RTCP (Real-time control protocol) [37]. Session control protocols define the messages and procedures to control the delivery of the multimedia data during an established session.

The transport protocols, TCP, UDP and TFRC provide basic transport functions while RTP and RTCP run on top of them. Because of TCP's flow control mechanisms and retransmissions UDP and TFRC are more appropriate for the stringent timing requirements of streaming applications. Functions provided by RTP include time-stamping, sequence numbering, payload type identification, source identification, QoS feedback, participant identification, control packets scaling, inter-media synchronization, and minimal session control information.

The Real-time streaming protocol (RTSP) [38] and session initiation protocol (SIP) [39] are the two known session protocols. The RTSP is mainly employed to support VCR-like control operations, and also to provide the means for choosing delivery channels (e.g., UDP, Multicast, TCP, or TFRC) and delivery mechanisms based upon RTP. The SIP is a session control protocol used in the creation and termination of sessions with one or more participants. It also supports user mobility by proxying and redirecting requests to the user's current location.

2.2 Overview of MPEG–4 Video Coding Standard

MPEG-4 was developed by the Motion Picture Expert Group of the ISO as a new video and audio coding standard in the late 1990s. This standard provides the techniques for the storage, transmission and manipulation of video. The video data can consist of natural and synthetic textures, and either fixed images or video streams. The MPEG-4 standard is very versatile and allows encoding of video and audio at various bit rates. The low bit rate and error resilience coding capabilities of MPEG-4 allow for robust video communication over limited and unpredictable channels such as wireless channels and the Internet. The very low bit rate video core (VLVB) of MPEG-4 provides algorithms and tools for applications operating at bit-rates typically between 5-64 Kbits/s, supporting image sequences with low spatial resolution and low frame rates (typically up to 15 Hz). The MPEG-4 also supports transmission and storage of high quality video at bit rates ranging from 64 Kbits/s to 10 Mbits/s.

Unlike MPEG-2 and MPEG-1, the MPEG-4 standard provides for the use of various video compression techniques. With content-based functionality the design of MPEG-4 is centred on a basic unit [40], [41] called the audio-visual object (AVO). In object coding mode, while coding a video the MPEG-4 encoder creates a hierarchical structure of objects in a scene in which the objects are dynamic. This allows for easy editing as individual objects can be removed or added to scenes quickly and easily using the tools developed for MPEG-4. The scenes are divided into the background and foreground. The background and foreground can be subdivided further into objects. The objects are then encoded individually as independent compressed bit streams.

MPEG-4 also supports numerous other enhancements and techniques that include, but not limited to: efficient compression of standard rectangular-sized image sequences at various levels, various frame rates, pixel depth, bit-rates, and different types of scalability. These are discussed in the following sections.

2.2.1 Scalability

Since an MPEG-4 video stream consists of encoded audio-visual objects it allows for content-based scalability. Thus, at various stages in the authoring/delivery/consumption process (i.e. streaming), content can be ignored or adapted to match the bandwidth conditions, complexity or even price requirements for implementing such a system [40]. This does however introduce additional complexity, as clients have to send information on the line integrity back to the server which then limits or increases the overall bandwidth sent to the client.

There are several different methods for scaling an MPEG-4 video. The key ones are listed below [40]:

- **Complexity scalability** in the encoder allows encoders of different complexity to generate valid and meaningful bitstreams for a given texture, image or video. Complexity scalability in the decoder allows a given texture, image or video bitstream to be decoded by decoders of different levels of complexity. The reconstructed quality, in general, is related to the complexity of the decoder used. This may entail that less powerful decoders decode only a part of the bitstream.

- **Spatial scalability** allows decoders to decode a subset of the total bitstream generated by the encoder to reconstruct and display textures, images and video objects at reduced spatial resolution. A maximum of 11 levels of spatial scalability are supported in so-called 'fine-granularity scalability', for video as well as textures and still images.

- **Temporal scalability** allows decoders to decode a subset of the total bitstream generated by the encoder to reconstruct and display video at reduced temporal resolution. A maximum of three levels are supported.

- **Quality scalability** allows a bitstream to be parsed into a number of bitstream layers of different bitrate such that the combination of a subset of the layers can still be decoded into a meaningful signal. The bitstream parsing can occur either during transmission or at the decoder. The reconstructed quality is generally related to the number of layers used for decoding and reconstruction.

- **Fine Grain Scalability** – a combination of the above in fine grain steps, up to 11 steps.

The functionality of these scaling methods is desired for progressive coding of streaming video sent over heterogeneous networks such as the Internet. It also allows for scaling down the video when the bandwidth is not the limiting factor, but where the receiving device cannot display the full resolution or quality of the encoded video stream. The server in this case reduces the quality while maintaining equal temporal and spatial resolution so as to keep the size of the displayed video unchanged. For scalability in the sense of streaming video over the Internet, the main tools used are Fine Granularity Scalability (FGS) in combination with Temporal Scalability [40]. These two types of scalability address the variety of problems encountered in delivering video over the Internet. FGS allows for a single compressed video stream to be delivered over different channels (such as the Internet or wireless) with a wide range of bit rates. This is expected to provide the best viewer experience under the constantly changing channel conditions. In other words, it makes compressed digital video behave similarly to analog video in terms of robustness while maintaining all the advantages of digital video [40].

2.2.2 Transport of MPEG-4 Coded Video

MPEG-4 was designed to be open in regards to transport protocols. MPEG-4 content can be carried over many different transport layers, and it is also easy to switch between them. Each video stream contains a set of descriptors used as configuration information. This information is used to determine the required decoder resources, and to determine the precision of the timing information that is embedded into the stream. Furthermore, the descriptors may carry hints to the QoS level it requests for transmission (e.g., maximum bit rate, bit error rate, and priority).

The Delivery Multimedia Integration Framework (DMIF) is a new transport framework that is used for interfacing MPEG-4 media sources to the various network transport systems. DMIF is a session protocol designed specifically for the management of multimedia streaming over generic delivery technologies [40]. Built into the specifications for the DMIF are hints/ways on how to perform such tasks over the different network types such as the Internet. DMIF relies on an application interface (DMIF-Application Interface (DAI)) to translate the messages between the MPEG-4 media stream and the particular transport network on which the video is being streamed.

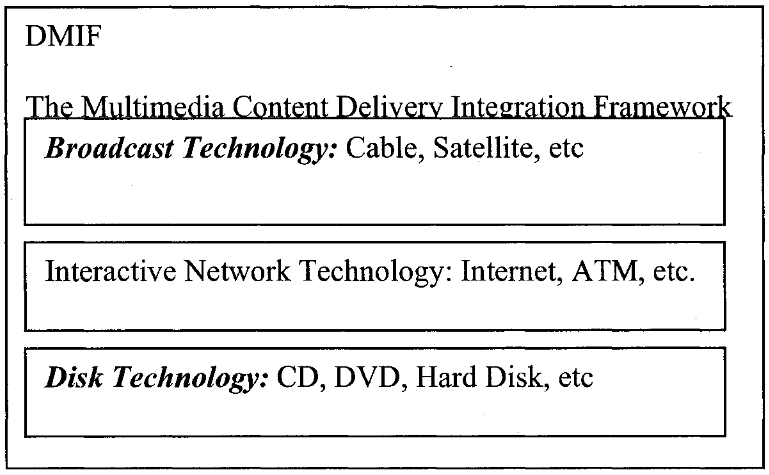
DMIF
The Multimedia Content Delivery Integration Framework
Broadcast Technology: Cable, Satellite, etc
Interactive Network Technology: Internet, ATM, etc.
Disk Technology: CD, DVD, Hard Disk, etc

Figure 2.2: DMIF addresses the delivery of MPEG-4 over all technologies

Therefore, applications that use the DMIF protocol for communication do not have to be concerned with the type of network that they are operating on. Instead the DAI communicates with the underlying communication channel, all the while presenting a simple and consistent interface to the various applications that are running. The DMIF Layer automatically determines whether a particular service is supposed to be provided by a remote server on a particular network. This basic concept is outlined in Figure 2.2. The DMIF interface is transparent to the streaming application. The local streaming application interacts with only the DAI which translates the requests into specific messages that can be delivered to the remote application taking care of all the intricate details needed for successful sending and receiving of data. The major features of DMIF are outlined below [40]:

- ***Support for all kinds of networks, including mobile networks:*** In conjunction with ITU-T, the H.245 specification has been extended (H.245v6) to include support to MPEG-4 Systems; the DMIF specification provides the appropriate walkthrough and mapping to H.245 signals. Mobile terminals can now use MPEG-4 System features such as BIFS and OD streams, although with some limitation.

- ***QoS Monitoring:*** DMIF knows the concept of monitoring the Quality of Service actually delivered by a network. Within the model there are three allowed modes of QoS monitoring: continuous monitoring, specific queries, and QoS violation notification.

- ***User Commands with ACK:*** The DMIF model allows peer applications to exchange user messages of any kind (including stream control messages). DMIF supports acknowledgment messages.

- ***Management of MPEG-4 Sync Layer information:*** The DMIF model allows applications to exchange application-specific data with the DMIF layer.

- ***DAI syntax in C language:*** DMIF includes an informative annex that gives C/C++ syntax for the DMIF Application Interface, as a recommended API syntax.

2.2.3 Timely Delivery

It is mainly through the DMIF that timely delivery of packets can be ensured. Since the delivery mechanism is not known to the application, it is assumed that a concurrent delivery of a number of streams with a constant end-to-end delay is achieved. The delivery layer, without the application's intervention, must therefore handle any jitters or changes to the determined values that are periodically changed at regular intervals. Time stamps are encoded into the object streams to ensure correct playback of the video stream. On the server side of a client-server system, the server monitors the space available in the client's buffers and schedules, ahead of time, delivery of critical data. The responsibility of this time stamping as well as object clock references falls upon the sync layer. The sync layer provides the information needed to be shared between the other layers especially the delivery layer to ensure timely delivery of the elementary streams. This in turn ensures the smooth and continuous playback of streaming video.

2.3 Overview of Congestion Control Algorithms

The dominant transport protocol in the Internet is TCP and the stability of the Internet depends on its end-to-end congestion control mechanism. Ideally, in TCP every packet transmitted must be acknowledged by the receiver to ensure data integrity. Any unacknowledged packets are assumed to be lost and the sender re-transmits copies until the transmitter receives an acknowledgement from the receiver or the number of retransmissions exceeds a certain threshold in which case the session is stopped. However, in order to improve the performance of TCP, a window-based mechanism is used where a number of packets called a window, can be transmitted before the sender receives an acknowledgement from the receiver. The size of this window can be increased or decreased depending on the congestion condition of the network. TCP's congestion control algorithm, known as Additive-Increase/Multiplicative-Decrease (AIMD), is a linear congestion control algorithm where the sending rate is controlled by reducing the congestion window every time a packet is lost and increased by one packet otherwise. Although the congestion control mechanism used by the TCP protocol is appropriate for certain applications such as bulk data transfer, it may not be suitable for timing constrained real-time data applications. The AIMD algorithm is too sensitive for real-time applications and may grossly impair the quality of the received signal. The high sensitivity of AIMD in this context refers to the fact that responding to an indication of congestion by reducing the sending rate on every single packet loss may be excessive. Also, the increased end-to-end delays and delay variations make this mechanism undesirable for streaming real-time audio and video. This has necessitated the need to explore different viable mechanisms to provide smoother congestion control over a longer period of time for real-time applications.

Congestion control algorithms can be broadly divided into two categories: window-based and rate-based. The AIMD algorithm as implemented in TCP is a window-based congestion control algorithm. Rate-based congestion control algorithms, on the other hand, typically control the rate directly although windows can still be translated into rates. Rate based control is naturally desirable in continuous media applications as the quantity of interest is the transmission rate. Most rate-based algorithms model the network using equations and are therefore commonly referred to as model or equation based congestion control algorithms. The implementation of the various congestion control algorithms varies between multicast and unicast applications. Efficient implementation of Multicast congestion control usually requires support from the network routers whereas unicast congestion control can and is usually implemented end-to-end. Of particular interest recently, is the receiver-based congestion control, which takes the congestion detection functions and rate estimation workload from the server to the client.

2.4 Rate-based Congestion Control

The aim of rate-based congestion control is to maintain a steady sending rate while remaining responsive to network congestion. It does not aggressively search for and occupy available bandwidth like the AIMD mechanism does. Rate-based control commonly uses a control equation to determine a maximum acceptable sending rate, as a function of the recent loss event rate. On the contrary, AIMD congestion control completely decreases the sending rate in the event of a data packet loss. Rate-based congestion control is also known as equation-based or model-based congestion control and the term equation-based congestion, where used in this thesis, refers to rate-based congestion control. In the equation-based mechanism the receiver provides the sender with information about the data packet loss rate so that the sender can correctly recalculate and readjust its sending rate to reduce fluctuations in the overall bit rate of the data being transmitted. Before specifying the control equation used to determine the sending rate, it should be noted that an application using a much more aggressive control algorithm than TCP could cause a starvation or an improper transmission of the TCP streams. The equation controlling the sending rate should provide a rate that is TCP-compatible. This means that a stream of data should not use more bandwidth than a TCP stream under similar network conditions. Many congestion control algorithms have, however, tended to be too sensitive in their effort to achieve TCP-friendliness. Long term TCP-friendliness, which balances TCP-fairness with quality fluctuation problems arising out of rate changes is the viable solution for real-time applications. Equation 2.1 is used to calculate an upper bound sending rate T as a function of data packet size s, round-trip time R, steady-state loss event rate p, and the TCP retransmit timeout value t_{RTO} [42]:

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)} \tag{2.1}$$

It should be noted that for data streams that are not competing with TCP traffic and are isolated from TCP traffic, a different control equation could be used to determine the sending rate. However, this is beyond the scope of this thesis.

In this control equation, although the round trip time R can be calculated at either the sender or at the receiver, the loss event rate p can best be determined at the receiver. The packet size s is measured as the observed average packet size and retransmit timeout value t_{RTO} can be estimated from the round trip time R (see Eqn. 2.2). In the implementation of this algorithm, the receiver may calculate the sending rate T and send it back to the sender (so that the sender can adjust its rate) or it may just send back the loss event rate p and force the sender to calculate the sending rate. Based on these parameters, the sender adjusts its rate.

In the following, the main design principles of equation-based congestion control which differentiate it from TCP behaviour are listed [42]:
Do not aggressively seek out available bandwidth. Increase the sending rate slowly in response to a decrease in the loss event rate.

Do not reduce the sending rate in half in response to a single loss event. The receiver should report feedback to the sender at least once per round-trip time if it has received any packets in that interval.

If the sender has not received feedback after several round-trip times, then the sender should reduce its sending rate, and ultimately stop sending altogether.

2.4.1 Sender Functionality

The sender calculates the round-trip time R and the retransmit timeout t_{RTO} values. The sender measures the round-trip time by based on feedback sent by the receiver. When the receiver sends feedback, it sends the sequence number of the most recent packet and the time elapsed since that packet was received. Based on this information from the receiver, the sender can determine the round-trip time. The sender can derive the retransmit timeout value t_{RTO} using the following algorithm typically used by TCP [42]:

$$t_{RTO} = SRTT + 4 * RTT_{\mathrm{var}} \tag{2.2}$$

Here, RTT_{var} is the variance of RTT and $SRTT$ is the round-trip time estimate. However, the modified algorithm, $t_{RTO} = 4*R$, provides considerable fairness with TCP.

2.4.2 Receiver Functionality

In the equation-based congestion control scheme, the receiver provides the sender with information to calculate the round-trip time and it also sends back another very important parameter, which is the loss event rate. If the loss event rate is calculated over a short period of time, a signal can respond very quickly to changes in the available bandwidth. Although this signal reacts faster to changes, it will be noisier than the one where the loss event rate has been measured over a long period of time. Here are guidelines for determining the loss event rate are [42]:

- The estimated loss event rate should track relatively smoothly in an environment with a stable steady-state loss event rate.

- The estimated loss rate should measure the loss event rate rather than the packet loss rate, where a loss event can consist of several packets lost within a round-trip time.

- The estimated loss event rate should respond strongly to loss events in several successive round-trip times.

- The estimated loss event rate should increase only in response to a new loss event.

- The estimated loss event rate should decrease only in response to a new loss interval that is longer than the previously calculated average, or a sufficiently long interval since the last loss event.

2.4.3 Limitations of Equation-Based Congestion Control

Although this congestion control mechanism reduces the oscillations in the sending rate, and it is judged to be fairer than the TCP protocol, it is not completely ideal. The protocol adjusts its sending rate at fixed time intervals. As a consequence, the performance at low time scales is poor. Also, since the loss rate is calculated at fixed time intervals, changes in the RTT and sending rate may lead to lost packets within these time intervals until these values are updated correctly.

2.5 WINDOW-BASED CONGESTION CONTROL ALGORITHMS

In window-based congestion control the transmission load is controlled by adjusting the size of a number of packets that can be transmitted before receiving an acknowledgement from the receiver. Window-based congestion control algorithms are generally referred to as binomial algorithms because they are described by a set of two equations: one for transmission rate increase and the other for transmission rate decrease.

Binomial congestion control algorithms provide a class of congestion control algorithms, for TCP and other Internet transport protocols, which are non-linear. In the previous section, equation-based mechanisms were considered and they were judged to be more effective than linear congestion control techniques (i.e. such as AIMD) because they provide a smoother congestion control without drastically adjusting sending rates. This is accomplished by tracking loss rates and by using a control equation to determine a more efficient sending rate. However, as considered in the previous section, using loss rates at fixed time intervals makes protocols using equation-based algorithms vulnerable when RTT and sending rate values change. Binomial congestion control algorithms enable the use of increase and decrease concepts without having the need to track loss rates. The binomial congestion control algorithms can be described by the following equations [43]:

$$I : w_{t+R} \leftarrow w_t + \alpha / w_t^{\,k} ; \alpha > 0 \tag{2.3}$$

$$D : w_{t+R} \leftarrow w_t - \beta w_t^{\,l} ; 0 < \beta < 1 \tag{2.4}$$

Here, *I* refers to the case when the sender increases the window size as a result of the receipt of at least one acknowledgement to a packet transmitted in the previous window. *D* refers to the decrease in window size on detection of congestion by the sender; w_t is the window size at time *t*, *R* is the RTT of the stream, alpha is the increase constant, β is the decrease constant, and *k* and *l* are constants. Generally, $\alpha = 1$ and $\mu = 0.5$. This set of equations can be modified (even just modifications to *k* and *l*) to model several different and important congestion control algorithms. For instance, if the k = 0 and $l = 1$, the equations become the classic linear congestion control algorithm known as AIMD. For $k = -1$ and $l = 1$, the equations model the Multiplicative-Increase/Multiplicative-Decrease (MIMD) algorithm, and for $k = l = 0$, the equations model the Additive-Increase/Additive-Decrease (AIAD). As a result, using these equations, all important linear control algorithms can be modeled. These algorithms are known as binomial control algorithms because two algebraic expressions are used in the control process; an increase equation and a decrease equation.

The k and *l* parameters of these equations can be varied to create an algorithm that will handle congestion as desired. A low *k* parameter represents aggressive probing for additional bandwidth, while the l parameter represents the conservativeness of the congestion response. It has been shown [43] that a binomial algorithm is TCP friendly only if $k+l=1$. The existence of this rule demonstrates that there is a trade-off between the aggressiveness of probing for available bandwidth and the responsiveness of window reduction. Also, if $k+l \leq -1$, then the congestion control mechanism is considered unstable because it will not reduce the sending rate in response to the loss of data packets [43].

An important variant of the binomial algorithm is the Square Root Congestion Control Algorithm. This is the case where $k=l=0.5$. As expected, based on the results of previous experimentation [43], the square root algorithm converges faster to fairness and is fairer to TCP streams in smaller time scales than AIMD. This is because the oscillations in sending rate are considerably reduced due to the fact that the sending rate is not decreased by a factor of 2 every time there is a packet loss. Another key characteristic of the square root algorithm as well as almost any other reasonable protocol is the *slow start* mechanism. This is to ensure that the sending rate of a data stream converges to a fair and network-friendly rate, relatively quickly. Slow start is especially useful in a scenario where there are many connections to the network. Sessions using the square root algorithm will be able to occupy a fair share of the bandwidth in the presence of other square root or TCP streams.

2.6 BUFFER UTILIZATION IN STREAMING APPLICATIONS

Streaming applications have a strict time constraint that variations in network delays can greatly affect the perception of the rendered media. To address this problem receivers usually incorporate some form of buffering. This receiver buffer is used to smooth out delay jitter. During increased delay periods the receiver buffer provides the required data for rendering without having breaks.

Although the receiver buffer was initially meant to be a media smoothing tool that allows the delay variations to be eliminated from the rendered media, it has also been exploited in other cases. Dejian et al [44] have used the buffer occupancy to implement algorithms for rate adjustment. Here, the observed receiver and server buffer occupancies are used to implement algorithms for scaling video and scheduling data transmission. In this work, availability of an underlying end to end congestion control algorithm is assumed. When the receiver buffer falls to a certain threshold level, the link is considered to be underutilized and therefore the transmission rate is increased. Likewise when the buffer tends to overfill the algorithm adjusts the rate downwards. This implementation obviously assumes a fixed buffer size. The main problem with this approach is that if the buffer occupancy is falling due to congestion effects, then it breaks down because whereas the underlying congestion control may have called for a reduction in transmission rate the rate controller will be assuming that the link has more capacity than is being exploited. This algorithm also uses empirical parameters whose values are arbitrarily chosen when implementing rate control.

Cheng and Lihao [45] use a control theoretic approach to integrate rate control with an underlying congestion control algorithm. They use the buffer occupancy in their strategy and also endeavor to maintain a constant buffer level. The problem with this technique is that the congestion control algorithm would imply that the data rate be reduced when the buffer occupancy reduces. Therefore attempting to maintain a certain buffer level becomes a situation of two opposites.

In [46], Wenjun and Magda implement resource renegotiation algorithms for VBR video transport using observed buffer level. Of course this scenario is different from the best effort situation because it assumes that the underlying network supports QoS and is therefore capable of providing a bit more resources if needed. It is an implementation of RSVP. But still gives an insight into the possible uses of the receiver buffer.

Cuetos et al [47] implemented a receiver buffer-driven system to stream FGS video over TCP. In this work, TCP is the underlying transport protocol and therefore its congestion control that is not very suitable for continuous media is employed. The buffer level is used to adjust the data rate based on the observed effects of TCP's congestion control. Again this work focuses more on how to adjust the transmission rate to match that imposed by the underlying congestion control scheme. It therefore does not introduce any new congestion control scheme.

An accumulation based congestion control model has been developed by Xia et al [49]. In this work, the authors use this model to estimate the backlog of packets held in the network buffers during congestion. They use a fluid model to estimate the backlog as a time-shifted, distributed sum of the queue contributions of a flow at a set of FIFO routers in its path. This work is an effort to improve on the capacity estimation of the TCP protocol and its results culminate in a new version of TCP called Monaco. It only goes to show how useful it is to monitor buffer levels as an indicator of the congestion condition of the network.

In conferencing applications such as voice over IP the playout buffer has been used to adapt the playout delay to the delay experienced by packets in a talkspurt. In [50], [51], the delay experienced by packets of a talkspurt is distributed among the packets of the talkspurt to create a uniform playout rate.

Chi-Woon and Liew [52] have designed a video streaming system that uses receiver buffer occupancy and loss rate to detect occurrence of congestion. The available channel bandwidth is then estimated by measuring the incoming data rate at the receiver. The buffer threshold is determined using the equation below

$$\sigma_{\min} > B\left(\frac{\lambda_r}{\lambda + l_s} - 1\right) + \sigma_m \tag{2.5}$$

where $\sigma_{\min}$ is the threshold buffer occupancy in frames, λ_r is the packet arrival rate at the receiver, l_s is the packet loss rate and σ_m is a safety margin. B is the backlog of packets accumulating in the network. Here we see that the transmission rate is measured directly and buffer occupancy is only used to detect congestion.

Kanakia et al [53] have also used buffer occupancy at the bottleneck nodes of the network to estimate the available bandwidth for a given connection and therefore adjust the transmission rate (by adjusting the frame rate) accordingly. In this work, the algorithm estimates the number of queued packets of a given connection at the bottleneck switch along the path of the connection. In fact the switches cooperate with source nodes by passing back information regarding the volume of queued packets for the connection. Accordingly, the sources adjust the transmission rate using the control function below

$$\lambda_n = \begin{cases} \lambda_{n-1} + \delta & \text{if } x_{n-k} = 0 \\ \hat{\mu}_n + \dfrac{x^* - \hat{x}_n}{gain \times F} & \text{otherwise} \end{cases} \tag{2.6}$$

where λ_n is the transmission rate for frame n, x^* is the target buffer occupancy, $\hat{x}_n$ is the observed mean buffer occupancy, $\hat{\mu}_n$ is the mean service rate at the bottleneck node queue, *gain* and δ are tuning parameters and $1/F$ is the fixed rate of frame display at the receiver. The rate of frames displayed at the receiver is fixed and any missing frames are replaced with dummies. This research assumes intelligence in the network where bottleneck nodes can identify themselves and can also pass back the queue information relevant to the relevant connections. Of course implementation of this scheme can have scalability issues.

In [54], Bajic et al argue that because congestion control is implemented at the source, more packet losses occur at the source than within the network. The authors claim that this packet loss is due to buffer overflows at the sender queues. Based on this argument they design an end to end streaming system that integrates source buffer management with congestion control. Their strategy uses an underlying congestion control scheme, the binomial scheme, a random packet dropping algorithm where packets are dropped from a set of low priority packets to prevent the sender buffer from overflowing. This scheme demonstrates the need to integrate buffer management with congestion control schemes which are responsible for adjusting the transmission rate.

Having reviewed all this work, we observe that the buffer is an important component of a streaming system. In much of the literature above, however, it has either just been used for its traditional purpose of play-out smoothing or used in adjusting the sending rate to that set by an underlying congestion control scheme. In this thesis we are proposing to integrate the congestion control with the observed buffer occupancy. For it is at the receiver buffer that we are able to observe the effect of congestion and then determine how much response is needed to adapt the stream. We use the fall in the receiver buffer occupancy as a trigger for occurrence of congestion and then estimate the effect of congestion on the relevant connection from the buffer deficit. This gives a close estimate of the price paid by the connection and hence helps in estimating network capacity available to the connection.

2.7 SUMMARY

In this chapter we have reviewed the architecture of a video streaming system. We have discussed among other issues, the MPEG-4 video coding standard which has introduced a new set of features useful for streaming video over unreliable bandwidth fluctuating networks. The focus of this is work is on end-to-end congestion control for Internet video streaming and we have presented two of the main techniques of implementing congestion control for video streaming. We have noted that existing congestion control techniques are largely based on packet loss, which is itself detrimental to video communication. We have also seen how the buffer, either at the receiver, transmitter or network nodes, is of great importance when it comes to managing traffic in streaming applications.

Chapter 3

Proposed Congestion Control Scheme

We have presented a review of streaming technology and also reviewed congestion control techniques in chapter 2. In the foregoing discussion, we also looked at some useful properties of buffers in managing streaming applications. In this chapter, we present an efficient congestion control algorithm for internet video streaming. The algorithm addresses deficiencies of existing algorithms by paying due consideration to the requirements of video communication while at the same time preserving the TCP-friendly requirement for all Internet traffic given that TCP is the predominant form of traffic in the Internet. This thesis concentrates on scalable MPEG-4 video, but the algorithm can be easily used in all scalable video streaming applications. Scalable video coding enables a streaming application to adjust the data transmission rate in response to congestion by adjusting the amount of video data transmitted. This is because decoding scalable video is possible with just a fraction of the data. Fine Granular Scalability (FGS, discussed in chapter 2) as implemented in MPEG-4 enables a video session to be continuous, only occasionally adjusting the quality or frame rate by dropping some of the video information in response to adverse network conditions that require a reduction in the session's transmission rate.

3.1 OVERVIEW

Most of the existing congestion control algorithms have focussed too much on TCP-friendliness without sufficiently addressing the constraining requirements of video communication. Implementing an effective congestion control algorithm for real-time video streaming entails balancing the timeliness and packet loss requirements of video communication against the general flow/congestion control requirements of the underlying transport network.

Window based congestion control algorithms tend to be too data-centric, responding to congestion on detection of packet losses yet such losses are detrimental to video communication. Although error control techniques have been developed to address minimal packet losses, video communication is greatly affected by such losses and a flow/congestion control algorithm that is triggered by packet losses is not the best option. Because video communication is rate based, using window based congestion control algorithms may not be appropriate enough in estimating the transmission rate for the video session. The proposed algorithm uses rate based binomial equations to address this problem. To avoid the dependence on packet losses as a trigger of congestion response, we use the buffer level at the receiver in addition to the detection of packet losses as an indication of the condition of the network. Incorporating the receiver buffer level in the congestion control scheme has the advantage of allowing congestion to be detected early enough even before packets get dropped by the network. This is because congestion leads to an increase in network delays before packets get dropped and the late arrival of packets causes a drop in the buffer level. Early detection of congestion allows the stream to prevent packet losses rather than wait for the adverse effects that may result from such losses.

Although some rate based congestion control algorithms have tried to address the problem of working directly with transmission rates, they have also had other disadvantages which affect their performance. Rate based congestion control algorithms, also called equation or model-based algorithms, have been developed from statistical models of a typical AIMD TCP session and are therefore mostly focussed on TCP friendliness without addressing the requirements of video communication. They also use the loss rate to estimate the appropriate transmission rate, an approach whose flaws have been highlighted in the discussion above. Moreover, estimating the loss rate is still an unsolved problem – it is not clear which, of packet loss rate and loss event rate is the most appropriate quantity to use. The other known problem of these equation based congestion control algorithms is the estimation of transmission rate at fixed time intervals making the stream susceptible to possible severe packet losses during the interval before any adjustments can be made to the rate. Again this is because these models are based on the behaviour of TCP's AIMD, a data centric congestion control algorithm, unsuitable for real-time video communication. The models consider packet re-transmissions which are not typical of video communication. In that respect, they end up giving the worst case scenario thereby under-estimating the acceptable throughput for the video session.

A reduction in buffer level is caused by either excessive delays or packet losses, both effects of congestion. It is therefore appropriate to use the reduction in buffer level as a congestion indicator. Additionally the relative magnitude of the reduction can be used as a measure of how much the session is being affected by the adverse network conditions and therefore an indication of how much the transmission rate needs to be adjusted to accommodate the obtaining situation. This ensures that the algorithm is TCP friendly only that the response to congestion is more considerate of the sensitive nature of video than the data-centric congestion control algorithms in use today. In the proposed congestion control algorithm, we also take TCP-friendliness to refer not to the short term per loss event rate adjustment, but rather on a long term basis giving due consideration to the past as well as present condition of the network. This allows the algorithm to be more responsive when the session is transmitting at higher rates than at the low rates, ensuring that the transmission rate is reduced by a smaller proportion at lower rates than at higher rates. On the other hand, when a session is recovering from a congestion condition, it is allowed to be more aggressive in acquiring more bandwidth from lower transmission rates than from relatively higher rates. This is to ensure fairness by allowing sessions that have been more adversely affected by congestion to acquire more of the available bandwidth than those that have been relatively less affected.

3.2 RECEIVER BASED CONGESTION CONTROL USING BUFFER OCCUPANCY

The proposed congestion control algorithm is based on the buffer occupancy at the receiver for congestion detection and congestion response. It also monitors the packet loss rate but any action taken in response to such losses is determined by how much the losses have affected the buffer occupancy. In this congestion control scheme, buffer occupancy is measured both in terms of media temporal length (in time units) and media size in bits. The use of temporal length is motivated by the fact that video is time rated in unequal data entities – frames and therefore working at this level ensures that frames are scheduled for transmission and rendered not later than their play out time. At the start of the video session the client determines the temporal length of media to be pre-buffered before play starts. This technique, also known as pre-roll, is normally used for smoothing out the effects of delay jitter due to the usually variable transmission delay incurred by the video packets as they traverse the packet network from the server to the client. Under perfect network conditions, the buffer should maintain a certain average occupancy. When congestion occurs, increased delays and packet losses affect the number of video packets that get through to the client. This results in the buffer occupancy to fall, which in adverse conditions, leads to buffer under-run as the service rate (the rate at which packets are extracted from the buffer for play out) exceeds the arrival rate (the rate which media packets are received at the client). Therefore, monitoring the buffer occupancy together with packet loss enables a video session to detect the incidence of congestion early enough and to take remedial measures before the video play out becomes discontinuous due to missing video packets or their late arrival.

The congestion control strategy can be looked at as a feedback system represented in the schematic below.

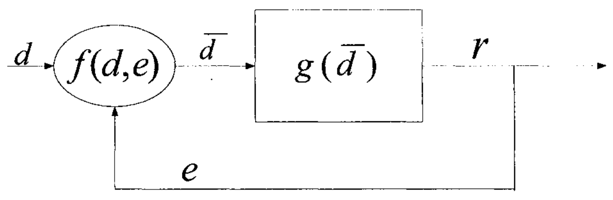

Figure 3.1: System representation of the congestion control algorithm

In the above schematic;

d represents the video data available for transmission

$\bar{d}$ is the smoothed data scheduled for transmission and $\bar{d} \leq d$

r is the received data at the client and $r \leq \bar{d}$

e is the feedback information from the client and reflects the buffer average occupancy in bits and the proportional temporal buffer occupancy

$g(\bar{d})$ is the system function representing the end to end network connection effects on the transmitted data

$f(d,e)$ is the rate control function that uses the feedback information to determine how much and when video data is to be transmitted. It is a binary function with a rate increase equation and a rate decrease equation.

$$f(d,e) = \begin{cases} f_I \\ f_D \end{cases} \tag{3.1}$$

f_I is the increase function that is employed to probe for available bandwidth when congestion has subsided while f_D is the decrease function executed when congestion is detected to adapt the video stream to the adverse network conditions.

3.2.1 Congestion Detection

Unlike with other schemes where congestion is detected by the resulting packet losses, in this scheme we use the buffer occupancy as the indicator of whether congestion is occurring or not. Detecting congestion based on packet losses ignores the increased delay effects that congestion causes, yet these delays are as damaging to the video as lost packets if they are excessive over an extended period of time. Detecting this trend early allows the system to make appropriate adjustments before the problem manifests itself to the viewer. Our scheme is able to do this by monitoring the receiver buffer to establish how much video data is available for transmission. If video data of length T_s seconds is to be pre-buffered, then the temporal buffer size is T_s. It is possible to set a threshold buffer length T_c (where $T_c < T_s$) below which congestion is presumed to be occurring. By detecting congestion while there is still some data available for rendering and making appropriate adjustments early enough, the system is able to minimize effects on the rendered video. Integrating congestion control with the receiver buffer also presents possibilities for implementing selective retransmission of important video packets if their loss is detected early. When such retransmissions are made, the transmission rate can also be adjusted downwards to accommodate the extra load.

As soon as the length of video available in the buffer falls to the threshold T_c, the algorithm takes the next step of determining how to respond to the condition. The session also responds to congestion if packet losses are detected that result in a significant reduction in the buffer occupancy. The appropriate buffer occupancy that triggers congestion response is T_{loss}.

$$T_c \leq T_{loss} < T_s \tag{3.2}$$

3.2.2 Congestion Response

The efficiency of a congestion control algorithm is determined by how it responds to congestion and of course its bandwidth probing capabilities and; how all these impact the viewer's experience. In addition to setting the temporal length of video available in the buffer, the client monitors the size of the video content in bits and calculates the average buffer occupancy in bits. Therefore when congestion is detected the degree of congestion response depends on the buffer emptiness measured in bits. For FGS MPEG-4 video, the amount of enhancement layer to be transmitted is proportional to the ratio of the buffer occupancy in bits to the average full buffer size. This makes a lot of sense because the buffer emptiness represents the penalty that congestion has imposed on the video streaming session in question. The client measures the relevant quantities and periodically furnishes the server with this information through feedback packets. This scheme only adjusts the proportion of the enhancement layer data to allow the video session to remain continuous during congestion. If congestion reaches such a level that the base layer has to be truncated then the stream is simply cut off. This is because the base layer is coarsely encoded with the essential representation of the video pictures and any data dropped from this layer would result in a highly distorted picture.

The full buffer size in bits, B_{av_size} can be determined as the amount of video data, in bits, spanning the temporal buffer length, T_s. Because video data units are usually of different size, the buffer size is constantly evaluated as a running average combining the previous average and the current evaluation. This quantity is measured by the server where data sizes are readily available. Using the video frame size as F_k bits for frame *k*, and *N* as the number of video frames in T_s seconds, the corresponding amount of video data in bits, B is given by

$$B = \sum_{k}^{N} F_k \tag{3.3}$$

At any instant *t*, the average buffer size can be determined as an average using the simple equation below

$$B_{av\_size_t} = \frac{B_{av\_size_{t-1}} + B_t}{2} \tag{3.4}$$

Of course initially, $B_{av\_size_{t=0}} = B_0$.

The observed average buffer occupancy, B_{av_obs} is determined as the average receiver buffer occupancy within the measuring interval T_{fb_k} defined in eqn. 3.10. This quantity measured at receiver. For each video data unit received during the measuring interval, the buffer occupancy is measured and an average over the interval is determined as

$$B_{av\_obs} = \tfrac{1}{n} \sum_{i=1}^{n} B_{obs\,i} \tag{3.5}$$

Here *n* is the number of samples in the measuring interval.

The rate adjustment coefficient *η*, can be calculated as follows

$$\eta = \frac{B_{av\_obs}}{B_{av\_size}} \tag{3.6}$$

Given the buffer temporal length the full buffer size in bits can be easily determined at the server by averaging the bit length of data spanning the specified temporal buffer length.

3.2.3 Server Side Operation

Rate adjustment, in this case, the amount of enhancement data to drop can be done at the temporal level or the SNR level. From subjective quality studies we learn that viewers prefer slower moving but high quality pictures to fast low quality pictures. Therefore, the SNR FGS layer is usually preferred over the FGS temporal enhancement layer. The server also includes a smoothing buffer that is used to optimally fit the buffered video data in the available bandwidth. Let this smoothing buffer be *1 second* of video data in length. The amount of video data scheduled for transmission, after effecting any reductions, is then uniformly distributed and transmitted over *1 second.* Let F_k be the size of any frame *k* in a selected *1 second* window. D_{txn} , amount of video data scheduled for transmission within *1 second* is given by

$$D_{txn} = \sum_{k=1}^{N} (F_{base_k} + \eta F_{enh_k}) \tag{3.7}$$

where

$$F_k = F_{base_k} + F_{enh_k} \tag{3.8}$$

F_{base_k} is the size of the base layer (in bits) of frame *k* and, F_{enh_k} is the size (in bits) of the enhancement layer of frame *k*.

Matching the transmission rate with the video play out rate and maintaining the transmission schedule at the server ensures that there will always be enough video data to be rendered at the client. Any drift is then easily captured through the receiver buffer emptiness and appropriate measures taken to restore the normal situation. After adjusting the data rate, the server also schedules the data to be transmitted taking into consideration the receiver buffer deficit to be made up. Therefore moments following the initial response, the server may momentarily reduce the amount of data per transmitted frame to achieve this.

Congestion recovery is determined from the fact that in the absence of congestion the receiver buffer length is consistently greater than the congestion threshold length. If this is observed over at least 3 feedback updates then the server begins restoring the data rate to the normal rate by gradually increasing the amount of transmitted video frame data. This process, also known as bandwidth probing, is done according to the following equation.

$$D_{txn} = \sum_{k=1}^{N} (F_{base_k} + [\delta + \eta(1-\delta)]F_{enh_k}) \qquad (3.9)$$

where δ is progressively increased in a sequence similar to 1/8, ¼, ½, ¾, 7/8, 1. This sequence represents the aggressiveness yet smooth bandwidth probing capabilities of the algorithm.

3.2.4 Client Operation

The client measures all the relevant data and is also responsible for determining the occurrence of congestion as explained in sections 3.1.1 and 3.1.2. The client is naturally suited for this task as the impact of congestion can be easily determined from the observable effects. The buffer occupancy in bits is a weighted average over the few most recent measurement rounds giving a higher weight to the most recent values in order to include the long term behaviour of the video session while giving more emphasis to the current condition. The client periodically sends the average buffer occupancy in bits and the proportional temporal occupancy to the server in feedback packets. This information is then used by the server to make transmission rate adjustments in response to the changing network conditions. The maximum feedback interval is determined from the temporal length of the receiver buffer and is adjusted downwards in proportion to the buffer emptiness to allow for less frequent updates under normal network conditions and more frequent updates when congestion is being experienced. Let T_{fb} be the maximum feedback interval, (which can be conveniently set to $T_s/2$) and T_{fb_k} be the feedback interval after interval k then,

$$T_{fb_k} = \eta T_{fb} \tag{3.10}$$

Frequent updates during congestion enable the session to quickly respond to any changes for the worse or indeed a quick recovery as soon as congestion subsides.

3.3 Performance of the Congestion Control Algorithm

The key issues to consider of an Internet congestion control algorithm are fairness to TCP traffic, aggressiveness in probing for available bandwidth, sensitivity to congestion and smoothness.

3.3.1 TCP Friendliness

Although many studies have developed conditions for TCP friendliness it should be observed that the analysis used in developing these conditions has been largely based on the AIMD algorithm, which is no longer predominant in the various types of TCP available today. In fact all TCP implementations have reduced their sensitivity by different degrees that they may themselves not be fair to each other let alone to AIMD. However, recent studies have redefined TCP friendliness as the ability of a congestion control algorithm to respond to the occurrence of congestion in the long term duration. The degree of back off should therefore be determined not only by the occurrence of congestion as is the case in AIMD, but take into consideration the requirements of the application being supported. This allows congestion control algorithms used in real time applications to spread their congestion response over a longer period and also to employ more appropriate means of detecting congestion than depending on loss of packets as the sole congestion indicator. It has been pointed out that congestion leads to increased queuing delays at the affected network nodes and loss of packets is just a breaking point, but one technique that has also been used to indicate the occurrence of congestion in traditional IP networks. This is changing in the newer routers. Therefore as far as TCP friendliness is concerned this algorithm is friendly to TCP traffic in as far as it is able to reduce the traffic load during congestion.

3.3.2 Smoothness

The integration of congestion control with media timing and the smoothing buffers ensures that any rate changes do not lead to adverse effects in the rendered video. Any response to congestion is managed with the timing requirements of the video under consideration to avoid situations of buffer under run which may cause annoying discontinuities. Smoothness is again a key issue when probing for available bandwidth when congestion subsides. The gradual restoration of the session to the required transmission rate is an effort to ensure that whereas the application needs to quickly acquire bandwidth and, this should happen at a rate that will not result in annoying oscillatory behaviour. Therefore the algorithm balances aggressiveness with smoothness which translates in higher subjective QoS to the viewer. It is annoying when the video show is constantly oscillating between different quality levels.

3.3.3 Aggressiveness and Sensitivity

As shown in section 3.1.3 the algorithm is aggressive in probing for bandwidth, but as discussed above this aggressiveness is in concert with the smooth quality variation requirements of video applications. The aggressiveness can be tuned by adjusting the sequence of δ in section 3.1.3. This congestion control algorithm is designed to detect congestion when it begins to cause unbearable transmission delays to the video packets. This is done to protect the application while at the same time being considerate to the rest of the communication sessions sharing the network. The sensitivity is therefore higher than that of traditional loss event based algorithms.

3.4 SUMMARY

In this chapter we have proposed a new congestion control algorithm based on the use of the receiver buffer for the detection of congestion. The effect of congestion on the video session is measured in the degree by which the receiver buffer is affected and any response to congestion is proportional to this. This ensures that the degree of rate change is in proportion to the observed throughput derived from the network. We have also shown that the algorithm is aggressive enough in probing for available bandwidth, and does so while taking the smoothness requirements of a video application into consideration.

Chapter 4

Performance of the Proposed Congestion Control Algorithm

This chapter presents a performance evaluation of the proposed congestion control scheme. The performance was evaluated by simulation using Network Simulator-2 (NS-2) [55], an increasingly popular open source network simulation package among researchers. The performance criteria considered included TCP-fairness, throughput efficiency for the video sessions and packet loss rate. In the following sections we present a description of the experimental setup followed by a discussion of the results.

4.1 Performance Measures

The most critical performance measure of a congestion control scheme in the Internet is its fairness to TCP traffic. Fairness to TCP is important because TCP, the predominant form of traffic in the Internet, employs a congestion control scheme(s) that prevents congestion collapse by adapting session traffic loads to the congestion condition of the network. In order to maintain network stability all other traffic is expected to behave almost similarly to TCP traffic, hence the notion of TCP-fairness requirement of Internet congestion control schemes. To measure fairness performance we compare the throughput of the video sessions to the throughput of the concurrent TCP sessions. We use the average throughput for this purpose.

While TCP fairness is important for the stability of the network, video communication is sensitive to variation of transmission rate and therefore efforts to make a congestion control scheme have to be appropriately balanced with the smoothness requirements of video. In order to avoid annoying effects to the viewer, a video should not be changing its quality very frequently. The design of the proposed congestion control algorithm ensures that any response to occurrence of congestion does not greatly impact the performance of the video session. To evaluate performance in this respect, and because the videos used in the experiment are variable rate, we use throughput efficiency by comparing the throughput of the video session with the throughput of an unencumbered video connection. This performance measure captures the smoothness and probing aggressiveness of the congestion control scheme.

The sensitivity of the congestion control algorithm in the advent of congestion is captured in how much it minimizes packet loss and also in how well it avoids damage to the video session. Rate oscillation is also captured in the standard deviation of the throughput for a connection and also the minima and maxima.

4.2 Description of the Experiment

As mentioned earlier, we use NS-2 to evaluate the performance of the proposed congestion control algorithm through a simulation experiment. NS-2 is an extendable C++ simulation package using TCL as scripting language to setup and define the simulation experiment. The various network protocols are plugged into the package as independent modules. Therefore in this study we had to build C++ modules for the algorithm.

Fig. 4.1 shows a layout of the experiment used to carry out the performance study of the congestion control algorithm. We simulated a network with 5 sessions all going through a bottleneck link. Two of them were UDP video streams and the others were all TCP applications. There are two network routers connected through a bottleneck link; router 1 uses the tail-drop queue management protocol (when the queuing buffer is full all packets at the end of the queue are dropped), while router 2 employs the fairer RED (Random Early Detection) protocol (if the queuing buffer reaches a certain threshold level packets are randomly dropped from all sessions). The rest are terminal nodes, with the ones on the left hand side as senders and the right hand side nodes as the receivers in a one-to-one arrangement. The drop tail queue presents the worst case scenario, typical of summed effect of a number of

Internet routers and links. The bottleneck link is 2 Mbits/s for the first experiment and 1 Mbits/s for the second experiment while all the other links are 5 Mbits/s. All data packets flow from left to right and only feedback/ACK packets run in the opposite direction. The video clients send feedback information to the video streaming server in feedback packets while ACK packets are used by the TCP sessions.

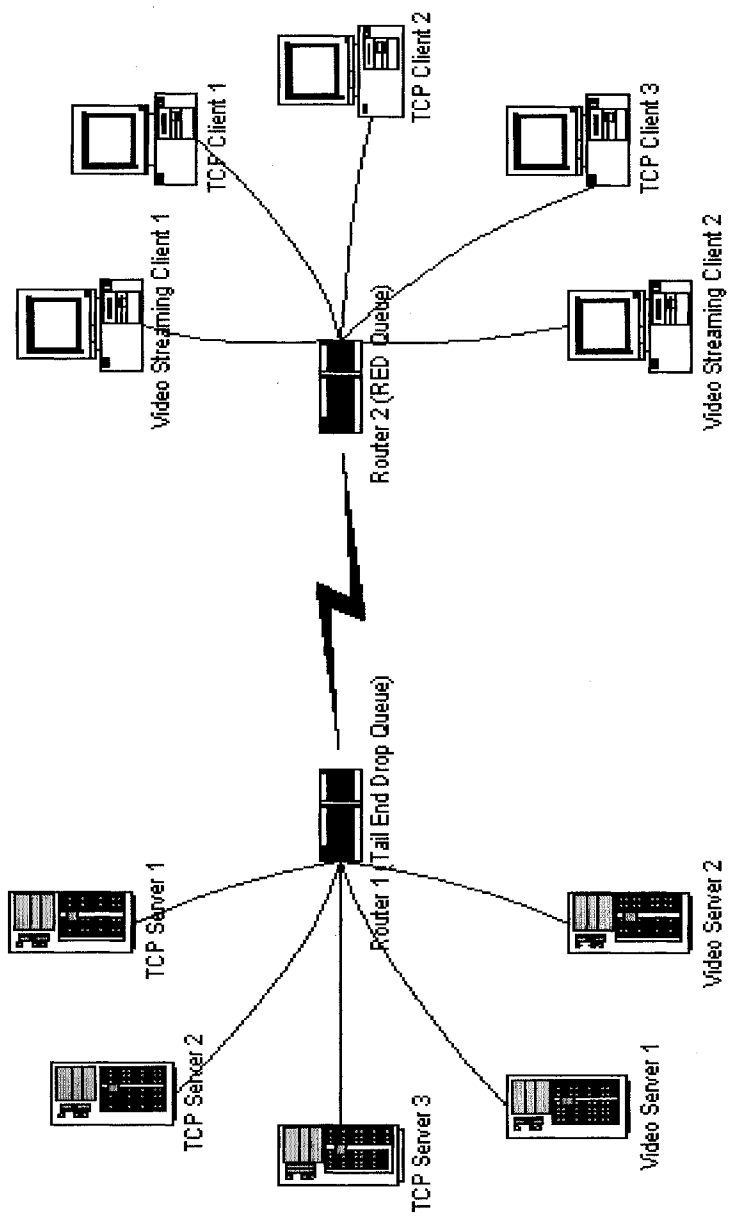

Figure 4.1: Layout of NS-2 Simulation Experiment

Fig. 4.2 shows a screen shot of the simulation animation.

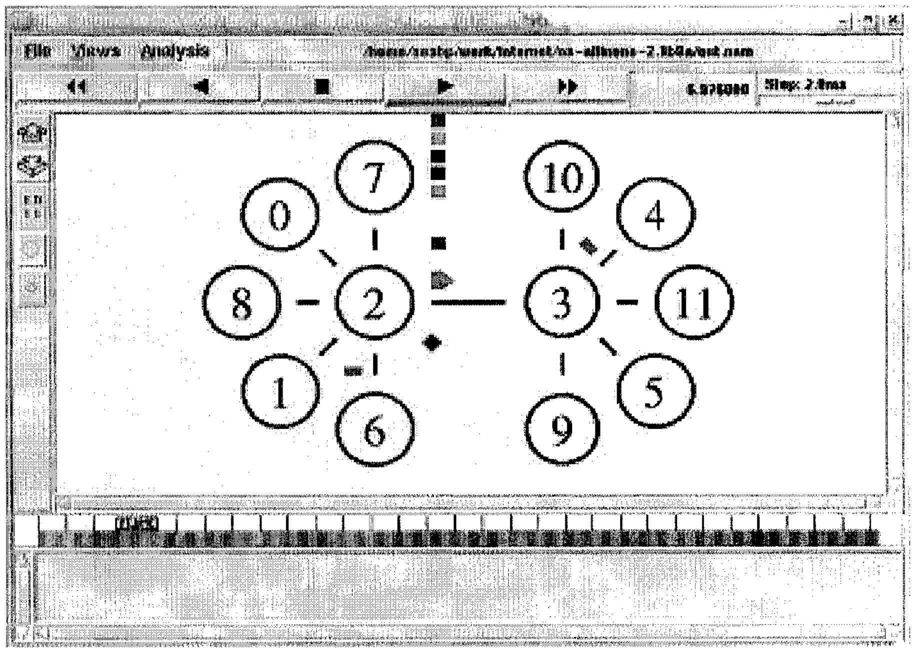

Figure 4.2: Animation of the NS-2 Simulation Experiment

4.3 RESULTS

As described above we carried out the experiments with five sessions. Two sessions were MPEG-4 video sessions using the proposed congestion control scheme while the other three were TCP data traffic sessions. We present graphical and statistical results of the simulation experiments.

Experiment when the bottleneck link is at 2 Mbits/s

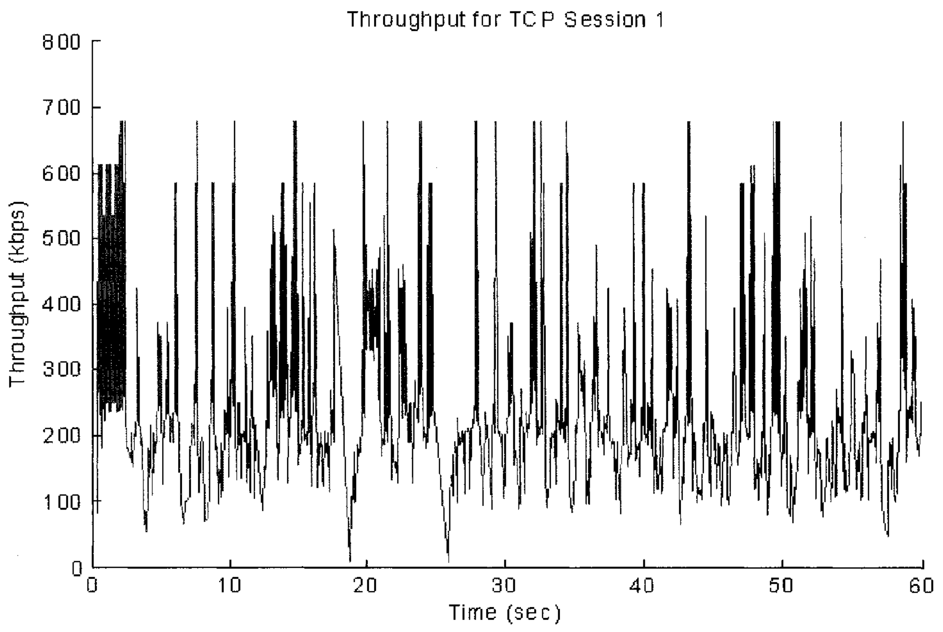

Figure 4.3: Throughput for data session 1 using TCP congestion control

Table 4.1: Statistics for TCP Data Session 1

Mean (kbps)	**Standard Deviation** (kbps)	**Minimum** (kbps)	**Maximum** (kbps)
263.684	138.649	8.650	678.168

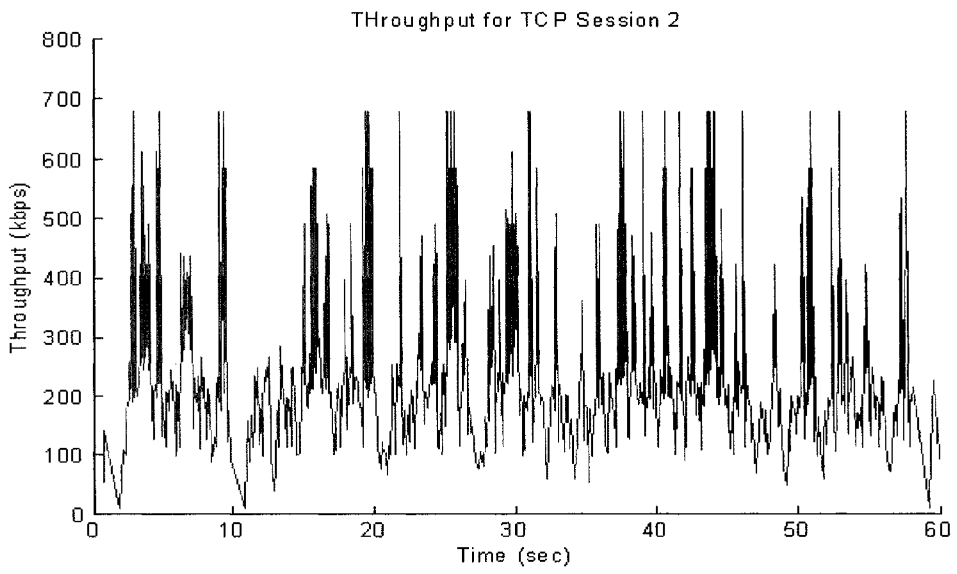

Figure 4.4: Throughput for data session 2 using TCP congestion control

Table 4.2: Statistics for the TCP Data Session 2

Mean (kbps)	**Standard Deviation (kbps)**	**Minimum (kbps)**	**Maximum (kbps)**
252.058	133.460	8.672	678.168

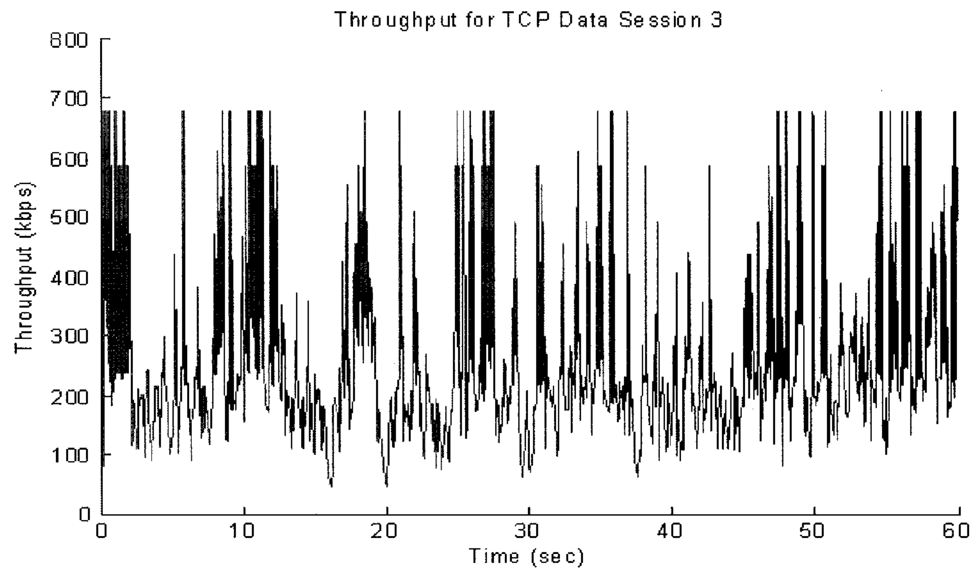

Figure 4.5: Throughput for data session 3 using TCP congestion control

Table 4.3: Statistics for TCP Data Session 3

Mean (kbps)	**Standard Deviation (kbps)**	**Minimum (kbps)**	**Maximum (kbps)**
285.130	149.357	44.479	678.168

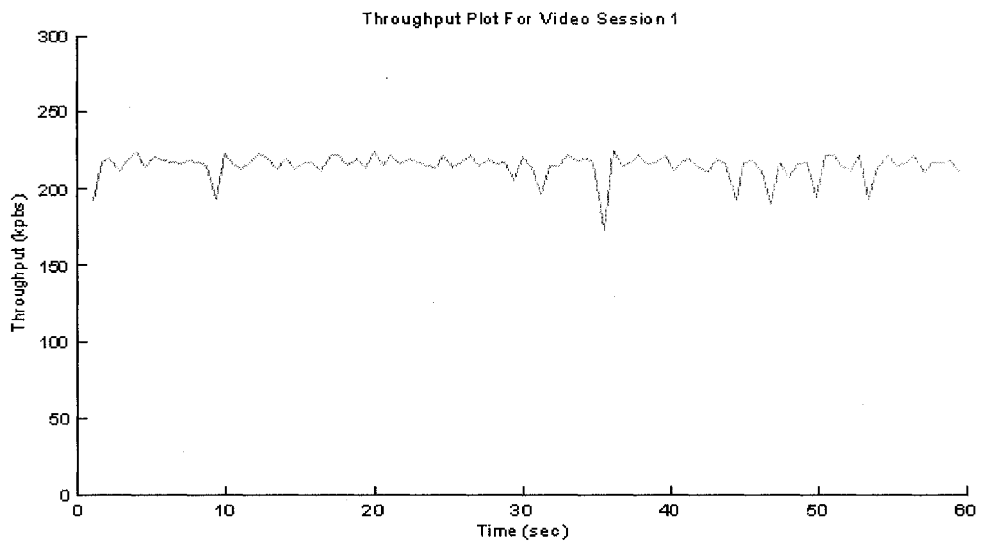

Figure 4.6: Video Session 1 Using the Proposed Congestion Control Scheme

Table 4.4: Statistics for Video Session 1 Using Proposed Congestion Control Scheme

Mean (kbps)	**Standard Deviation (kbps)**	**Minimum (kbps)**	**Maximum (kbps)**
215.720	7.919	173.950	225.979

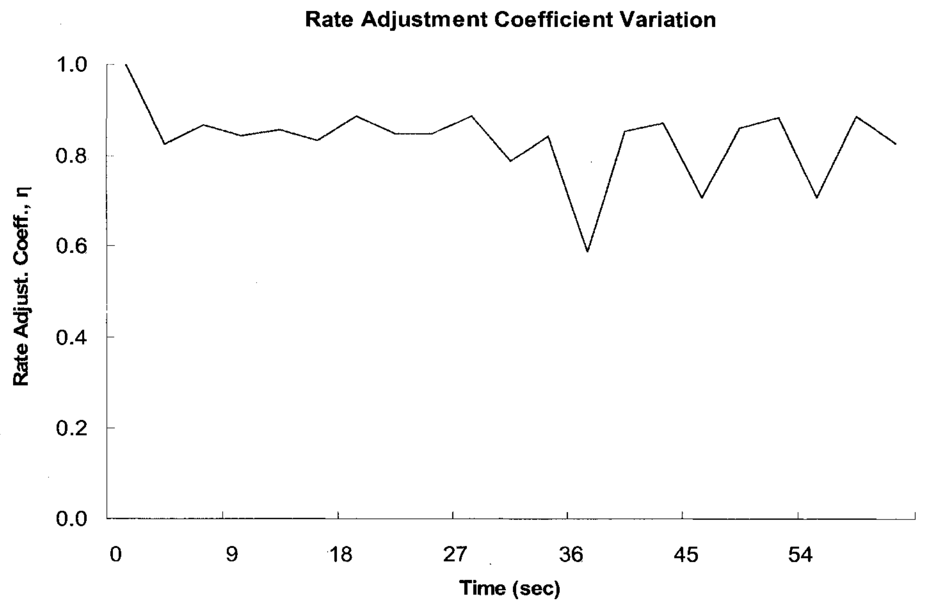

Figure 4.7: Rate Adjustment Coefficient Variation with time for Video Session 1

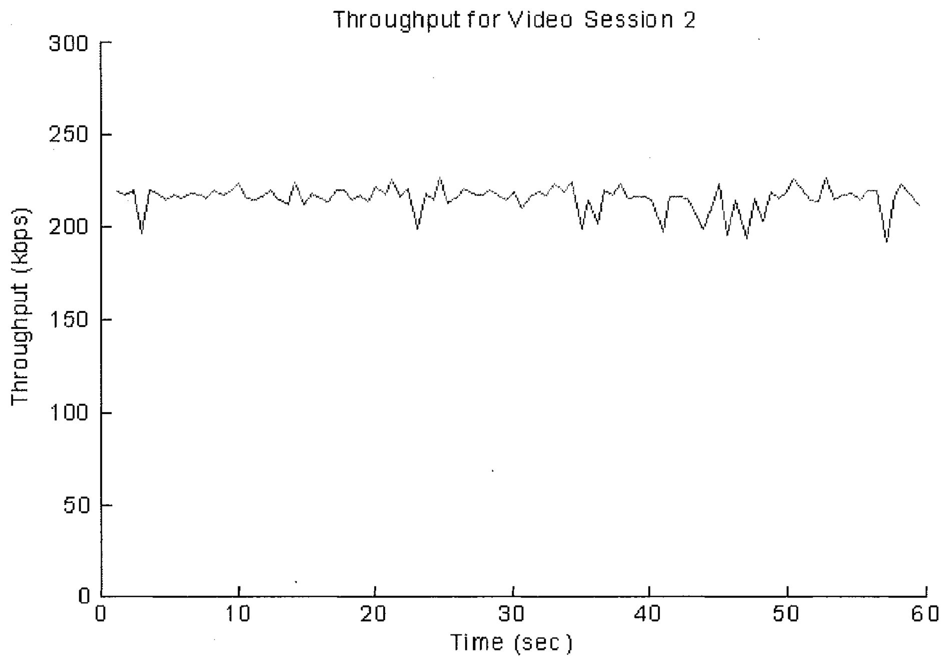

Figure 4.8: Video Session 2 Using the Proposed Congestion Control Scheme

Table 4.5: Statistics for Video Session 2 Using Proposed Congestion Control Scheme

Mean (kbps)	Standard Deviation (kbps)	Minimum (kbps)	Maximum (kbps)
215.354	6.985	192.212	226.286

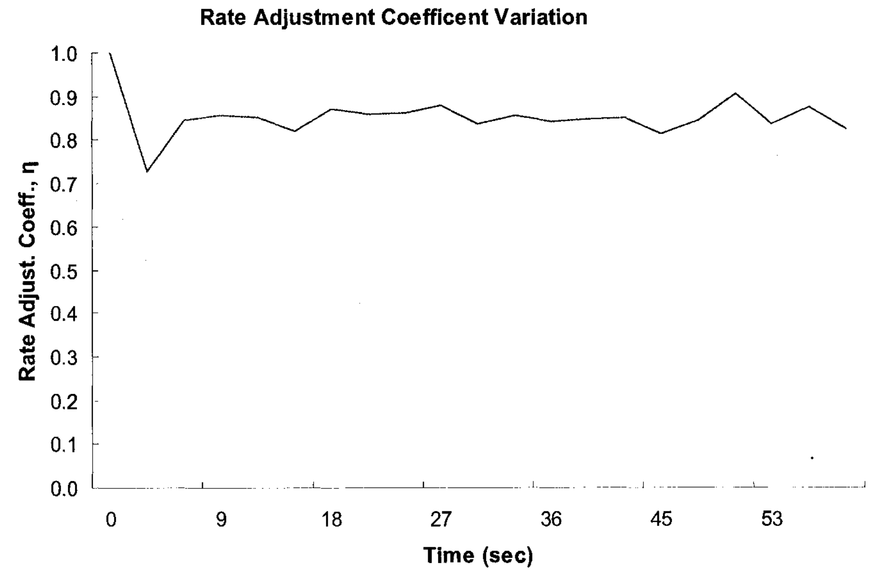

Figure 4.9: Rate Adjustment Coefficient Variation with time for Video Session 2

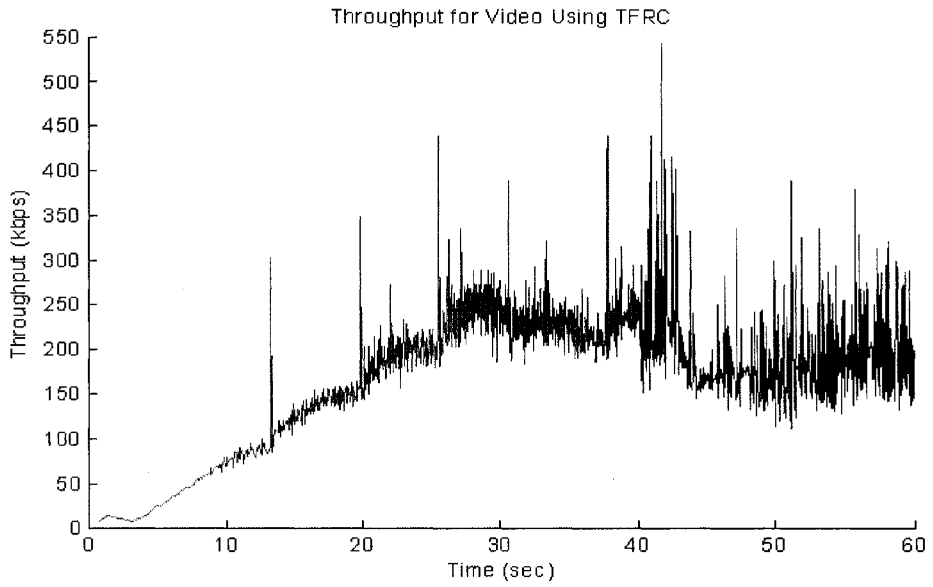

Figure 4.10: Video Session Using the TFRC Congestion Control

Table 4.6: Statistics for Video Session Using TFRC Congestion Control

Mean (kbps)	**Standard Deviation (kbps)**	**Minimum (kbps)**	**Maximum (kbps)**
196.666	58.975	7.492	542.535

Experiment when the bottleneck link is at 1Mbits/s

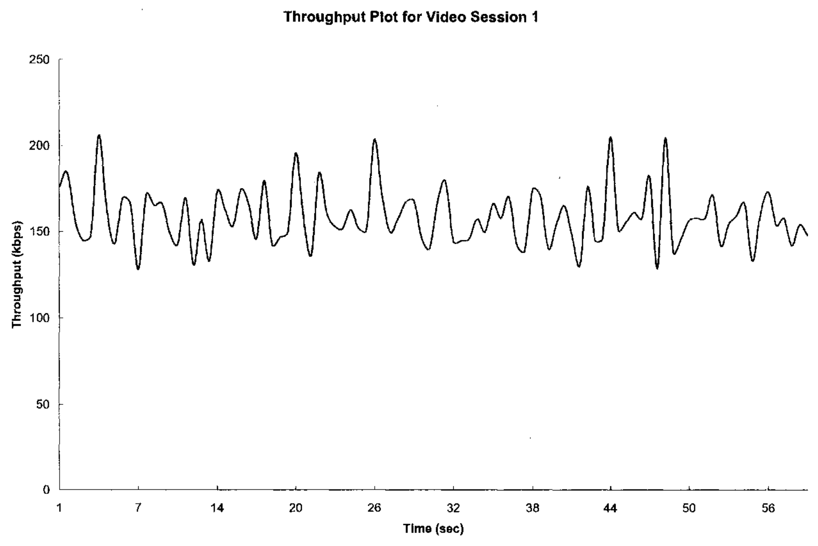

Figure 4.11: Video Session 1 Using the Proposed Congestion Control Scheme

Table 4.7: Statistics of Video Session 1 using proposed technique for 1 Mbps bottleneck link

Mean	Standard Deviation	Minimum	Maximum

(kbps)	(kbps)	(kbps)	(kbps)
158.358	16.785	128.053	205.820

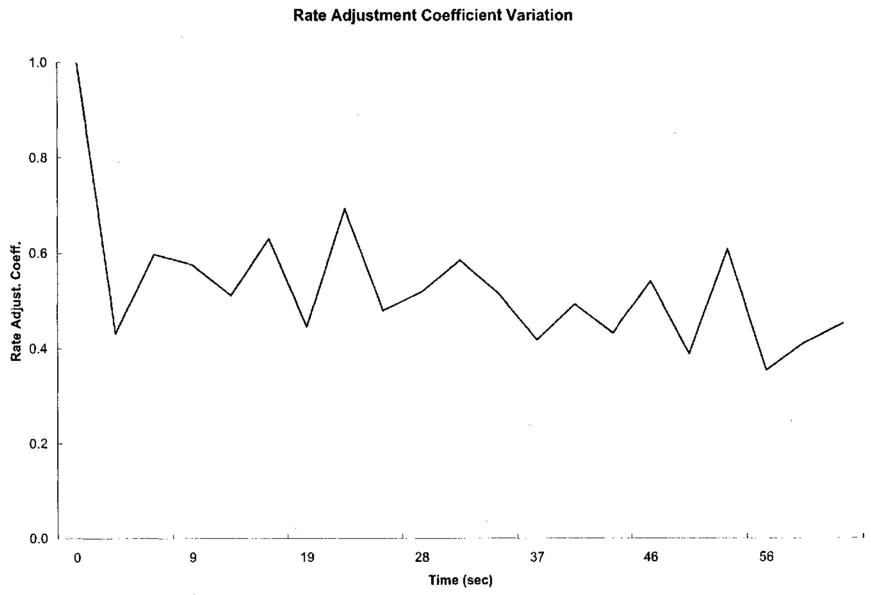

Figure 4.12: Rate Adjustment Coefficient Variation with time for Video Session 1

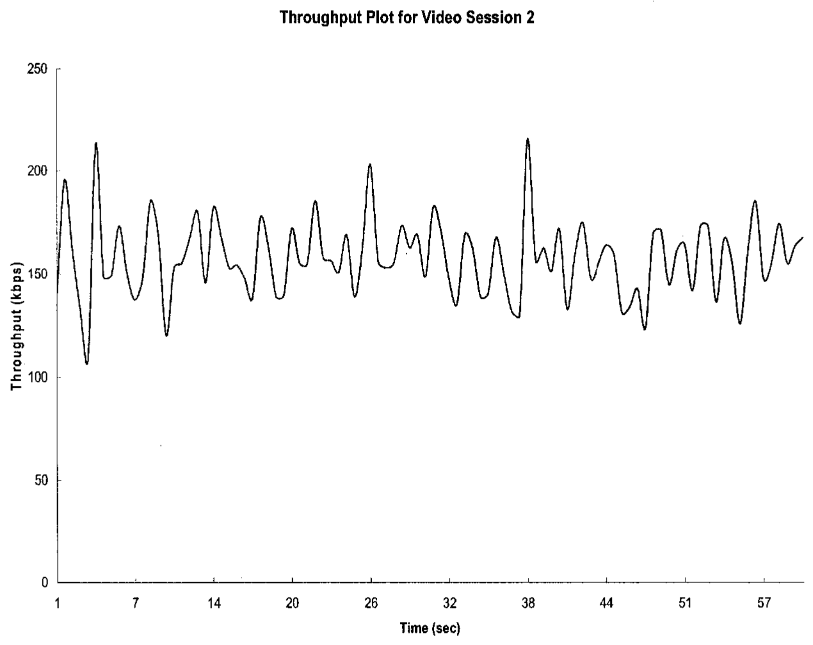

Figure 4.13: Video Session 2 Using the Proposed Congestion Control Scheme

Table 4.8: Statistics of Video Session 2 using proposed technique for 1 Mbps bottleneck link

Mean (kbps)	**Standard Deviation (kbps)**	**Minimum (kbps)**	**Maximum (kbps)**
157.592	18.777	109.078	215.486

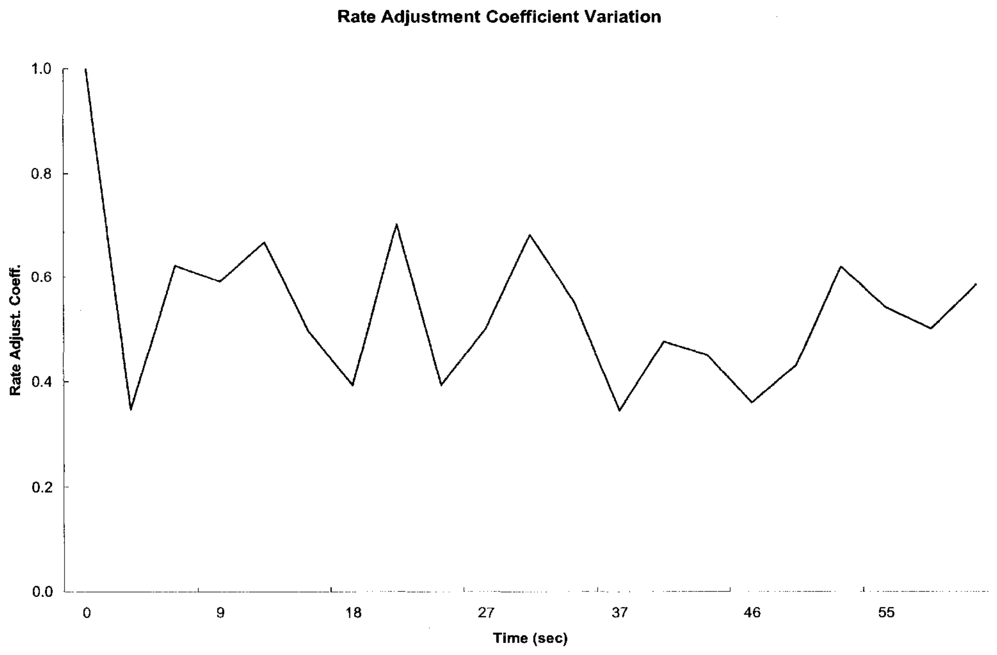

Figure 4.14: Rate Adjustment Coefficient Variation with time for Video Session 2

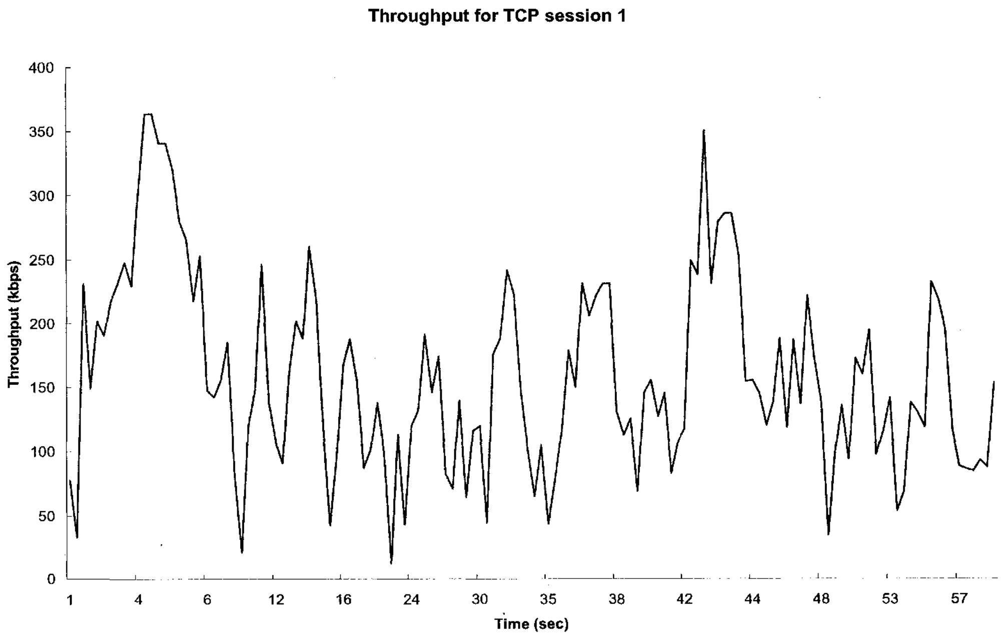

Figure 4.15: Throughput for data session 1 using TCP congestion control

Table 4.9: Statistics of Data Session 1 Using TCP with the 1 Mbps bottleneck link

Mean (kbps)	Standard Deviation (kbps)	Minimum (kbps)	Maximum (kbps)
159.171	75.481	12.000	363.668

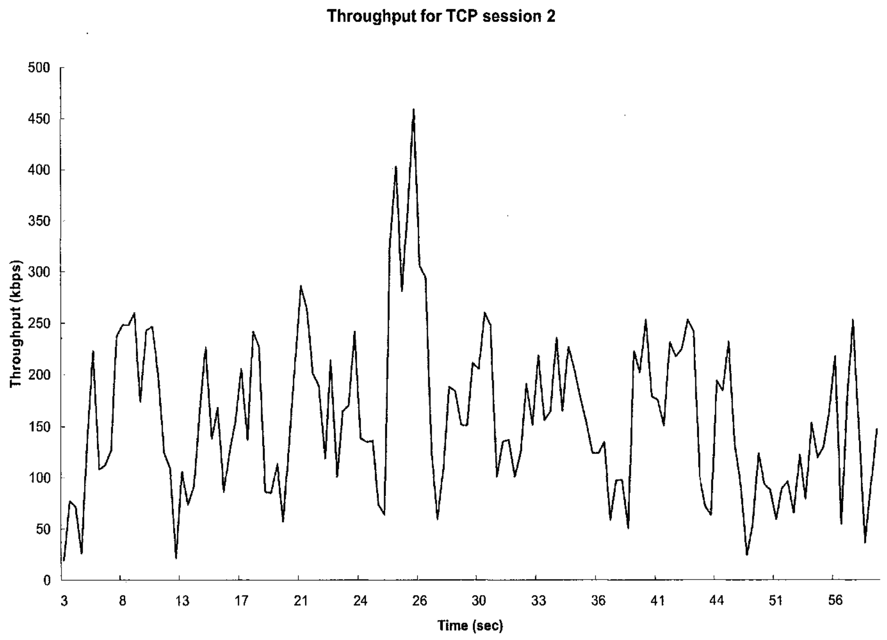

Figure 4.16: Throughput for data session 2 using TCP congestion control

Table 4.10: Statistics of Data Session 2 Using TCP with the 1 Mbps bottleneck link

Mean (kbps)	Standard Deviation (kbps)	Minimum (kbps)	Maximum (kbps)
159.620	78.058	19.532	459.370

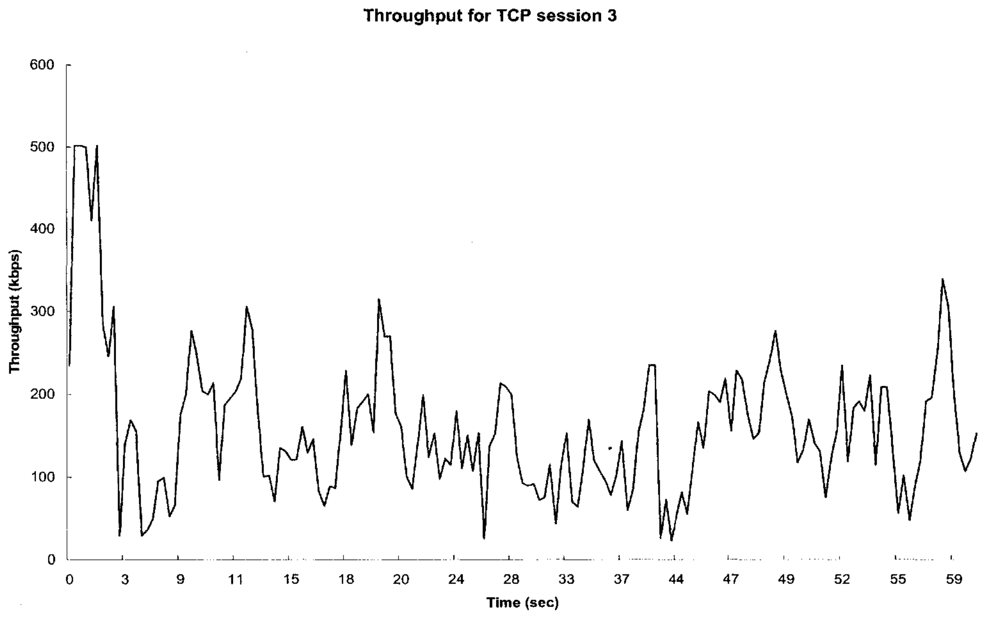

Figure 4.17: Throughput for data session 3 using TCP congestion control

Table 4.11: Statistics of Data Session 3 Using TCP with the 1 Mbps bottleneck link

Mean (kbps)	Standard Deviation (kbps)	Minimum (kbps)	Maximum (kbps)
160.638	88.058	22.734	501.845

4.4 Discussion of Results

Figure 4.3 to fig. 4.5 show results for the three data communication sessions that employed TCP as the transport protocol. Table 4.1 to Table 4.3 show the corresponding statistical data of the throughput for the sessions. The video sessions in the experiment employed the proposed congestion control scheme and the performance results are shown in Fig 4.6 and Fig. 4.8. Corresponding to that are the statistics for the sessions in Table 4.4 and Table 4.5. Fig. 4.7 and Fig. 4.9 show the corresponding rate adjustment variation with time.

It can be observed, as expected that the TCP sessions have an oscillatory throughput profile. On the other hand, the video sessions have a fairly smooth throughput profile thanks to the effective design of the new congestion control scheme. This smoothness can also be captured from the deviation statistics which show the throughput standard deviation values for the video sessions to be much lower than for the data sessions which employ TCP. Indeed a low standard deviation means less fluctuations in the transmission rate which directly implies that the perceived video quality will be fluctuating less.

Whereas the congestion control scheme enables the video communication to have low oscillating throughput profile it acts fairly to the TCP connections. This can be observed from the close average throughput magnitudes for all the communication sessions. The low oscillation experienced by the video sessions is further manifested in the smaller range between the minimum and maximum throughput magnitudes compared to the higher range exhibited by the TCP data connections.
TCP Friendly Rate Control (TFRC) is one of the most prominently promoted congestion control algorithms for video streaming applications. In this research we used it as a testing benchmark. From the results shown in Fig. 4.10 and Table 4.6 the performance of TFRC falls below that of our proposed scheme. It can be observed that the TFRC has a lower average throughput and worst of all a higher rate oscillation. The high rate oscillation captured both in the throughput profile and the standard deviation of the throughput is not desirable in video applications. This re-enforces our proposal of integrating congestion control with the video timing schemes to ensure that in the event of congestion the response does not significantly affect the video application.

The first experiment was carried out with a bottleneck link of 2 Mbits/s. To further evaluate the versatility of the proposed congestion control technique, we carried out another experiment with the bottleneck link reduced to 1 Mbits/s. Figure 4.11 to 4.17 and Table 4.7 to Table 4.11 show the results of this experiment. Again the results show an improvement in the smoothness of the throughput profile. The statistical averages too, show how fairly the available bandwidth is shared between the communicating video and data sessions. Observing the minima and maxima and the standard deviation of the through further emphasises the assertion that the proposed congestion control scheme maintains a smooth transmission rate which implies a more consistent and less fluctuating video quality as perceived at the client.

In both of the experiments we track the variation of the rate adjustment coefficient with time. The graphs shown in Fig. 4.7, Fig. 4.9, Fig. 4.12 and Fig. 4.14 illustrate how closely the proposed technique follows the variation in available bandwidth and makes the appropriate corresponding adjustments in the transmission rate. It is the argument in this thesis that congestion occurrence is closely tracked by observing the client buffer occupancy and adjusting the transmission rate in proportion to that quantity. I contend that the buffer emptiness is a measure of the degree of congestion and other network effects such as excessive delays as observed by the client and that any rate adjustments should be in proportion to this observation. The rate adjustment coefficient is a parameter that captures this information and is used by the corresponding server to adjust the video data transmission rate. The rate adjustment coefficient plots against time show how the buffer occupancy changes with time. Observing the rate adjustment coefficient plots together with the corresponding video throughput plots shows that the video throughput, a measure of the transmission rate, closely tracks the rate adjustment coefficient and as well it should.

4.5 SUMMARY

This chapter has presented a performance evaluation of the proposed congestion control scheme. The performance was evaluated by simulation using ns-2. The experiments used five concurrent connections. Two were video connections and three data connections employing TCP. From the discussion above, we see that the scheme not only maintains a smooth throughput profile for the video session but is sensitive enough to congestion and also fair to TCP as required. A comparison of the proposed scheme with TFRC has shown an improved performance with our congestion control scheme. The variation of the rate adjustment coefficient with time is an indication of how congestion is captured by monitoring the buffer occupancy. Observing the rate adjustment coefficient plots together with throughput plots shows how the proposed scheme enables the streaming session to promptly and appropriately respond to congestion and related network effects.

Chapter 5

Conclusions and Future Research Direction

In this thesis we have proposed a congestion control scheme for use in streaming scalable real time multimedia data. The proposed congestion control scheme is based on monitoring the client smoothing buffer for congestion detection and using the buffer occupancy as measure of how much the connection should respond to the congestion. The implementation is a cooperative approach between the multimedia server and the client. The client captures the relevant information and periodically sends it back to the server which uses the information to determine if congestion is being experienced and if true how much it is penalising the connection. The response, which comes in the form of reducing the load offered to the network by the multimedia session, is in proportion to the congestion penalty.

The design of this scheme is responsible for the improved performance both in response to congestion and for the smooth effect on the video application. Congestion control is integrated with buffer management and ensuring that the application has sufficient data to render at all times. Therefore transmission schedules for the multimedia packets are closely monitored to ensure their timely arrival. Although this thesis is based on MPEG-4 video, the scheme can be integrated in any other scalable multimedia streaming application. MPEG-4 was used for its fine scalability introduced with the FGS coding technique.

Performance studies have shown the congestion control scheme to be effective in managing congestion while also minimizing the attendant effects the condition has on the video applications. This is due to the integration of congestion control with media timing by combining congestion control and receiver buffer management. Receiver buffer monitoring and management is useful because congestion effects and their extent manifest themselves in the emptiness of this buffer. Any response to congestion that takes this into consideration is bound to have a closer prediction of the available network capacity than traditional techniques. Indeed this is the case with our proposed congestion control scheme, as exhibited in its comparative performance against TFRC.

5.1 Future Research Direction

In this thesis we carried out performance tests through simulation. Whereas much care has been taken in ensuring that simulation experiments represent reality, it is still much more controlled environment than real life. It would therefore be great to implement this scheme in a video streaming application to have a real life subjective performance evaluation. The thesis also focussed on SNR FGS coded MPEG-4 video. This scheme can however be extended to other fine scalable coded multimedia and more importantly hybrid SNR-Temporal coded video. MPEG-4 has introduced hybrid SNR-Temporal FGS video coding, which is also applicable to this congestion control scheme with an introduction of a decision algorithm to optimally decide between SNR and Temporal data reduction in the event of congestion.

Bibliography

1. G. Thomas, "Burst switching architecture for a video-on-demand server," *Proc. SPIE Vol. 4519*, pp. 258-265, 2001.

2. K. Lee, T. Kim and V. Bharghavan, "A Comparison of End-to-End Congestion Control Algorithms: The Case of AIMD and AIPD," *Proc. IEEE GLOBECOM'01 San Antonio, TX, US, Vol. 3*, pp. 1580-1584, 2001.

3. A. Eleftheriadis and D. Anastassiou, "Meeting arbitrary QoS constraints using dynamic rate shaping of coded digital video," *Proc. 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 95-106, April 1995.

4. W. Tan and A Zakhor, "Real-time Internet video using error resilient scalable compression and TCP-friendly transport protocol," *IEEE Trans. on Multimedia, Vol. 1, no. 2*, pp. 172-186, June 1999.

5. X. Wang and H. Schulzrinne, "Comparison of Adaptive Internet multimedia applications," *IEICE Trans. on Communications*, *Vol. E82-B, no. 6*, pp. 806-818, June 1999.

6. Q. Zhang, G. Wang, W. Zhu, and Y.-Q. Zhang, "Robust scalable video streaming over Internet with network adaptive congestion control and unequal loss protection," *http://research.microsoft.com/asia/dload_files/group/wireless/pv01-4th.pdf*.

7. J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A Model Based TCP-Friendly Rate Control Protocol," *http://citeseer.nj.nec.com/padhye99model.html*, 1999.

8. B. Zheng and M. Atiquzzaman, "A novel scheme for streaming multimedia to personal wireless handheld devices," *IEEE Trans. Consumer Electronics, Vol. 49, no. 1*, Feb. 2003.

9. R. Puri, K.-W. Lee, K. Ramchandran and V. Bharghavan, "Forward Error Correction (FEC) Codes Based Multiple Description Coding for Internet Video Streaming and Multicast," *Signal Processing: Image Comm., Vol. 16, No. 8,* pp. 745-762, May 2001.

10. Y. Wang, M.T. Orchard, and A.R. Reibman, "Multiple description image coding for noisy channels by pairing transform coefficients," *Proc. IEEE Workshop on Multimedia Signal Processing*, pp. 419-424, June 1997.

11. Y. Wang and Q.-F. Zhu, "Error control and concealment for video communication: a review," *Proceedings of the IEEE, Vol. 86, no. 5*, pp. 974-997, May 1998.

12. Y. Wang, Q.-F. Zhu, and L. Shaw, "Maximally smooth image recovery in transform coding," *IEEE Trans. Communications*, *Vol. 41 no. 10*, pp. 1544-1551, Oct. 1993.

13. H. Sun and W. Kwok, "Concealment of damaged block transform coded images using projection onto convex sets," *IEEE Trans. Image Processing, Vol. 4, no. 4*, pp. 470-477, April 1995.

14. M. Ghambari, "Cell-loss concealment in ATM video codes," *IEEE Trans. Circuits and Syst. for Video Technol., Vol. 3, no. 3*, pp. 238-247, June 1993.

15. J. Liebeherr, M. Nahas, and S. Weisheng, "Application-layer Multicasting with Delaunay triangulation overlays," *IEEE Selected Areas in Comm., Vol. 20, no. 8*, pp. 1472-1488, Oct. 2002.

16. Z. Miao and A. Ortega, "Scalable proxy caching of video under storage constraints," *IEEE Selected Areas in Comm., Vol. 20, no. 7*, pp. 1315-1327, Sept. 2002.

17. A. Mourad, "Doubly-striped Disk Mirroring: Reliable Storage for Video Servers," *Multimedia, Tools and Applications*, *Vol. 2*, pp. 253-272, May 1996.

18. C.L. Liu and J.W. Layland, Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of the ACM, Vol. 20, no. 1*, pp. 46-61, Jan. 1973.

19. J.Y. Chung, J.W.S. Liu, and K.J. Lin, "Scheduling periodic jobs that allows imprecise results," *IEEE Trans. on Computers*, *Vol. 19, no. 9*, pp. 1156-1173, Sept. 1990.

20. J. Gemmel and S. Christodoulakis, "Principles of delay sensitive multimedia data storage and retrieval," *ACM Trans. on Information Systems*, *Vol. 10, no. 1,* pp. 51-90, Jan. 1992.

21. H.M. Vin, P. Goyal, and A. Goyal, "A statistical admission control algorithm for multimedia servers," *Proc. ACM Multimedia*, pp. 33-40, Oct. 1994.

22. J. Gemmel, H.M. Vin, D.D. Kandlur, P.V. Rangan, and L.A. Rowe, "Multimedia storage servers: a tutorial," *IEEE Computer Mag., Vol. 28, no. 5*, pp. 40-49, May 1995.

23. P.J. Shenoy and H.M. Vin, "Efficient striping techniques for multimedia file servers," *Performance Evaluation, Vol. 38, no.3-4*, pp.175-199, Dec. 1999.

24. D.H.C. Du and Y.-J. Lee, "Scalable server and storage architectures for video streaming," *Proc. IEEE Int. Conf. Multimedia Computing and Syst.*, pp. 62-67, June 1999.

25. A. Gaha, "The Evolution to Network Storage Architectures for Multimedia Applications," *Proc. IEEE Conf. Multimedia Computing and Syst.*, pp. 68-73, June 1999.

26. G.A. Gibson and R.V. Meter, "Network Attached Storage Architecture," *ACM Communications, Vol, 43, no. 11*, pp. 37-45, Nov. 2000.

27. S. Berson, L. Golubchik, and R.R. Muntz, "Fault Tolerant Design of Multimedia Servers," *Proc. ACM SIGMOD* pp. 364-375, May 1995.

28. B. Ozden, R. Rastogi, P. Shenoy, and A. Silberschatz, "Fault-tolerant Architectures for Continuous Media Servers," *Proc. ACM SIGMOD*, pp. 79-90, June 1996.

29. R. Tewari, D.M. Dias, W. Kish, and H. Vin, "Design and Performance Trade-offs in Clustered Video Servers," *Proc. IEEE Intl. Conf. Multimedia Computing and Syst.*, pp. 144-150, June 1996.

30. R. Steinmetz and K. Nahrstedt, *Multimedia: Computing, Communications and Applications*, Upper Saddle River, NJ: Prentice Hall, 1995.

31. J.P. Jarmasz and N.D. Georganas, "Designing a Distributed Multimedia Synchronization Scheduler," *Proc. IEEE Intl. Conf. Multimedia Computing and Syst.*, pp. 451-457, June 1997.

32. N.D. Geoganas, "Designing a distributed multimedia synchronization scheduler," *Proc. IEEE Intl. Conf. Multimedia Computing and Syst.*, pp. 3-6, June 1997.

33. TCP, *RFC 793*, *Internet Engineering Task Force*, http://www.ietf.org.

34. TFRC, *RFC 3448*, *Internet Engineering Task Force*, http://www.ietf.org.

35. UDP, *RFC 768*, *Internet Engineering Task Force*, http://www.ietf.org.

36. RTP, *RFC 3550, 3556, 3640*, *Internet Eng. Task Force*, http://www.ietf.org.

37. RTCP, *RFC 3605, 3611*, *Internet Eng. Task Force*, http://www.ietf.org.

38. RTSP, *RFC 2326*, *Internet Engineering Task Force*, http://www.ietf.org.

39. SIP, *RFC 3261*, *Internet Engineering Task Force*, http://www.ietf.org.

40. R. Koenen, "MPEG-4 Overview - (V.21 – Jeju Version)," *ISO/IEC JTC1/SC29/WG11 N4668*, March 2002.

41. M. K. Mandal, *Multimedia Signals and Systems*, Kluwer Academic Publishers, ISBN 1-4020-7270-8, Boston, Nov. 2002.

42. D. Bansal and H. Balakrishnan, "Binomial Congestion Control Algorithms". *Proc. IEEE INFOCOM. Joint Conference of the Computer and Communications Societies, Vol. 2,* pp. 631 –640, Anchorage, AK, US, 2001.

43. S. Floyd, M. Handley, J. Padhye and J. Widmer, "Equation-Based Congestion Control for Unicast Applications," *Proc. ACM SIGCOMM,* , *vol. 30, no. 4*, pp. 43-56, Stockholm, Sweden, March 2000.

44. Y. Dejian; W. Qiufeng; Z. Zuo, "Receiver-buffer-driven approach to adaptive Internet video streaming", *Electronics-Letters.* v 38 n 22 Oct 24 2002, pp. 1405-1406

45. C. Huang and L. Xu, "SRC: Stable Rate Control for Streaming Media," Proc. Globecom 2003, pp. 4016 – 4021.

46. W. Luo and M.E. Zarki, "Receiver initiated resource renegotiation for VBR video transport," *Proc. ICIP 1999. International Conference on Image Processing, IEEE Computer Society*, October 24 - 28, Vol. 3 pp. 105 – 109.

47. P. Cuetos, P. Guillotel, K.W. Ross and D. Thoreau, "Impementation of adaptive streaming of stored MPEG-4 FGS video over TCP," *Proc. IEEE ICME,* Lausanne, Switzerland, August 2002, pp. 405-408.

48. H.M. Smith and M.W. Mutka, "Pattern smoothing for compressed video transmission," *Proc. IEEE International Conference on Communications, ICC'97*, Montreal, Quebec, Canada, June 8 – 12, Vol. 3 pp. 1335-1339.

49. Y. Xia, D. Harrison, S. Kalyanaraman, K. Ramachandran and A. Venkatesan, “An accumulation-based congestion control model,” *Proc. ICC '03. IEEE International Conference on Communications*, Vol. 1, pp. 657 – 663, 11-15 May 2003.

50. A. Shallwani and P. Kabal, “An adaptive playout algorithm with delay spike detection for real-time VoIP” *Proc. Canadian Conference, Electrical and Computer Engineering, 2003. IEEE CCECE 2003.* Vol. 2, pp. 997 – 1000, May 4-7, 2003

51. J. Rosenberg, Q. Lili and H. Schulzrinne, “Integrating packet FEC into adaptive voice playout buffer algorithms on the Internet,” *Proc. INFOCOM 2000.* Vol. 3, pp. 1705 - 1714, 26-30 March 2000

52. F. Chi-Woon and S.C. Liew, “End-to-end frame-rate adaptive streaming of video data,” *Proc. IEEE International Conference on Multimedia Computing and Systems,* Vol. 2, pp. 67 - 71, 7-11 June 1999.

53. H. Kanakia, P.P. Mishra and A.R. Rebman, “An adaptive congestion control scheme for real time packet video transport,” *Trans. On Networking, IEEE/ACM,* Vol. 3, No. 6, pp. 671-682, 1995.

54. I.V. Bajic, O. Tickoo, A. Balan, S. Kalyanaraman, and J.W Woods, "Integrated end-to-end buffer management and congestion control for scalable video communications," *Proc. International Conference on Image Processing, ICIP 2003*, Vol. 3, pp. 257 - 260, Sept. 14-17, 2003

55. NS-2 http://www.isi.edu/nsnam/ns.

56. H. Alnuweiri, K. Asrar Haghighi, A. Kaheel, Y. Pourmohammadi, S. Vuong, "On the Design of a QoS-Aware MPEG-4 Multimedia Server," *http://lan.ece.ubc.ca/QoSMPEG4_IST2001.pdf*, 2001.

57. H.M. Alnuweiri, A.M Mohamed, "MPEG-4 Broadcast: A Client/Server Framework for Multi-Service Streaming Using Push Channels," *Proc. IEEE 4th Workshop, Multimedia Signal Processing*, pp. 543-548, Oct. 2001.

58. H. Alnuweiri, K. Asrar Haghighi, A. Kaheel, Y. Pourmohammadi, "Realizing MPEG-4 Streaming over the Internet: A Client/Server Architecture using DMIF," *Proc. IEEE Intl. Conf. IT: Coding and Computing*, pp. 23-29, April 2001.

59. H. Alnuweiri, K. Asrar Haghighi, A. Kaheel, Y. Pourmohammadi, "Streaming MPEG-4 over IP and Broadcast Networks: DMIF Based Architectures," *http://lan.ece.ubc.ca/DMIF-PV2001.pdf*, 2001.

60. Y. J. Chung, Y. Kim, J. W. Kim, C.-C. Jay Kuo, "Receiver-Based Congestion Control mechanism for Internet Video Transmission," *Proc. IEEE ISCAS., Vol. 4*, pp. 239-242, May 1999.

61. B. Li, H. Shojania, "Experiences with MPEG-4 Multimedia Streaming," ACM Multimedia, *http://citeseer.nj.nec.com/469170.html*, Ottawa, Oct. 2001.

62. T.M. Liu, W. Qi, H.J. Zhang, F.H. Qi, "A Systematic Rate Controller For MPEG-4 FGS Video Streaming," *http://citeseer.nj.nec.com/liu01systematic.html*, 2001.

63. X. Lu, S. Tao, M. El Zarki, R. Guerin, "Quality-based Adaptive over the Internet," *http://einstein.seas.upenn.edu/mnlab/paper/cnds03.pdf*.

64. X. Sun, F. Wu, S. Li, W. Gao, Y. Zhang, "Seamless Switching of Scalable Video Bitstreams for Efficient Streaming," *http://research.microsoft.com/~fengwu/papers/switch_iscas_02.pdf*, 2001.

65. J. Widmer, R. Denda, M. Mauve, "A Survey of TCP-Friendly Congestion Control", *IEEE Network, Vol. 15, no. 3*, pp. 28-37, 2001.

66. C. Lin, J. Zhou, J. Youn and M. Sun, "MPEG Video Streaming with VCR functionality," *IEEE Trans. Circuits Syst. Video Technol.*, *Vol. 11, No. 3*, pp. 414-425, March 2001.

67. M. Vojnovic and J. Le Boudec, "On the Long-Run Behaviour of Equation-Based Rate Control," *http://icwww.epfl.ch/publications/documents/IC_TECH_REPORT_200370.pdf*.

68. H.M. Radha, M.v.d. Schaar and Y. Chen, "The MPEG-4 Fine-Grained Scalable Video Coding Method for Multimedia Streaming Over IP," *IEEE Trans. Multimedia*, *Vol. 3, no. 1*, pp. 53-68, March 2001.

69. G.J. Conklin, G.S. Greenbaum, K.O. Lillevold, A.F. Lippman, and Y.A. Reznik, "Video Coding for Streaming Media Delivery on the Internet," *IEEE Trans. Circiuts and Syst. for Video Techn.*, *Vol. 11, no. 3*, pp. 269-281, March 2001.

70. L.M. Pao and M.T. Sun, "Encoding Stored Video for Streaming Applications," *IEEE Trans. Circuits and Syst. Video Techn., Vol. 11, no. 2*, pp. 199-209, February, 2001.

71. R. Puri, K-W. Lee, K. Ramchandran and V. Bharghavan, "An Integrated Source Transcoding and Congestion Paradigm for Video Streaming in the Internet," *IEEE Trans. Multimedia, Vol. 3, no. 1*, pp. 18-32, March 2001.

72. Z. Li, G. Shen, S. Li and E.J. Delp, "L-TFRC: An end-to-end Congestion Control Mechanism for Video Streaming Over the Internet," http://web.ics.purdue.edu/~li50/ICME03-zli.pdf

73. J. Padhye, S. Floyd, and E. Kohler, "Profile for DCCP Congestion Control ID 3: TFRC Congestion Control," IETF Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-dccp-ccid3-04.txt, Oct. 2003.

74. J. Padhye, S. Floyd, and E. Kohler, "RFC 3448 - TCP Friendly Rate Control (TFRC): Protocol Specification," *IEEE Network Working Group*, http://www.faqs.org/rfcs/rfc3448.html, January, 2003.

75. N. R. Sastry and S.S. Lam, "CYRF: A Framework for Window-based Unicast Congestion Control," *http://citeseer.nj.nec.com/sastry02cyrf.html*, January 2002.