Mixed Streaming of Video over Wireless Networks

A Thesis Submitted to the College of

Graduate Studies and Research

in Partial Fulfillment of the Requirements

For the Degree of Master of Science

in the Department of Computer Science

University of Saskatchewan

Saskatoon, Saskatchewan

by

Justice Daka

Permission To Use

Abstract

In recent years, transmission of video over the Internet has become an important application. As wireless networks are becoming increasingly popular, it is expected that video will be an important application over wireless networks as well. Unlike wired networks, wireless networks have high data loss rates. Streaming video in the presence of high data loss can be a challenge because it results in errors in the video.

Video applications produce large amounts of data that need to be compressed for efficient storage and transmission. Video encoders compress data into dependent frames and independent frames. During transmission, the compressed video may lose some data. Depending on where the packet loss occurs in the video, the error can propagate for a long time. If the error occurs on a reference frame at the beginning of the video, all the frames that depend on the reference frame will not be decoded successfully.

This thesis presents the concept of mixed streaming, which reduces the impact of video propagation errors in error prone networks. Mixed streaming delivers a video file using two levels of reliability; reliable and unreliable. This allows sensitive parts of the video to be delivered reliably while less sensitive areas of the video are transmitted unreliably. Experiments are conducted that study the behavior of mixed streaming over error prone wireless networks. Results show that mixed streaming makes it possible to reduce the impact of errors by making sure that errors on reference frames are corrected. Correcting errors on reference frames limits the time for which errors can propagate, thereby improving the video quality. Results also show that the delay cost associated with the mixed streaming approach is reasonable for fairly high packet loss rates.

Acknowledgements

I would like to express my gratitude to my advisor Dr Dwight Makaroff for his support and help on all aspects from technical to material. His guidance and advice helped me overcome many technical obstacles.

I am grateful to Dr. Mark Eramian, Dr. Derek Eager and Dr. Dave Klymyshyn (External) for their valuable advice on the direction of my research which has helped improve the quality of this work.

Special thanks to all my friends for making my stay in Saskatoon feel like home. I would also like to express my gratitude to Stephanie for her encouragement and support. This acknowledgment would not be complete without mentioning the Graduate Correspondent Jan (mom for graduate students). Thanks for caring and making sure all the administrative issues are taken care of.

This thesis is dedicated to my parents. It is their love, encouragement and understanding that has helped me get this far.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AIMD –Additive Increase Multiplicative Decrease

ACK – Acknowledgment

ARQ – Automatic Repeat Request

FEC – Forward Error Correction

FTP – File Transfer Protocol

GOP – Group of Pictures

GOV – Group of VOPs

GOB – Group of Blocks

HTTP – Hyper Text Transfer Protocol

HTML – HyperText Markup Language

ITU – International Telecommunications Union

IP – Internet Protocol

ISO – International Standards Organization

JVT – Joint Video Team

MPEG – Moving Picture Expert Group

M-JPEG – Motion-JPEG

MSE – Mean Square Error

NACK – Negative Acknowledgment

OD – On-Demand

PSNR – Peak Signal to Noise Ratio

RTP – Real Time Protocol

RTSP – Real-Time Streaming Protocol

TELNET – Protocol for Remote Access

TFRC – TCP Friendly Rate Control

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

VO – Video Object

VS – Video Session

VOP – Video Object Plane

VOIP –Voice Over IP

WWW – World Wide Web

Chapter 1

Introduction

There has been an increase in the use of video over the Internet in the last decade. For example, one study found a growth of 50.2% in the number of video streams viewed in 2005 and predicted a growth of 27% in 2007 [1]. Broadband video clips made up 84.9% of the total video clips streamed [2]. The popularity of streaming media has been facilitated by the increase in the number of broadband connections, giving viewers access to high bitrate videos. Another factor that has contributed to this growth is the widespread use of the World Wide Web (WWW). According to Statistics Canada, the number of households using the Internet in Canada rose from 41.8% in 1999 to 61% in 2005[3]. The web acts as an interface to the Internet allowing easy access to video hosting websites such as Youtube[4]. The increase in the variety of applications producing multimedia content such as audio streaming, video streaming, interactive video games, on-line learning systems, Voice over IP (VOIP), and live broadcasts has also added to the growth in streaming media. In spite of the increase in video data transmitted on the Internet, streaming is still challenging in environments with high levels of packet loss.

Video streaming applications fall into two broad classes: *live streaming* applications and *on-demand* applications. Live video applications (e.g streaming of live sports events, surveillance video and video conferencing) are delivered as the video data is being captured and encoded [60]. In this approach, there are timing requirements for the delivery of data. In contrast to live video streaming, *on-demand*

[1] www.marketresearch.com/product/display.asp?productid=1218347
[2] Ibid.
[3] http://www.statcan.ca/Daily/English/060815/d060815b.htm
[4] http://www.youtube.com

applications deliver *stored* media files to the client at the time when they are requested [58].

On-demand applications can be delivered using three techniques: *download*, *progressive download* and *streaming*. These techniques are described in Section 1.2.

Also, wireless networks have become popular in recent years [3]. A number of places such as airports, hotels, coffee shops, offices and increasingly, private homes are equipped with wireless hotspots and access points. Wireless networks are emerging as an alternative to wired networks, and as a result, most of the applications that are used on wired networks are now common on wireless networks.

Transmission of video over wireless networks comes with a number of challenges. In comparison to wired networks, wireless networks have limited bandwidth [39] because of the fluctuating wireless channel conditions which may cause the bandwidth of the wireless link to drop. Wireless networks are also characterized by high packet losses [15, 48]. This poses a problem for compressed video because data loss results in loss of video quality. This work presents *mixed streaming*, a transport layer mechanism to improve the video quality in high packet loss environments such as wireless networks. Mixed streaming reduces the impact of packet loss on encoded video, thereby improving the video quality.

The remainder of this chapter is arranged as follows. Section 1.1 lists some definitions used in this thesis, while Section 1.2 gives an overview of media streaming and Section 1.3 gives the streaming architecture. Section 1.4 gives the motivation and Section 1.5 outlines the specific goals of the thesis.

1.1 Definitions

This section outlines some of the definitions used in this study in the context of this thesis.

World Wide Web: According to the World Wide Web Consortium (W3C) the web is defined as "an information space in which the items of interest, referred to as resources, are identified by global identifiers called Uniform Resource Identifiers

(URI)"[5]. The WWW is not the same as the Internet. The Internet is specifically defined as an international network of interconnected networks that communicate using Internet protocols [49].

Reference frames: In this thesis, reference frames refer to video frames on which other frames depend. This includes key frames for a group of pictures and other dependable frames which are members of the group of pictures.

Multimedia: Refers to data that incorporates different types of information such as video, text, sound and images.

1.2 On-demand Streaming Overview

1.2.1 Download

The *download* technique transfers the entire video from the server and stores it on the client computer before playback [31]. In this method, the video file is delivered like traditional Internet applications, such as File Transfer Protocol (FTP)[6]. Traditional Internet applications use the TCP/IP protocol suite to deliver data over the Internet. One of the most important protocols in the TCP/IP suite is the transmission control protocol (TCP). TCP is a reliable transport protocol that uses congestion control and flow control algorithms to ensure that data is delivered reliably. TCP works together with the Internet Protocol (IP), a network layer protocol responsible for the segmenting, routing and reassembly of data packets [49].

The download technique typically uses web servers to provide video clips to clients. Web servers use the Hyper Text Transfer Protocol (HTTP), an application level protocol employing TCP/IP protocol suite to transfer web pages to clients [19]. A video download uses the HTTP protocol to transfer the video from the server to the client. Downloading is usually initiated by clicking a hyper link pointing to the video file. Clicking the link in the browser sends a HTTP request to the web server

[5] http://www.w3.org/TR/2004/REC-webarch-20041215/

[6] FTP is an application level communication protocol that allows the reliable transfer of files from one computer to another.

specified in the *href* tag in HTML. The server will start sending the video data embedded in HTTP messages to the requesting browser after receiving the request. The browser writes the video data to the disk until the entire video is downloaded. After finishing the download, the viewer can playback a file at the desired time. One of the advantages of using the HTTP protocol is that it can transfer videos through firewalls, which are often configured to prevent other protocols from transferring data. Another advantage is that the downloaded video can be stored and played for as long as the user wants. The download method offers smooth playback once the download of the file is complete, but it requires clients to wait for the *download time* before commencing playback. Since media files are typically large, downloading results in a large playback delay.

1.2.2 Progressive Download

Unlike downloading, which waits for a file to be completely downloaded before playback, the *progressive download*(also called pseudo-streaming or HTTP streaming) lets the user play the video file while it is being downloaded [57]. The viewer starts playing the video after waiting for a short time required to fill the buffer. One benefit of progressive streaming over the download technique is that the viewer has the option to stop streaming after seeing the beginning of the file if the content is not relevant.

Progressive download uses the HTTP protocol to deliver data, and as a result utilizes an ordinary web server for streaming. The streaming process on the web starts after clicking a hyper link. The link points to a *metafile* which points to the actual video file in turn. The metafile contains information describing the file. First, it indicates the protocol to be used for streaming, the type of player to use, and the location of the video file. Upon executing the link, the browser parses the path contained in the *href* tag in HTML and sends a request for the metafile to the web server. The server responds by sending the specified metafile to the browser. Upon receiving the metafile, the browser checks the file extension of the metafile

and passes its contents to the required player. The player reads the contents of the metafile and sends an HTTP request for the video data. The server then starts sending the data to the player using the HTTP protocol bypassing the browser. The player starts playing the data after the initial buffering period.

Progressive download is able to traverse firewalls because of its use of HTTP, but using HTTP is also a disadvantage because it runs on top of TCP. TCP reduces its sending rate when there is packet loss because of congestion, so that additional packets are not dropped. Cutting the sending rate will delay the video packets possibly resulting in video being stalled during playback and a long buffering time before playback.

1.2.3 Streaming

The *streaming* approach overcomes the limitations of downloading by transmitting and playing the video simultaneously. In this method, clients only wait for a short time required to fill a buffer with the initial portion of the media file before playback. The remainder of the media file is delivered while it is being played [58]. Unlike the previous techniques, streaming requires a specialized server.

Like progressive download, streaming starts after a browser sends a request for the metafile to the server where the metafile is stored. After downloading the metafile, the browser delivers the contents of the metafile to the media player. The player extracts the location of the media file and the protocol to be used for streaming from the metafile. The player then contacts the media server indicated in the metafile and commences streaming using the specified protocol.

Streaming employs specialized streaming protocols to achieve the streaming process. One common protocol is the Real-Time Streaming Protocol (RTSP) [72], which is used to control and establish the streaming. RTSP does not deliver video packets in itself, but instead provides remote control-like functions (i.e Play, Rewind, Fast Forward) to control the streaming. Video packets are usually delivered using the Real Time Protocol (RTP) [71], an application level protocol which supports the

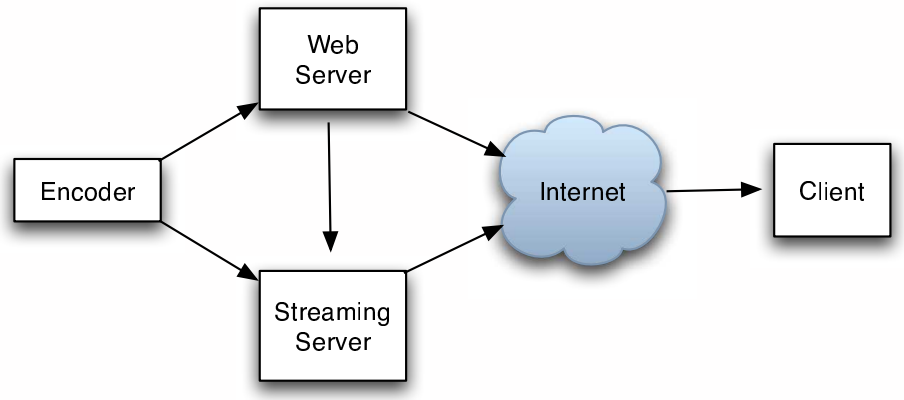


Figure 1.1: Unidirectional media streaming architecture

transmission of media files by providing sequence numbering, time stamping and delivery monitoring. RTP works together with the transport protocol of choice to transmit the video.

1.3 Unidirectional Streaming Architecture

The unidirectional media streaming architecture is made up of a variety of components, which pass information from one to the other during streaming. The most basic architecture is made up of the following components; encoders, servers and clients. Figure 1.1 shows a basic unidirectional streaming architecture.

1.3.1 Encoder

Video information is captured using a video camera. Raw video files have huge sizes and are not suitable for streaming over the Internet. Before a video can be streamed, it must be converted to a streamable format. The encoder converts (encodes) video data into a format that can be used for streaming on the Internet. Encoding reduces the size of the video file by removing details that make little difference in quality. Some of the details removed include common details between frames and common details within a frame. Qualitative and quantitative parameters for the encoder are

set with the target network in mind. The video should ideally be encoded at a bitrate that is lower than the expected available network bandwidth, however this is a challenge because the bandwidth might change with time.

1.3.2 Server

Servers are responsible for storing data and serving requests for media files to clients. Two types of servers are used for streaming; *web servers* and *specialized media servers*.

Web servers are designed to respond to HTTP requests for objects. When a web server is used for streaming, it treats streaming media files like any other web object. Web servers embed video data inside HTTP messages and send them to the requesting browser. As a result, web servers can only be used for downloading and pseudo-streaming. One of the advantages of using a web server for streaming is that it is easy to deploy. The second benefit is that no additional infrastructure is needed, the same server that is used to serve web requests can be used for streaming.

Unlike web servers, which serve web pages, specialized media servers are optimized to serve media content. Media servers cater to requests from clients using specialized streaming protocols such as RTSP. Media servers usually work together with web servers, which provide media metafiles. After streaming begins, the web server is not contacted; the media server communicates directly with the media player.

1.3.3 Client

The media player acts as the client for Internet streaming. The media player is a software program that is responsible for playing streamed or stored media files. Media players are usually launched as stand alone applications or as an embedded application using a plug-in. Plug-ins are software applications which extend the functionality of the browser. After a player is launched, it sends a request to a media server using a streaming protocol such as RTSP. This enable players to support

various streaming functions (such as pause and rewind).

1.4 Motivation

The Internet was initially dominated by traditional applications such as FTP and telnet. After the introduction of the web in the early 90s, there has been an increase in non-traditional applications such as multimedia. The web provides links to video content, thereby making access easy. The use of multimedia over the Internet is expected to increase as the number of broadband Internet connections increase. The widespread use of multimedia applications, such as video streaming, is not limited to wired networks, but it has also found its way to non-traditional networks such as wireless networks. One of the factors that have facilitated the growth in video streaming use over wireless networks is the development of efficient compression methods such as MPEG-4, which makes it possible to stream video over networks using minimal bandwidth [90].

The Internet is a best effort network and therefore does not in itself provide any guarantees for the delivery of data in a timely fashion [49]. This can be a challenge to video streaming applications that need stable bandwidth. Traditional applications, such as FTP, can tolerate fluctuating bandwidth as they do not have stringent timing requirements. Unlike the former, streaming media applications require guaranteed bandwidth and have timing requirements. The problem becomes even bigger if the network to be used is a wireless network where the amount of packet loss is high.

1.4.1 Data Loss in Wireless Networks

In networks, data loss is caused by transmission errors and congestion losses. In wireless networks, transmission errors are caused by interference, multi-path effects and the distance between the transmitter and the receiver [5, 11, 16, 55]. A longer distance between the transmitter and receiver weakens the signal strength, resulting in high bit error rates and packet loss.

Congestion losses occur when data arrives at a router or base station faster than its output link rate [41]. During congestion, packets that are not lost will be delayed because of too much queuing in the network. After the buffer space is used up, the network will drop incoming packets. A congestion control scheme helps the network to recover from the congestion state.

Video data has a lot of redundancy amongst subsequent frames. To remove the inter-frame redundancies, video information is temporally encoded. Temporally compressed video is made up of independent frames (called reference frames) and dependent frames. Independent frames are encoded individually, while dependent frames use information in other frames to perform decoding successfully [27].

When a video is sent over the wireless network, the compressed video may be affected by errors. This is because some data is lost during transmission. If the lost data belongs to a reference frame, it will cause an error on the reference frame when it is decoded and will propagate to all the frames that depend on that reference frame. Errors that occur on a reference frame will persist for a long time, until a new reference frame is reached. Errors that occur on a frame which is close to, and precedes a new reference frame, will only persist for a short time. In a case where the initial reference frame is lost, the perceived quality of the frames after the lost reference frame is bad, even if all the other dependent frames are received correctly [70]. Figure 1.2 illustrates the extent of the damage that can occur on a dependent frame if a reference is lost or corrupted. In Figure 1.2, losing reference frame 213 causes the quality of frame 215 to degrade from 39 dB to 23 dB. The quality drops because the motion vectors in the successive frames after the lost reference frame cannot be decoded correctly.

1.4.2 Solutions

The propagation error problem can be solved with two extreme approaches: the proactive approach and the reactive approach. For a proactive approach, the system takes action before the error occurs, while in a reactive approach, action is only taken

(a) Frame 215(39dB) without data loss (b) Frame 215(23dB) after losing frame 213

Figure 1.2: Impact of data loss on dependent frames

when the error occurs [88].

One proactive solution to avoid the propagation of errors would be to compress each frame independently [7, 60]. Compressing the frames independently can reduce propagation errors, but it also increases the size of the video. This decreases the effect of compression, which is to reduce the size of the video. Another solution may be to use reference frames more frequently. If reference frames are used very frequently, again the full benefits of compression will not be achieved because reference frames have a bigger size compared to the dependent frames [69]. Forward Error Control (FEC) [7], which transmits the video together with redundant data used for correcting errors, is another proactive solution. Transmitting redundant data results in additional bandwidth being consumed irrespective of packet loss.

Reactive approaches on the other hand, employ error recovery at the receiver after data loss. They conceal the effect of the lost data so that the received image is more pleasing to the eye. Reactive approaches can take the form of retransmissions or post-processing error concealment schemes. Error concealment attempts to conceal the error by using information from the same frame or by using information from the neighboring frames [79, 84].

1.5 Goal of the Thesis

Motivated by the above discussion, the transmission of video over error-prone wireless networks is studied in this thesis. The aim is to develop the concept of mixed reliability (mixed streaming) in video transmission over wireless networks and conduct a performance study using simulation. In particular, the performance study has the following goals:

- to show that mixed reliability can be used to reduce the impact of wireless transmission errors on video.
- to study the quantitative performance aspects of mixed reliability in transport of video data. The video will be transported over a simulated network. The quality of the resulting video with losses will then be analyzed using the Peak Signal to Noise Ratio (PSNR).

The remainder of this thesis is organized as follows. Chapter 2 provides background of the study. Chapter 3 presents the related work. Chapter 4 outlines the experimental environment and protocol specification. Chapter 5 gives the results on video quality. Chapter 6 presents the results on delay. Finally, Chapter 7 gives the conclusions and direction for future work.

Chapter 2

Background

2.1 Reliability for Multimedia Transport

Communication networks usually follow a layered model (referred to as a stack) that provides clearly defined functions that enable internetwork connectivity. Each layer has a defined task that provides services to the next upper layer and processes data from the next lower layer. The Internet stack has 4 layers and is based on the TCP/IP protocol stack. The layers of the TCP/IP stack are the following: link layer, network layer, transport layer, application layer [76]. The link layer delivers segment frames through a physical link using the network interface card. The network layer (IP) is responsible for routing the datagrams between the source and the destination. The IP network layer is unreliable and does not guarantee the correct delivery of data, because data may be lost due to issues such as router buffer overflows and channel error. The network layer leaves the responsibility of providing reliability to the upper layers. The transport layer provides transparent end-to-end communication between two hosts. At the sender, it receives data from the application layer, divides it into packets and passes the data to the network layer. This process is reversed at the receiver, where data is obtained from the network layer and delivered to the application. The transport layer provides two protocols: TCP and UDP. TCP is a reliable connection-oriented protocol while UDP is an unreliable connectionless transport protocol [49]. The application layer provides services for application based protocols. Some of the common application layer protocols are FTP, HTTP and TELNET. Applications require different levels of reliability. Most traditional Internet applications such as FTP require total reliability because losing

any part of the data leads to file inconsistency. These applications are delivered using TCP's features, such as ordered delivery and retransmission of lost data. As a result, application developers do not need to implement these details in their applications. Applications such as streaming media do not require high reliability because they can tolerate some losses. Such applications are often delivered using best effort protocols such as UDP, which does not implement TCP's aggressive congestion and error control algorithms. Four levels of reliability can be used by video applications: full reliability, mixed reliability, conditional reliability and best effort.

2.1.1 Full Reliability

In full reliability transmission schemes, all the data is delivered as it was sent by the sender. Any part of the data that is lost during transmission will be retransmitted. The reliable connection-oriented service is typically provided by TCP. Full reliability has been considered as not being appropriate for streaming media [42, 52, 79] because it requires at least one additional round trip time to recover from a lost packet. If the transmission delay from the sender to the receiver is large, retransmission will not work at all. This problem can be solved by buffering the video before playout. Although retransmission adds extra latency when recovering from a loss, it still can be an attractive option [65] because it only requires the sender to resend data which is equivalent to the lost information. In spite of the cost associated with reliable data delivery, it has been shown that the price is worthwhile for delivering streaming media on the wired Internet [46].

2.1.2 Best-Effort Transmission

In best effort transmission, the data that is lost during initial transmission is not retransmitted. In the related protocols, the sending rate is determined solely by the sending application. Packets are delivered to the application as soon as they arrive, even if they are out of order. The receiving application is responsible for rearranging the data and dealing with packet loss [67]. The most common best effort transport

protocol is UDP. UDP introduces virtually no increase in delay or reduction in throughput and does not provide any reliability in its purest form. The protocol is connectionless, and delivery and duplicate protection are not guaranteed. UDP uses a checksum to validate headers and data payload. If an incoming packet has an error, it will fail the checksum. Packets that fail the checksum will be dropped, no matter how small the error is [67]. Some researchers have added modifications to basic UDP to improve its reliability and rate control [44, 50, 66, 82]. UDP is considered selfish and ill-behaving to other flows using the same link because it does not reduce its sending rate due to congestion and packet loss [30, 80], while other protocols like TCP reduce their sending rates.

In order to maintain fairness between TCP and non-TCP flows, a TCP-friendly protocol may be used [64]. TCP-friendly protocols change their transmission rate according to the round trip time and packet loss probability to get the throughput that a TCP connection passing through the same link would get [22].

TCP-friendly behavior can be achieved in two ways. One way [80] is to directly mimic the TCP congestion control mechanism by implementing AIMD (Additive Increase Multiplicative Decrease) [38]. Achieving TCP friendliness in this way may come with some of the disadvantages of TCP. For example, TCP halves the sending rate when there is a packet loss due to congestion. Reacting to congestion like this may be too severe for multimedia applications.

The second way TCP friendliness can be achieved is by utilizing the TCP throughput equation [22]. In this case, the sender gets the desired throughput using the throughput equation, and responds appropriately according to the specific algorithm. Floyd *et al.* [22] utilize the second approach. Instead of halving the sending rate for every packet loss, gradual changes are made in the sending rate. The TCP throughput equation is shown in equation 2.1,

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

where s is the packet size, RTT is the round trip time, t_{RTO} is the retransmission time

out and p is the loss event rate. The loss event rate p is calculated at the receiver by using a weighted average method to average the packet loss over several loss intervals. Every time the receiver receives a packet, it sends feedback to the sender stating the sequence numbers which can be used by the sender to measure the RTT. The sender then smoothes the round trip time using the exponentially weighted moving average (EWMA). This determines the responsiveness of the transmission to changes in round trip time. The value of t_{RTO} is estimated from the RTT [22].

2.1.3 Conditional Reliability

Conditionally reliable protocols [17] will retransmit a lost packet if a certain condition (such as packet priority or bandwidth availability) is true. The receiver has to make a decision whether to request for a retransmission from the server when data is lost. For example, when a packet is lost in a system that uses a conditional reliability scheme based on packet priority, a packet will be retransmitted only if the lost packet has a high priority and there is sufficient bandwidth. Feamster *et al.* [17] use a conditional reliability scheme which retransmits data based on the priority of the lost packet.

2.1.4 Mixed Reliability

Early Internet applications such as FTP delivered data in a reliable manner. The introduction of multimedia applications increased the popularity of best effort delivery because multimedia applications such as video can tolerate some types of data loss. However, not all types of data in a multimedia stream can withstand data loss without a significant negative effect. This may result in a video with low quality being delivered. In this thesis, a delivery approach with mixed reliability is explored. This is based on the fact that data in a video file is composed of frames with different levels of sensitivity to errors.

A mixed reliability (mixed streaming) scheme uses a combination of reliable and best-effort transmission simultaneously to deliver a file using multiple streams. Im-

portant data is sent reliably while the less important data is transmitted unreliably. Mixed reliability is similar to conditional reliability in principle, but differs in the implementation and in the manner in which data is classified into important and less important categories.

Since the mixed approach provides reliability for data based on the sensitivity of frames to errors, it is expected that it will result in a video with losses occurring only in parts where data loss has little effect. This will result in better quality video being delivered.

2.2 Video Encoding

Video images are captured as RGB (red, green, blue) signals. Since these signals are highly correlated, they are represented in a color space called *YUV*, which codes the brightness or luminance(Y) at full bandwidth and the color signals or chrominance (UV) at half the bandwidth [26]. Transmitting YUV video data over a network requires a lot of bandwidth because YUV video files are large. For instance, a 5 minute YUV video with frame size 352 x 288 pixels, played at 30 fps could have a file size of 1.27GB. Compressing the video reduces its size. For example, when the above video is compressed at 200 kbps using MPEG-4, its size reduces to 7.5 Mb. Video compression methods, such as MPEG exploit temporal and/or spatial techniques to achieve high compression levels [17].

Temporal-based compression is achieved by using one or more reference frames and representing the majority of frames as the difference between each individual frame and one or more reference frames [26]. It looks at the video information on a frame by frame basis for changes between the frames. Some amount of redundant information between frames is removed depending on the encoder. For example, in a video clip showing a person reading, there would be a lot of redundancy between subsequent frames. The only difference between the frames would be the movements around the mouth of the person. For a clip showing a movie with a lot of high speed motion, the redundancy would be small. The redundancy is small between scene

changes as well. When there is a scene change, the video is indexed with a reference frame. The size of the file is larger when a lot of reference frames are used.

Spatial compression deletes information that is redundant within a frame. Areas of redundant information are defined by using coordinates instead of specifying each pixel in the area [26]. Spatial compression is performed using data compressors, such as transform coding. In transform coding, redundancies are removed from a frame by mapping the pixels in a transform domain before being compressed. The image energy of most natural scenes is concentrated mainly in the low frequency region, and hence can be represented using few transform coefficients. These coefficients are quantized to remove the unimportant coefficients, without significantly degrading the quality of the reconstructed frame [27].

2.3 Digital Video Standards

There are various video standards that can be used for digital video. Publication of digital video standards has primarily been done by two organizations, the International Organization for Standardization (ISO) and the International Telecommunications Union (ITU). Amongst the well known video standards are the following: the MPEG family of standards from ISO, H.261 and H.263 from ITU and JVT from ITU/ISO. These standards are discussed briefly below, while a detailed discussion can be found in the literature [26]. Other proprietary video systems such as Real Player, Windows Media Player and Motion-JPEG are also in wide use.

The MPEG standards [25, 29, 35, 85] were developed by the Moving Picture Experts Group (MPEG). These standards facilitated the development of interactive video on CD-ROM and Digital Television. Amongst the most notable of these standards are MPEG-1, MPEG-2 and MPEG-4.

MPEG-1 was originally designed for digital storage applications with a target bitrate of 1 to 1.5 Mbps, but it has been used by a wide range of applications [24]. MPEG-2 was designed to improve on the MPEG-1 standard. MPEG-2 picture resolutions are suitable for DVD and HDTV while MPEG-1 picture resolutions are

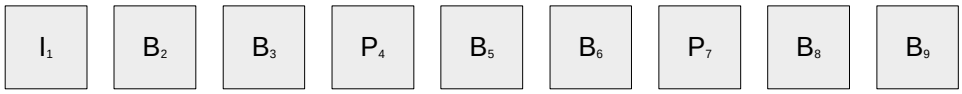
designed for CD-ROM or Video CD. MPEG-2 has typical bit rates in the range of 2 to 15 Mbps.

There are two notable differences between MPEG-1 and MPEG-2. The first one is that MPEG-2 supports interlacing and High Definition (HD) while MPEG-1 does not [85]. The second difference between MPEG-1 and MPEG-2 is the different scalability levels. MPEG-2 provides scalable coding while MPEG-1 does not. Using scalable coding, receivers and networks of varying capabilities can be accommodated. Scalable coding allows a receiver to decode a subset of the full bit-stream in order to display an image sequence at a reduced quality.

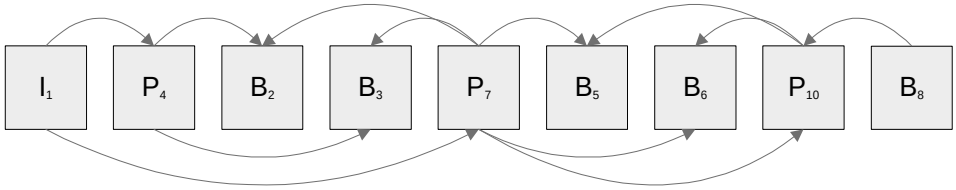
MPEG-4 is a relatively new MPEG standard which has typical bit rates of 64 kbps to 2 Mbps. One of the major differences between MPEG-4 and the preceding standards is that it is object-based. In MPEG-4, a scene is encoded into one or more objects. Each object can be manipulated and accessed independently. MPEG-4 is organized in a hierarchy of objects: Video Session (VS), Video Object (VO), Video Object Layer (VOL) and Video Object Plane (VOP) [90]. A VS includes one or more VOs, which are part of a scene. A VO is made up of several VOLs, which are ordered sequences of frames or snapshots. The snapshots are called VOPs.

MPEG encoders exploit both the spatial redundancies and temporal redundancies. Spatial redundancies are exploited by using block-based Discrete Cosine Transform (DCT) coding [24]. Encoders exploit temporal redundancies by using motion compensated prediction. Using motion-compensated prediction reduces the inter-frame prediction error.

Generally, MPEG encodes video information into three types of frames: Intra frames (I-frames), Predictive frames (P-frames) and Bidirectionally predictive frames (B-frames). I-frames (I-VOP in MPEG-4) are coded separately without depending on any other frames, as a result they are called *independent* frames. All the frames starting from a particular I-frame to the next I-frame are grouped in a pattern called a Group of Pictures (GOP) or Group of VOPs (GOV). Inside the GOP, there are a number of P-frames (P-VOP) frames which are coded in reference to previous I or P frames. The remainder of the frames in the GOP are B-frames,



(a) MPEG video frames in display order



(b) MPEG video frames in transmission order

Figure 2.1: MPEG frame dependencies

which are coded based on immediate previous I or P-frames as well as immediate next I or P-frames. P-frames and B-frames are called *dependent* frames because they need information from other frames to be decoded successfully. Figure 2.1 shows the dependencies between different types of frames. Since B-frames are bidirectionally predicted, transmitting the video sequence in display order results in B-frames being delivered to the decoder before the future frames they depend on. This might cause an error because the decoder can not display the frame correctly without data from the reference frame. To avoid this, the frames are rearranged accordingly. For instance, a sequence with a GOP display order of $I_1B_2B_3P_4B_5B_6P_7B_8B_9$ shown above will rearranged as $I_1P_4B_2B_3P_7B_5B_6P_{10}B_8$ in transmission order. In this manner, frames will always arrive after the frames on which they depend [85].

The H.261 standard was standardized by the ITU in 1990 and it was the first video standard to be accepted by the ITU. It was designed for bit rate ranges of 64 Kbps to 384 Kbps. These data rates make H.261 suitable for use over ISDN lines. H.261 is typically used for video conferencing and video telephony. The compression in H.261 is achieved by using intracoded frames and intercoded frames, just like in the MPEG standard [24].

H.263 improves on the initial H.261 standard. It included some improvements from MPEG-1 and MPEG-2 such as half-pixel motion compensation, median prediction of motion vectors, improved entropy coding, unrestricted motion vectors, and more efficient coding of macroblock and block signaling overhead. H.263 has

data rates of 64 Kbps to 1 Mbps and is used for videoconferencing over dedicated telecommunications lines, as well as over IP based networks [26].

The Joint Video Team (JVT) is a collaboration between the ISO and the ITU. It is called MPEG-4 part 10 and H.264 by the ISO and ITU respectively. JVT accommodates bit rates of 32 Kbps and upwards. It incorporates some of the techniques used in H.263 and its extensions such as H.263+ and H.263++. JVT provides the same visual quality as MPEG-4 advanced simple profile at half the bit rate. JVT is intended to be used for video streaming systems [24].

Motion-JPEG (M-JPEG) is another compression scheme that is in common use. It is based on the JPEG still image compression standard. In M-JPEG, a video clip is encoded as a sequence of successive JPEG images [26]. M-JPEG can be used in video applications, where a lot of editing is required.

In this thesis, the video compression studied is limited to the MPEG-4 video standard because MPEG-4 and MPEG-like standards are in wide use. The discussion can still be applied to other modern standards, since some of the compression principles used are similar. Another reason is that MPEG-4 is based on an open standard and there are a number of MPEG-4 encoding and analysis tools that can be accessed freely.

2.4 Error Control Methods

There are four techniques that are used to deal with errors in video transmission. These techniques are retransmission (ARQ), forward error correction, error concealment and error resilient coding [65, 69, 87].

In ARQ-based error recovery, the data that is lost will be retransmitted. ARQ is usually rejected as an error control method in live media streaming because of the delay it introduces while retransmitting the lost packets. The retransmitted packet takes approximately one and a half round trip times, which might exceed the delay tolerated by an application. For instance, the round trip time can be on the order of 100 ms or more [6, 34]. For interactive multimedia applications, the acceptable

delay is between 100 and 200 ms [34]. Therefore, any retransmission of the lost data must be made within 100 ms after the data is lost.

In networks where the delay is small, such as LANs [14], ARQ can work very well because the round trip time is small. Retransmission-based schemes can also be used in delay-constrained networks, if the application in question can accommodate some buffering at the receiver [88]. Buffering the packets at the receiver makes it possible for packets to be retransmitted without missing their playout time. In some retransmission schemes such as the one proposed by Rhee [69], retransmitted packets that have missed their deadline can still be useful. This is because some future video frames may depend on them.

An ARQ error control scheme is made up of three components: loss detection, receiver feedback and a retransmission strategy [49]. A loss detection mechanism is needed in order to detect errors. Errors may be detected if a duplicate acknowledgment or negative acknowledgment is received by the sender due to a missing sequence number. In order to provide feedback, the receiver can send an acknowledgment when data is delivered. If some data is missing, either duplicate acknowledgments or negative acknowledgments are sent. Data that is received in error will be retransmitted by the sender after receiving a negative acknowledgment or duplicate acknowledgments.

Forward error correction schemes add redundant data to the original data and transmit it to the receiver. When an error occurs during transmission, the redundant data is used to construct the lost data. FEC divides the video stream into segments. Each segment is then packetized into k packets. For each segment, a block code is added to the segment to generate an n packet block, where $n > k$. In order to recover the k packets after a loss, the receiver needs to receive any k packets in the n packet block that was sent [88].

Error resilience encoding is a proactive error correction technique [88]. It is performed at the source before packet loss occurs. It enhances the robustness of compressed video in the face of packet loss. Error resilient methods include resynchronization marking, multiple description coding and data recovery. Resynchro-

nization limits the impact of incorrect data within a local area because the decoder has the option of advancing to the next resynchronization point or to continue decoding when it encounters an error. Data recovery methods such as Reversible Variable Length Codes (RVLC) recover as much data as possible when an error occurs. In RVLC, code words are generated such that they can be decoded in both directions. When an error occurs, the decoder moves to the next resynchronization point and decodes backwards. Another error resilient method is Multiple Descriptions Coding (MDC). MDC produces multiple streams or descriptions. The more descriptions that are received, the better the quality.

The final method to be discussed is error concealment. Error concealment is a reactive error control technique performed at the receiver [84, 87]. Error concealment is used to cover up for the lost data, so that it is less visible to the viewer. Error concealment is performed in two ways: by using spatial interpolation and temporal interpolation. Spatial interpolation is used to reconstruct data within a reference frame from the neighboring pixels. Temporal interpolation is used to reconstruct lost data in inter-coded frames. Error concealment methods take advantage of the fact that most of the information remains the same over a number of frames. Therefore, replacing erasures with information from previously received frames can greatly reduce the impact of errors. One of the advantages of error concealment is that it does not need additional data to be sent for errors to be corrected. Errors can be corrected by using the information which is already available at the decoder. In this work, default error concealment algorithms implemented by the MPEG-4 decoder are utilized.

2.5 Wireless Networks

Wireless networks [48, 62, 89] have become widely deployed in recent years. Wireless networks allow portability and are relatively easy to install, this makes them a viable alternative to wired networks. As a result, users can have access to the network as long as they are within the transmission range of an access point or base

station. Using the radio waves as a transmission medium makes wireless networks susceptible to interference, because the surrounding environment interacts with the signal, blocking signal paths and introducing noise and echoes.

When the signal is blocked, the wireless client will be disconnected from the network. The noise and echoes in the signal will introduce errors in the data. When an error that causes packet loss is introduced, the packet that is received in error may be retransmitted and in the case where the packet is not received, the retransmission timer or duplicate acknowledgments will trigger retransmissions.

The bandwidth in wireless networks is lower than in their wired counterparts [39]. For instance, a wired LAN can support bitrates of 1000 Mbps or more, while wireless LANs currently only support bitrates up to 254 Mbps. In wireless networks, the bandwidth is divided among users sharing the base station or access point. Therefore, the capacity of the network is more dependent on the number of users sharing a base station. If the base station is overloaded, the quality of the connection will degrade. Additional base stations can be installed in places where the demand for connections is high to deal with this problem. In this thesis, video data is streamed over a wired network combined with a wireless network. The wireless part of the network simulates a 802.11 network. As a result, the characteristics associated with wireless networks mentioned in this section are expected.

2.6 Summary

In this chapter, some issues related to streaming such as media transport reliability and digital video encoding have been discussed. Error control methods used to protect data during retransmission are also outlined. Finally, a brief discussion of the nature of wireless networks and associated challenges was presented.

Chapter 3

Related Work

This chapter discuses related work on multimedia delivery. The relevant work presented is in four categories. To provide an account of streaming media characteristics, measurement and characterization studies relevant to media streaming are discussed first in Section 3.1, then wireless video streaming is presented in Section 3.2, error control methods are discussed in Section 3.3 and lastly, work on video quality is presented in Section 3.4.

3.1 Measurement and Characterization of Media

Measurement and characterization studies concerning network delivery of multimedia provide relevant information, which is useful when evaluating the performance of the delivered media. Some information that can be obtained from characterization studies include the data loss levels, usage characteristics and transmission channel behavior. These parameters can then be incorporated into the simulation to get a more realistic simulation environment. There have been a number of papers [3, 9, 54] that have been published concerning measurement and characterization of streaming media traffic.

Loguinov and Radha [54] analyzed video traffic between an ISP and dial-up clients across the USA. To conduct the experiments, clients dialed-in to stream MPEG-4 video clips from the server. The experiments were conducted in 1999 with CBR MPEG-4 video clips encoded at 14 kb/s and 25 kb/s respectively. The findings of the study showed that the average packet loss rate was around 0.5%, which seems small. Their study also showed that some packet loss is common in most streaming

sessions. Specifically, they reported that only 38% of the streaming sessions do not experience any packet loss. For the sessions with packet loss, most sessions (91%) had a loss rate smaller than 2%. They also found that 2% of all the sessions that experienced data loss had a data loss rate higher that 6%. This level of loss might lead to impairments in the video when it is decoded. The reason for the observed loss levels could be that experiments were conducted using dial-up connections, which are slow and unstable. In the last few years, there has been an increase in broadband connections. For instance, one recent survey in 2005 found that only 18% of home users did not have a high speed Internet connection in Canada[1]. As broadband connections become popular, fewer sessions may experience the data loss observed because wired broadband connections are more stable. Loguinov and Radha study's also established that retransmission can be used as an effective error control method in spite of the latency it introduces when retransmitting lost packets. They noticed that 94% of lost packets that were retransmitted arrived before their playback deadline.

Boyce and Gaglianello [9] examined the error characteristics of MPEG-1 video clips transmitted over the public Internet. The study was conducted in 1998 with video clips encoded at 384 kbps and 1 mbps respectively. The Real-time Transport Protocol (RTP) transmitted on top of UDP was used to deliver the video data. Boyce and Gaglianello observed that even low packet loss rates could have a huge impact on the video. For example, a packet loss rate of 3% could lead to loss of 30% of the frames after decoding the video. The reason for difference in the loss level at the packet level and at the video frame level is due to the way MPEG encoded videos. Losing a reference frame results in dependent frames not being decoded correctly. Given that MPEG-like codecs are in common use, it is expected that this occurrence will be common.

Li *et al.* [51] conducted a study to characterize media clips stored on the web in 2003, which was a follow up study to a similar one in 1997 by Acharya and Smith [1]. A media crawler was used to search the web for media clips. The study

[1]http://www.statcan.ca/Daily/English/060815/d060815b.htm

showed that Real Media accounted for 63% of all clips, while Windows Media and Quicktime accounted for 32% and 5%, respectively. About half of all clips analyzed were video, which accounted for 57% while audio accounted for 43% of the clips. They also reported that 10% of clips have a duration less than 30 seconds and that 10% of clips have a duration greater than 30 minutes. The clip durations observed are significantly higher than those reported by Acharya and Smith, who found that more than 75% of clips were less than 30 seconds. The discrepancy in the video lengths could be due to the fact that the studies were conducted in different years.

In the studies described above, it can be learned that small data loss can result in huge errors when the video is decoded. Another useful observation is that the majority of video files on the Internet are between 30 seconds and 30 minutes. This gives a rough idea of the range of video lengths that can be used in the experiments.

Sripanidkulchai *et al.* [75] conducted an analysis of live streaming video and audio workloads. The study was based on logs collected from Akamai Technologies in 2004. Their results revealed that only 7% of the traffic observed is video while audio accounts for 71% of the streams. The percentage of the video traffic seems small for the video streams compared with those observed by Li *et al.* [51]. The reason could be that audio clips are more popular than video clips for live content. Another reason could be due to the method that was used for identifying audio and video. Streams with bitrates less than 80 kbps are considered as audio and content with bitrates over 80 kbps is considered as video. Their findings also show that nearly half (40-50%) of live multimedia streams use TCP for streaming. The number of streams using TCP could have been influenced by the media server that is used. This study indicated that the Windows Media Server delivers more than 80% of its media using TCP, while Real Media Server and Quicktime Server only deliver about 38% and 36% of media using TCP respectively.

Guo *et al.* [31] studied traces in 2004 from a large number of websites, they observed that a large fraction of multimedia content is downloaded using HTTP and TCP. The traces examined at the server side showed that 87% of the video clips were downloaded, 9.6% used HTTP streaming and only 2% used true streaming

using UDP. Client side traces from the Gigascope [13] database showed that 64.7% of streams were downloaded, 24% Pseudo-streamed, and 10% used true streaming. While these results may seem surprising as it is widely believed that UDP based protocols are ideal for streaming, similar results were also observed by Van Der Merwe *et al.* [78], after analyzing traces from a commercial site. They reported that most of the video bytes seen used TCP, which accounted for 70% of the bytes for both live and on-demand video. The remainder of the data was transmitted using UDP-based protocols. The percentage of clips using TCP for streaming is higher than what was reported by Sripanidkulchai *et al.*[75] because the latter only considered live workloads.

Guo *et al.* [32] analyzed the quality and resource utilization for streaming media delivered over the Internet in 2005. They collected traces from thousands of home and business users. They observed that *fast streaming techniques*, which stream video clips at higher bitrates than their encoded rates are widely used on the Internet, accounting for 50% and 21% of the traffic observed for home and business clients respectively. Their experiments suggested that using fast streaming leads to higher CPU utilizations at the server. Their results also show that shorter video clips have a smoother playback compared to longer clips. For instance, 87% and 59% of clips for home and business users respectively had smooth playback for clips that were 30 seconds or less. For clips longer than 300 seconds, 56% and 19% of clips for home and business users respectively had a smooth playback. The observed behavior could be due to the fact that network conditions change when the video is being streamed. If the duration of the video is longer, it is more likely to experience these changes, leading to re-buffering and packet loss.

The recent findings reported by Guo *et al.* [32] and Sripanidkulchai *et al.* [75] refute the argument that only unreliable protocols are useful for streaming media as the greater percentage of streaming sessions use TCP.

A measurement study specific to wireless streaming was conducted by Kuang and Williamson [47]. They analyzed the transmission of RealMedia over an 802.11b wireless network in 2004. The video clips were encoded at 180 kpbs and 71 kpbs

respectively. The measurements were conducted in four wireless channel scenarios; *excellent* (greater than 75% signal strength), *good* (45-75% signal strength), *fair* (20-45% signal strength) and *poor* (less than 20% signal strength). They found that after retransmission of data at the MAC layer, the effective data loss was zero for the excellent and good channel categories, 1.3% for the fair category, and 28.3% for the bad category. The high data loss for the bad category resulted in low video quality.

Hillestad *et al.* [36] conducted a simulation study using the NCTUns simulator to investigate the capacity of 802.16 networks for streaming video. They considered a rural network scenario where Subscriber Stations (SS) stream video clips from a server located on the wired Internet. The video clips used were encoded as H.264 media at bitrates between 500 and 900 kbps. RTP was used to deliver the videos. Their results showed that 802.16 can support 9 or 10 SS streaming media at average bitrates of 750 kbps successfully. However, this study did not indicate how the media would be affected in the midst of high packet loss and congestion.

After observing these characterization and measurement studies, two things are clear. First, packet loss causes severe degradation to video data [9]. It is also known that wireless networks experience substantial data loss [9, 47]. It is expected that these high data loss levels will have a negative impact on encoded video clips and will result in propagation errors. The data loss levels observed in previous measurement studies are used as a rough guide in selecting packet loss rates in the experiment section. Second, it is evident that TCP is not only a dominant protocol for traditional Internet applications but for streaming media streaming applications as well. One of the reasons TCP is used is that it is a reliable protocol, as result it will deliver good quality video. Another reason is that it is easy to implement and can traverse over networks with firewalls. In spite of its advantages, TCP has some shortcomings when used for streaming media. It cuts its sending rate after losing data and implements an ARQ scheme which retransmits every lost packet. This can result in major delays in times of congestion. To reduce the impact of these delays, client buffers are used. Even so, the user has to wait for a long time to fill

these buffers before the video can be played. It can be expected that most of the videos streamed from the Internet to wireless clients will be streamed using TCP, unless clear benefits at low cost-of-use can be shown for alternatives. Since wireless networks are usually used as a last hop from a wired network, it is not clear how much effect TCP will have on the quality and delay of the video clip in the presence of wireless errors.

In order to overcome the high delays TCP presents, due to its ordered delivery and packet retransmissions of lost data, a streaming technique which uses two parallel connections is used in this thesis. Furthermore, the highly sensitive video frames are transmitted reliably to protect them from wireless packet losses, thereby reducing the amount and effect of errors.

3.2 Wireless Video Streaming

Some studies [3, 47] have shown in particular that wireless networks have a higher error rate than wired networks due to fluctuating channel conditions. Due to this, the use of wired streaming techniques is not always straightforward when used on wireless networks. Various streaming methods have been proposed to deal with the wireless streaming conditions. In this section, various works concerned with wireless streaming are discussed.

Chen and Zakhor [10] observed that the wireless network is usually underutilized when streaming media. With that idea in mind, they described a transmission scheme that seeks to utilize the available bandwidth efficiently when transmitting non-scalable streaming media information. They showed that using one TCP or TCP-Friendly Rate Control (TFRC) connection underutilizes the available bandwidth, and then proposed using multiple TFRC connections for a given streaming wireless application. Their results showed that using more than one connection efficiently utilizes the bandwidth. However, the dynamic change in the number of connections may lead to fluctuations in video quality.

Cranley and Davis [12] studied the performance of video traffic transmitted over

an 802.11b network in the presence of background traffic. They used the Darwin Media Server[2] to stream MPEG-4 video traffic using RTP on top of UDP. A traffic generator called MGEN was used to generate the background traffic. Their results showed that using a small packet size for the background traffic results in the video stream being starved earlier during playback than when large packets are used for the same load. They also showed that number of contributing sources to background traffic has little impact on the video.

Huang *et al.* [38] studied the transmission of streaming media over the wired Internet to wireless networks. They proposed a proxy-based approach that splits the connection into two parts: wired and wireless. The approach is based on the observation that congestion usually occurs in the wired Internet where there is competition with other flows. Therefore, a protocol with congestion control, such as TCP, should be used on the wired part of the connection. The wireless part of the connection should use a protocol without any congestion control. Using TCP only on the wired network shields TCP from cutting its sending rate due to non-congestion losses on the wireless network. While this may improve the performance by reducing the buffering delay, it is unclear how it may perform in regard to the quality of the video in cases where there is a high loss rate on the wireless network.

Other approaches used to improve video streaming over wireless have considered making modifications to the wireless MAC layer [20, 73]. Fitzek and Reisslein [20] examined the use of unmodified TCP over wireless. In order to hide the errors that occur on the wireless channel from TCP, they used multiple wireless channels to reduced MAC layer retransmission delays. Their results showed that using multiple channels for video transmission significantly increases the performance of video over wireless networks. Singh *et al.* [73] proposed another link layer method that works together with a modified transport layer protocol. Specifically, they used a modified UDP (called UDP-lite), which ignores the packet checksum if data loss occurs in the insensitive part of the packet. Packets that lose data in the sensitive part of the packet (such as the header) are dropped just like in traditional UDP. This means

[2]http://developer.apple.com/opensource/server/streaming/index.html

that if the header of a packet is received correctly, the packet will be delivered even if part of the payload is corrupt. On the link layer, the point to point protocol (PPP), a protocol responsible for checksumming and point to point communication was also modified to ignore checksums just like UDP-lite. Relaxing the checksum at the link layer and transport layer allowed them to deliver partially corrupted packets to the decoder. Their results showed that using their approach resulted in shorter end to end delays and lower packet loss rates compared to traditional UDP. It can also be speculated that their approach will deliver slightly better video quality than traditional UDP since it delivers partially corrupt data that would not be delivered by UDP.

The streaming scheme used in this thesis is similar to the one proposed by Chen and Zakhor in that it uses more than one connection. However, they differ in their goal and implementation. Chen and Zakhor's approach aims to increase the throughput over a wireless network, while the goal of this work is to improve the quality of the video in the midst of wireless packet losses. The approach used in this study is also different because it delivers the video file using two levels of reliability. The most important video frames are transmitted reliably while the remainder of the video frames are transmitted using best effort.

3.3 Error Control and Recovery

A number of methods have been used for error control and recovery for video applications. Video error control methods fall into three broad classes: error control algorithms in the decoder, forward error control [7, 42, 87, 88] and retransmission [14, 65, 69].

Decoder-based error control methods were studied by Gringeri [29]. Specifically, built-in MPEG-4 error mitigation techniques, such as data partitioning, RVLC and FEC were used to localize errors, recover data, and cancel the effect of errors. They reported that built-in MPEG-4 error control techniques can reduce the effect of errors for small error rates but require considerable overhead for multiple packet loss

scenarios. Given that wireless networks have bursty packet losses, built-in MPEG-4 error correction methods are unlikely to help much.

Bolot and Turletti [7] utilized a FEC-based error control method that attempts to minimize the visual impact of packet loss at the receiver. The method adapts the data produced by the video application to the characteristics of losses over the network. FEC methods are effective if the loss is predictable. It is not clear what the impact will be on the quality of the delivered video since wireless networks have been reported to exhibit bursty packet losses [3].

Liu and Claypool [53] presented a forward error correction scheme that aims to improve the quality of the transmitted video in the presence of packet loss. Before a video clip is transmitted, it is encoded into two versions with different quality levels. One is encoded with high quality and the other with low quality. During transmission, each high quality frame is piggybacked with a low quality frame version of the previously seen high quality frame. For instance, a high quality version of frame F1 will be piggybacked with a low quality version of frame F0. At the receiver, the decoder primarily decodes the high quality frame versions if they are present. If a high quality frame version cannot be decoded due to corruption or packet loss, the low quality version of the frame is used. To evaluate their scheme, video clips were transmitted over UDP using this scheme and the resulting clips were evaluated by a group of 42 viewers. The results showed that using redundant data improved the perceptual quality of the video. For example at 1% and 20% data loss, users reported an improvement of 20% and 65% respectively in video quality.

The experiment was conducted in an environment where only 4 consecutive packets were lost. In a wireless environment with high channel and congestion losses, the number of consecutive packet losses may be higher. In this case, the effectiveness of the scheme may be reduced especially if key frames are lost.

Another common error control method is retransmission. Traditionally retransmission was rejected as a suitable error correction method for streaming media applications due to the delay it introduces. To test this assumption, Dempsey *et al.* [14] investigated the feasibility of performing retransmission based error control

for multimedia applications. An analytical model and empirical measurements were used to show the effectiveness of retransmission for multimedia packets. They used latency and retransmission probabilities for evaluation in their study. The results show that retransmission is a feasible solution with a small amount of start-up delay. Furthermore, they observed that it is sufficient to use a fixed start-up delay at the receiver, rather than adapting the start-up delay according to fluctuating network delays.

Rhee [69] proposed a retransmission based scheme that uses retransmitted packets even after their playout times have passed. The scheme does not add any delay in the playout times. Frames are displayed at their normal playout times without delay. A packet that is lost during transmission will be retransmitted. If the retransmitted packet belonging to a frame is received before the frame is played, it will be used to restore the frame before the playout time. If a packet belonging to a frame arrives late, the frame will still be displayed at the normal playout time without the lost packet. However, the late packet is not discarded. It can be used to stop the propagation of errors to the dependent frames. This is because reference frames are used as the basis for decoding future frames. A reference frame that is restored even after its play out time has passed may still be useful to its dependent frames.

Sinha and Papadopoulos [74] presented an algorithm for performing multiple retransmissions for continuous media. They observed that failed retransmissions take a long time to detect because they have to wait for the retransmission timers to expire before another retransmission can be initiated. They went on to propose a *timerless* algorithm for quick detection of failed retransmissions. In their unicast algorithm, packets from the sender have two sequence numbers; the sequence number of the packet and the sequence number of the highest NACK so far. A gap in the sequence number indicates that a packet has been lost. A failed retransmission is detected when there is a gap in the sequence numbers of the highest NACK so far contained in the data packets. After a gap is detected in the sequence numbers, a retransmission is requested if there is sufficient time. Their results showed that

timerless protocols provide better performance than timer-based protocols in regard to packet loss.

Hou *et al.* [37] presented a protocol that conducts multiple retransmissions of lost video frames on a wireless network. The protocol first buffers packets at the base station before sending them to mobile clients. At the base station, packets are divided into equal segments before being transmitted. At the receiver, delivered segments are acknowledged or a NACK is sent to trigger retransmissions. After a NACK is sent, the protocol performs multiple retransmissions of lost segments based on the frame type and its location in the GOP. For instance, in a video clip with a GOP sequence of IBBPBBPBBPBB, the retransmission retries for I-,P1-, P2-, P3-, and B-frames could be assigned as 3, 3, 2, 1, and 1, respectively.

Results show that when this protocol is used, the probability to discard I-frames is lower than the probability to discard B-frames. This will translate into better quality video because receiving I-frames limits error propagation. The results also show that using this protocol results in a higher frame rate than when UDP is used. There was no congestion introduced into the study. When congestion is introduced, it might lead to more packets being dropped which might affect the frame rate. Since lost data is retransmitted, more dropped frames may also increase the start-up delay necessary to maintain smooth playback.

Feamster and Balakrishnan [17] described a system that enables the adaptive unicast delivery of video by responding to varying network conditions and dealing with packet losses. Specifically, they focus on a conditional reliability scheme called *selective retransmission*. The protocol extends RTP by allowing it to retransmit lost packets. Thier work primary focused on wired networks, while this thesis focuses on wireless networks that have high packet loss probability.

Grinnemo and Brunstrom [30] presented PRTP-ECN, a partially reliable protocol which is an extension to TCP. The protocol differs from TCP in the way it reacts to packet loss. The application specifies a reliability level, which must be maintained, called the *required reliability* level. A measure of the reliability so far is called *current reliability* and is used in the retransmission decision. Upon detection of a packet loss,

the protocol decides whether to retransmit or not to retransmit the lost data based on the reliability level set by the application. If a packet is lost, and the current reliability is greater than the required reliability, the packet will be acknowledged to avoid retransmission. Acknowledging lost packets might result in congestion because TCP uses duplicate acknowledgments to indicate congestion; to avoid this, explicit congestion notification is added to the PRTN algorithm to indicate congestion. In cases where an out of sequence packet is received and current reliability is less than the required reliability, the last packet received in sequence will be acknowledged just like in TCP. This will result in a retransmission of the lost packet.

In the related work on error correction, it has been shown that retransmissions are often used for error control but it is not clear if completely retransmitting all the packets is a feasible solution for wireless networks. In this thesis, retransmission-based error control is used. Instead of transmitting all the lost packets, only packets that contain the most important data are retransmitted.

3.4 Video Quality

Rejaie *et al.* [68] proposed a quality improvement scheme that adds or removes a layer of the encoded video stream to control the quality. Adding an extra layer improves the quality of the video. A layer is added to the transmitted video when the instantaneous bandwidth exceeds the consumption rate of the decoder. A layer is dropped if the buffering at the receiver becomes less than the estimated required buffering due to a reduction in bandwidth.

Krasic *et al.* [46] proposed a reliable streaming protocol that demonstrates a simple encode once, stream anywhere model where a single video source can be streamed across a wide range of network bandwidths. The system uses priority dropping in order to adapt to network conditions. A priority mapping algorithm, which processes a group of data units and prioritizes them according to their data dependence rules, is used to assign priorities to packets before they are sent. When there is congestion, the timeliness of the stream is maintained by dropping low

priority data units at the sender, before they reach the network. This reduces the amount of data that has to be sent over the network in the midst of insufficient bandwidth. Dropping low priority data packets ensures that available bandwidth is used to transmit high priority data. By dropping only low priority data, the quality of the video can be degraded gracefully in the midst of congestion.

Tripathi and Claypool [77] presented a scheme that reduces the amount of video transmitted by performing content aware scaling to avoid data loss in the presence of congestion. Scaling was conducted by either dropping less important frames (temporal scaling), such as B-frames or by re-quantizing the DCT-coefficients with a larger quantization factor thereby reducing the quality (quality scaling). The algorithm decides on the type of scaling to use based on the amount of motion present in the video clip. The motion in the video is measured by analyzing the percentage of interpolated micro-blocks in B-frames. A high percentage (greater than 45%) of interpolated blocks represents low motion while a low percentage (less than 45%) of interpolated microblocks represent high motion.

Evaluation of video quality was performed by a group of users. The results show that content-aware video scaling can improve the perceived quality as much as 50%. However, this is only if the motion characteristics are the same for the entire file. As a full movie may contain a mixture of different motion characteristics, it is unclear how the algorithm will perform in such scenarios. In wireless networks, where congestion is not the only source of packet loss, reducing the data sent may not be very helpful because even if the data transmitted is reduced, data will still be lost due to wireless channel losses.

Bolot and Turletti [8] used a mechanism that adapts video encoding according to the bandwidth requirements and packet loss. When the data is received, receiver reports (RR) are sent to the sender. The RRs are from all destinations in the multicast group. The information in the RRs is used to estimate network congestion and the quality of the received video at destination. The maximum output rate of the video encoder is also adjusted according to the information in the RRs. Results show that the mechanism produced reasonable degradation in video quality during

long periods of network congestion.

A video improvement scheme specific to wireless networks was proposed by Zhao *et al.* [91]. The scheme stores the video data into two separate queues (wireless transmission buffers) at the link layer. For the instance presented in the study, I-frames are stored in the high quality queue, while P frames are stored in the low quality queues. The clip used does not contain any B-frames. The two queues are assigned different priorities, one high and the other low. Data in the high priority queue is assigned a bigger number of retransmission retries (i.e 4) while data from the low priority queues is assigned a lower one (i.e 3). A simulation was conducted implementing the algorithm. Results show that using this scheme reduces the errors that occur on I-frames. This results in the overall quality of the video being improved. While the protocol improves the quality of the video, retransmitting all the lost data may result into a high waiting time for the viewer.

This thesis differs from the above mentioned works because it improves the video quality by reducing the effect of propagation errors caused by packet losses. The method used to prevent the propagation of errors is similar to the one used by Krasic *et al.* [46]. It takes advantage of layered video encoding and protects important I-frames from errors. However, the work by Krasic *et al.* [46] differs from this thesis in the following ways; their approach drops packets to attempt to prevent congestion from occurring in the network. In a wireless environment, packet loss can occur not only from congestion but wireless losses as well. In such a case, the video will encounter additional packet loss thereby reducing the quality. Secondly, they only transmit data using a single connection. This results in less important packets blocking more important packets due to TCP's ordered delivery algorithm.

3.5 Summary

This chapter has presented related work on workload characterization, wireless streaming and error control. After considering the related work various questions arise. Some of these questions discussed in this thesis are outlined below.

- It has been shown that progressive download and downloading are popular media delivery techniques for wired networks, as indicated by the amount of streaming sessions utilizing them [46, 81]. The impact of using downloading (using TCP) or progressive download (using TCP) on video quality over a wireless network is yet to be discussed.

- Using more than one connection has been shown to increase the throughput during streaming. Does using more than one connection offer an improvement in video quality in a wireless environment as well?

- Video is made-up of sensitive and non-sensitive data. Are there any benefits in progressively downloading or downloading sensitive information with high reliability, and streaming non-sensitive data using best effort protocols?

Chapter 4

Protocol Specification and Methodology

This chapter presents the *mixed streaming* technique and the experimental environment. It begins by outlining the advantages and disadvantages of common streaming techniques. It then gives the specification of the mixed streaming technique. The remainder of the chapter describes the workload, metrics, tools, simulation environment and experimental design used in the study.

4.1 Mixed Streaming

4.1.1 Features of Common Streaming Techniques

There are currently three popular techniques for delivering streaming media: downloading, progressive download, and streaming. The download and progressive download techniques use TCP to deliver multimedia. TCP is attractive because it is a standard protocol, which is not blocked by firewalls and is friendly to other flows. TCP is also used because it delivers high quality video, especially in low data loss environments. This is because it provides reliability features, such as retransmissions and congestion control. When a packet is lost due to congestion or channel losses, TCP retransmits the lost packet and cuts its sending rate. While this is likely to work well in network scenarios with very small or no data loss, it results in a big delay for high loss environments such as wireless networks. The delay emanates from TCP's ordered delivery which ensures that packets are handed to the receiver in a sequential order. This means that current packet losses add a delay to future packets. In encoded video applications, this will result in less important

video frames blocking the more important I-frames. Figure 4.1 illustrates the transmission of a video file with an associated transmission schedule (TS). The frames are arranged in transmission order as specified in Section 2.3. In a transmission system without data loss, the frames will be transmitted according to the transmission schedule. Figure 4.2 shows how TCP's transmission schedule is affected after losing some data. Losing frames P_2, B_4 and B_{10} results in the overall transmission schedule being extended by three time slots, assuming it takes one time slot to retransmit a lost frame. For instance, B_{13} is transmitted at time slot 16 instead of time slot 13. It can also be noticed that, losing less important frames result in frame I_{11} being delayed by two time slots. In addition to extending the transmission schedule, TCP will assume the packet loss is due to congestion even when it is from bad channel conditions. As a result TCP will cut the transmission rate to half in order to control congestion [49]. This will further extend the transmission schedule.

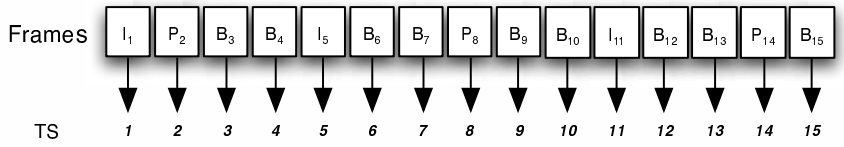


Figure 4.1: Sequential transmission of a video clip without data loss

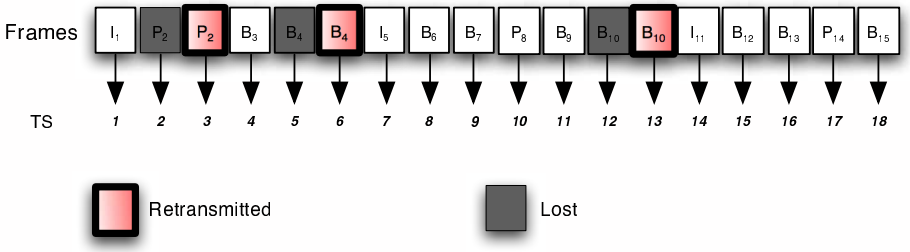


Figure 4.2: Sequential transmission of a video clip with data loss

To eliminate the delay associated with downloading via TCP, UDP is used for streaming. UDP maintains the timeliness of the stream by not retransmitting any lost packets. Due to this, UDP transmits the video according to the TS even in the midst of packet loss. One disadvantage of UDP is that it can be over aggressive

if there is no rate control. Another weakness of UDP lies in the fact that it does not restore lost data. This results in dependent frames suffering from propagation errors, which leads to low video quality. This problem is made worse in high packet loss environments, such as wireless networks. In order to overcome some of the limitations of true streaming and downloading, such as low quality due to propagation errors and high delay due to retransmissions respectively, the *mixed streaming* approach, which transmits the video using two connections is used in this thesis.

4.1.2 Specification of Mixed Streaming

To stream a video clip using the mixed approach, the video file is split and stored into two parts. The first file contains the highly sensitive video data (I-frames) and the second file is composed of the less sensitive video data (P and B-frames). The highly sensitive data is progressively downloaded using a reliable TCP connection and the less sensitive data is streamed unreliably using UDP. The I-frames are classified as highly sensitive data because losing an I-frame will result in the whole GOP not being decoded correctly. This results in propagation errors that can persist for a long time. The I-frames are streamed back to back while the less sensitive P and B-frames are streamed when they are needed. Figure 4.3 illustrates the mixed streaming technique in the presence of packet losses. To simplify the illustration it is

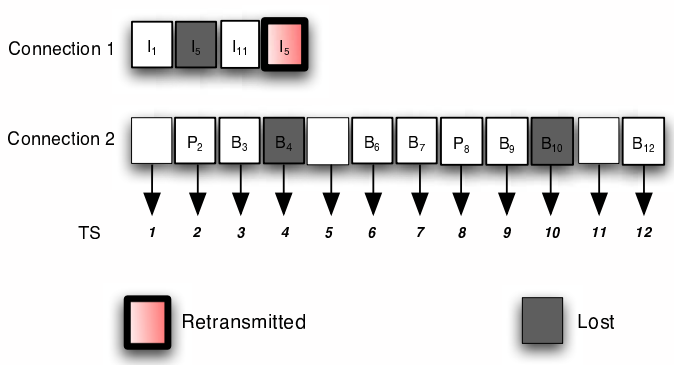


Figure 4.3: Transmission of a video clip using mixed streaming

assumed that each frame will fit into a single packet. While this may be true for B-

frames and P-frames, it is unlikely to be the case for the I-frames. However, the net result is the same since delayed packets will still result in delayed frames for TCP. It can be observed in Figure 4.3 that I-frames are not blocked by less important P and B-frames because they are transmitted on a separate transport layer connection. An additional advantage of transmitting the I-frames on a separate connection is that lost I-frame packets can be retransmitted without altering the original playout schedule. For instance, frame I_5 is initially sent at time slot 2, and then resent at time slot 4 but is not needed until time slot 7. As a result, I_5 can be retransmitted in time for playback.

It can also be noted that the loss of P and B-frames does not add delay to the original transmission schedule. The frames on connection 2 are transmitted over UDP according to the time specified by the encoder. This means that when a video clip is streamed with the mixed technique, it is likely to have a short start-up delay compared to downloading and progressive download.

4.2 Experimental Environment

4.2.1 The Workload

The workload refers to the demands or requests placed on a system that is being evaluated or analyzed. The workload forms an important part of any performance study, because it has a huge influence on the results of an experiment. For a system to be evaluated correctly, the workload used in an experiment should be representative of the traffic on the actual system. The considerations necessary in workload selection can be found in the literature [40].

In this thesis, the workload to the simulator is a video file trace. The trace file is produced by first encoding a raw uncompressed video file to MPEG-4. The MPEG-4 video file has all the information that is required to create a video trace file. An MPEG-4 parser is used to parse the encoded video file and extract the required data to produce a video trace file. The content of a sample trace file is

0	H	52	62
1	I	92	9884
2	P	132	5917
3	B	175	271
4	B	215	260
5	P	255	7121
6	B	295	1590
7	B	335	1573
8	P	375	5625
9	B	415	264
.	.	.	.
.	.	.	.
.	.	.	.

Figure 4.4: Contents of a sample video file trace

shown in Figure 4.4. Each line in the trace file includes the following fields: the *frame number* that identifies each frame, the *frame type* showing the type of frame, the *frame size* indicates the length of the frame in bytes and *time* in milliseconds indicates the time at which the frame was produced. One of the advantages of using a video file trace is that it can easily be incorporated in simulators, since it only contains the information about the video file, which is relevant to the volume and timing of the traffic.

4.2.2 Performance Metrics and Factors

4.2.2.1 Performance Metrics

The first performance metric that is used in the study is the *Peak Signal to Noise Ratio(PSNR)*. The PSNR is the most common method used to measure video quality because it is easier than other methods. For the interested reader, Wolf and Pinson [86] provides a detailed discussion of other video measurement techniques. PSNR

shows the objective quality of each frame using a single number. The PSNR is considered for the luminance component because the human visual system is more sensitive to the luminance component than the chrominance component [85]. PSNR is calculated as follows for an 8-bit luminance channel:

$$PSNR = 10 \times log_{10}(\frac{255^2}{MSE}), \tag{4.1}$$

where the MSE is the mean square error and is calculated using the following equation:

$$MSE = \frac{\sum_{\forall,i,j}[x(i,j) - \hat{x}(i,j)]^2}{M \times N}, \tag{4.2}$$

where M and N are the dimensions of the frame in height and width respectively and $x(i,j)$ and $\hat{x}(i,j)$ are the original and reconstructed pixel luminance or chrominance values at position (i,j).

Another performance metric used to measure the video quality is the *frame rate*. Generally, the number of correctly received frames, which is used to calculate the frame rate, can be obtained in three ways. The first approach would be to get a list of correctly received frames from the video evaluation tool before the video is decoded. While this gives an indication of the number of frames that are received correctly over the network, it does not exclude dependent P and B-frames that can not be decoded because of lost reference frames. Dependent P and B-frames need to be excluded from the frame count if their key frame is lost because the player cannot play them correctly.

The second approach is to get a trace from the decoder indicating which frames have been decoded successfully. Most decoders do not give an accurate trace of decoded frames because partially decoded frames and predicted frames are sometimes not included in the trace.

Another method is to set a PSNR threshold (i.e. 20 dB) and treat all frames that fall below the threshold as damaged. This works well for individual video files although it is difficult to have a common threshold for video with different activity

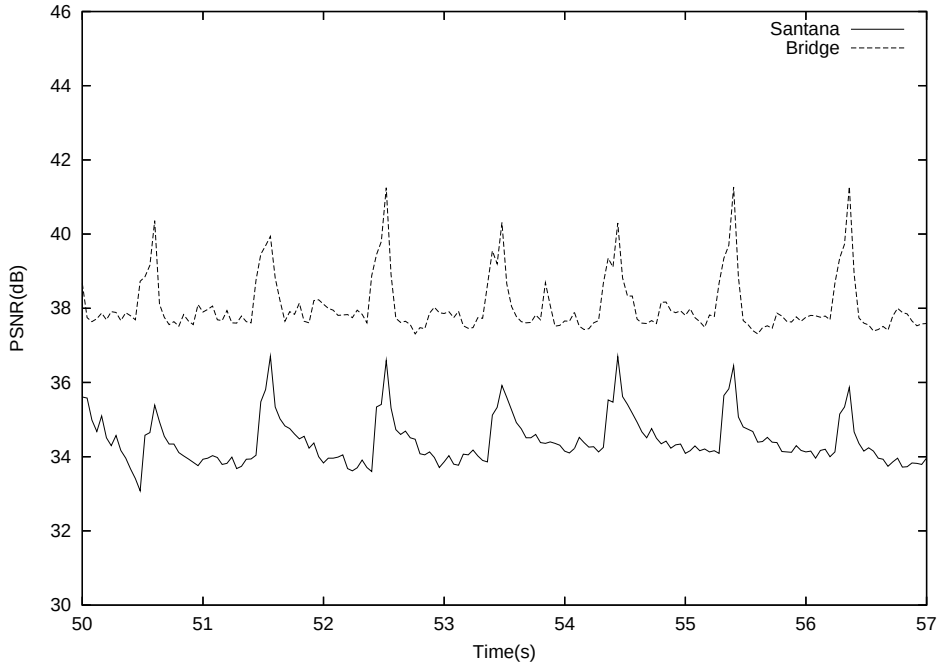


Figure 4.5: PSNR values for Santana and Bridge video clips

levels because individual video sequences have different PSNR values. For example Figure 4.5 shows the PSNR values of two video sequences; Bridge and Santana. Even though the encoded video clips did not lose any packets, they have different PSNR values. The reason for this is that the compression method used to encode the video is lossy. During decoding, the lost data results in a difference between the original video and the decoded one. If a lossless compression method is used for encoding the videos, there will be no difference between the decoded video and the original one. This will make the MSE zero, resulting in the PSNR value being undefined. Due to PSNR differences between different video files, a PSNR threshold is set for each individual video by observing the PSNR value at which the video becomes unwatchable. This is the approach that is used in this thesis.

Secondly, the performance of the schemes will be studied in terms of the *delay* that is introduced during transmission. Video transmission requires packets to arrive at the receiver by a specific time before they are needed for display. If a packet is delayed or misses its playout deadline, it will cause undesirable jitter. Packets that

are lost in the network are also considered late because they miss their playout deadline. The Internet does not have a fixed network delay, but instead delivers data on a best effort basis. In order to ensure continuous display of frames, a start-up delay is introduced at the receiver. In this thesis, the start-up delay required for a smooth display and the number of delayed packets will be considered.

4.2.2.2 Factors

Video transmission performance is affected by many factors. In this study, the following factors are examined: packet loss, start-up delay, video type, delivery method and the wireless bandwidth.

Packet loss - When data is transmitted over the network, it experiences packet loss due to buffer overflows, link errors and congestion. Packet loss has a negative effect on compressed video data because it introduces propagation errors. If the packet loss rate is high, the encoded video will not be decoded successfully.

Congestion - Congestion in the network occurs when more packets arrive at a router than it can handle. Packets that can not be transmitted right away will be placed in the router's buffer. When the buffer is full, packets will be dropped causing retransmissions in retransmission-based protocols. In order to reduce packet loss due to congestion, congestion control algorithms can be used [21].

Start-up delay - Due to varying network delays, multimedia systems allow clients to wait for a short time before beginning playback. The start-up delay is needed to fill a buffer before playback in order to allow smooth display. A larger start-up delay allows smooth playback, but it also means that the viewer has to wait for a long time before viewing the video.

Video type - Video files are classified according to the type of action in the video. Video clips with low action can be compressed effectively because they contain a lot of redundant data. On the other hand, video clips with a lot of action have less redundancy. Therefore, the compression gained in high action videos is more limited than in low action clips.

Delivery method - Video data is delivered using different methods depending on

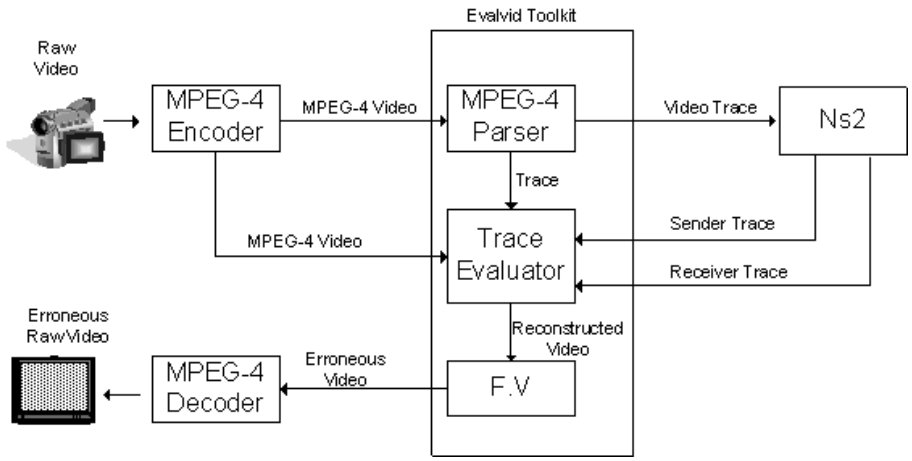


Figure 4.6: Experimental tools and setup

the properties desired for the clip. If a high reliability is needed, a video clip will be delivered using a reliable protocol, such as TCP. If a minimum delay is the major focus during delivery, UDP streaming is used. In this thesis, three delivery methods will be considered.

4.2.3 Experimental Tools

In this study, various tools are used for simulation and evaluation. Figure 4.6 shows the tools and experimental setup used in this thesis. Each of the individual tools are discussed briefly below.

4.2.3.1 Encoder

The raw videos for experiments are stored in YUV format. A video is usually compressed for efficient storage and transmission. An encoder is used to encode the raw video into a desired format. In this thesis, an MPEG-4 encoder called FFMPEG [59] is used to compress the raw video into an MPEG-4 video. The parameters used when encoding the raw video to MPEG-4 are summarized in Table 4.1. The FFMPEG encoder is publicly available online [18, 59].

Table 4.1: Parameter settings for the encoder

Parameter	Setting
High Quality Settings	enabled
B-frames	2
VOP sequence	IBBPBBPBB...
GOP	25
Error Concealment	enabled
Error Resilience	enabled
Codec	MPEG-4

4.2.3.2 The Simulator

The *ns-2* [23] network simulator is used for protocol performance evaluation and to provide a simulated network environment for video transmission. Substantial support for simulation of transport protocols, wireless networks, routing protocols and queuing algorithms is provided by *ns-2*. One of the facilities that *ns-2* offers is the wireless network implementation. The wireless network implementation is based on the CMU wireless extensions, which have now been incorporated into the main *ns-2* distribution. The implementation supports all layers of the Internet. For the Physical layer, it simulates propagation models (such as TwoRayGround for longer distances and Friss-space attenuation for shorter distances), radio interfaces and the antennas. At the Link layer, *ns-2* supports the 802.11 MAC and ARP protocols. For the network layer, it supports routing protocols, such as, Ad hoc On Demand Distance Vector (AODV), Dynamic Source Routing(DSR)[1]. The MAC used is the generic 802.11, propagation model is TwoRayGround, interface queues are drop tail and the routing protocol used is Destination-Sequenced Distance Vector (DSDV).

Figure 4.6 shows how *ns-2* is used in the experiments. The simulator is used to conduct a trace-driven simulation. Before an experiment can be conducted in a trace-driven simulation environment, the trace must be produced. In this study, the trace is produced by the video evaluation tool. The trace is then fed into the simulator for processing. After processing the video trace, *ns-2* produces a packet trace of received packets, that can be used to analyze the video.

[1]http://www.isi.edu/nsnam/ns/ns-documentation.html

4.2.3.3 EvalVid Tool-Set

The Evalvid [43] tool-set provides routines that are used for parsing and evaluating videos. Evalvid's trace evaluation module was adapted so that it can parse traces generated by *ns-2*. Three modules are significant in this study: the parser, the trace evaluator(ET) and the FV module.

The MPEG-4 parser reads in an MPEG-4 encoded bitstream and produces a trace of the video. The trace that is produced by the parser includes all the information that is necessary for a simulator to simulate the transmission of the video over the network.

The ET is used to analyze traces produced by a network simulator. It reads in the trace of the packets sent, the trace of the packets received and the encoded MPEG-4 video. Since the packets have unique IDs, the trace evaluator can identify the lost packets by comparing the two trace files produced by the simulator. Each packet is then assigned a type. The video with errors is reconstructed packet by packet, using packets that were received correctly. If the first packet of the frame is lost, the entire frame will be dropped because a frame can only be decoded successfully if the frame header is present.

The received video may have some missing frames due to packet loss. Since video quality assessment is performed on a frame by frame basis, the received video must have the same number of frames as the original one to make assessment possible. The FV module takes a reconstructed video produced by the trace evaluator as input and inserts a previously decoded frame for every lost frame in the video for easy quality assessment. The output of the FV module is a reconstructed video, which has the same number of frames as the original video. The FV module is not used in this thesis because the FFMPEG decoder replaces lost frames with neighboring frames.

Table 4.2: Characteristics of the video sequences

File	Santana	Matrix	Silent
Resolution	CIF(352 x 288)	CIF	CIF
Duration	15 min	12 min	12 min
Bitrate	200 Kbps	200 Kbps	200 Kbps
Frames	22500	18000	18000
Action	high	alternating	low
FPS	25	25	25

4.2.3.4 Video Sequences

Three raw video files grabbed at a rate of 25 frames per second are used for the simulations. The video files are encoded from YUV format to MPEG-4 using FFMPEG. The MPEG-4 files are parsed by Evalvid to generate three traces which are classified according to the activity in the video. The video activity levels that are used are high, alternating, and low. For the high action category, the Santana video is used. The video clip contains scenes of a musical concert with people dancing and singing. For the alternating action category, a video clip from the movie "The Matrix" is used. It contains a combination of fast paced chases and low action scenes of a person seated on a computer. For the low action category, the Bridge video clip is used. The video contains a scene of a bridge with water under it. The videos are encoded at 200 kbps and have a resolution of 352 x 288 pixels. The remaining characteristics of the video clips relevant to the analysis of this thesis are shown in Table 4.2.

4.2.4 Simulation Environment

The *ns-2* simulator generates its traffic by reading a video trace file that is produced by the MPEG-4 parser of the Evalvid tool-set. The frame sizes from the trace file are used to create network packets with the appropriate sizes.

The simulation experiments are conducted using a dumbbell network topology as shown in Figure 4.7. This simple network topology is similar to the one used by Balakrishnan *et al.* [5]. The sources are wired nodes while the sinks are wireless

nodes. R1 is a router and R2 is a base station node.

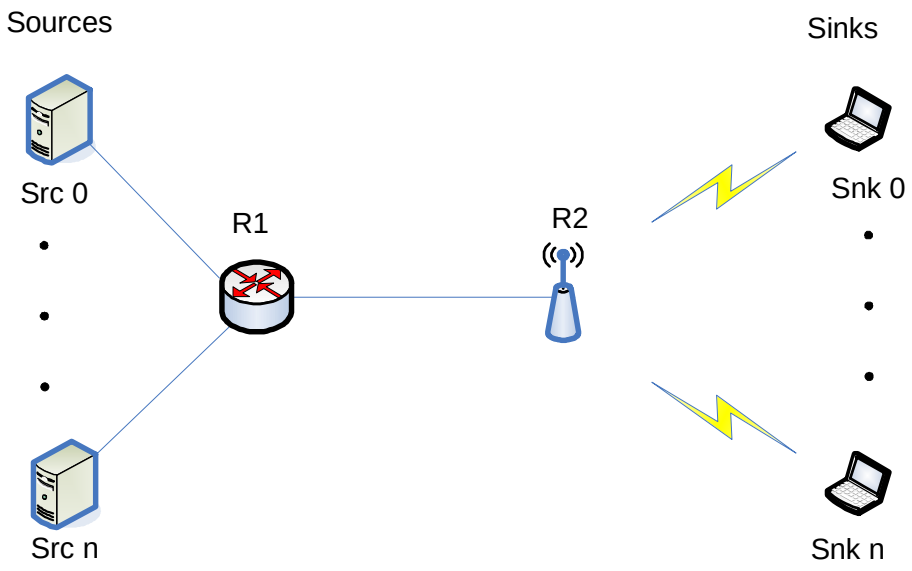


Figure 4.7: Topology

The link capacity between each source and R1 is 1 mbps and 1 mbps between R2 and each sink. The sources and the sinks are connected through a common bottleneck link between the R1 and R2. The bottleneck capacity is set to 0.7 mbps unless specified differently. The routers use FIFO queuing with drop-tail queue management and have a queue length of 30 packets. The queue length has a value that is larger than the bandwidth delay product of the bottleneck link. Setting this parameter very low will cause packets to be dropped when the queue is full and setting it too high will increase the latency. The queue length should be greater than or equal to the bandwidth-delay product [2]. The one-way propagation delay is set to 2 ms between source 0 and R1, 100 ms for the bottleneck link. Measurement studies have shown that typical propagation delays on the Internet can be 100 ms or more[6].

The network link between R2 and sink 0 is a 802.11 wireless network with a propagation delay of 2 ms. The remainder of the configurations for the wireless network are summarized in Table 4.3.

Table 4.3: Wireless network configuration

Parameter	Value
MAC	802.11
Propagation	TwoRay Ground
IFQ	Droptail
Antenna	OmniAntenna
Interface queue	60
Routing	DSDV

4.2.5 Error Model

The wireless channel is characterized by lossy periods and loss-free periods. The losses on the wireless channel are caused by link impairments, such as, multipath fades and impulsive noise. To capture the bursty nature of the wireless channel, the wireless channel is usually modeled with Markov models [45, 4, 61, 63]. One of the commonly used models is a two-state Markov model introduced by Gilbert [28].

In this thesis, the wireless channel is modeled using the Gilbert model. Figure 4.8 shows the Gilbert model state transition diagram. The Gilbert model has two states, the good state G and the bad state B. In the good state, the channel is assumed to have no loss; therefore no errors are introduced. State B corresponds to a lossy channel, as a result errors are introduced. The transition from one state to the next follows a transition probability matrix shown below:

$$P = \begin{pmatrix} P_{GG} & P_{GB} \\ P_{BG} & P_{BB} \end{pmatrix}$$

where P_{GG} is the probability of staying in the good state, P_{GB} is the probability of changing from the good state to the bad state, P_{BG} is the probability of changing from the bad state to the good state, and P_{BB} is the probability of staying in the bad state. The probabilities are assigned the following values $P_{GG} = 0$, $P_{GB} = 1$, $P_{BG} = 1$, $P_{BB} = 0$. The time spent in the good state and the bad state is set dynamically for each packet loss rate. The time spent in the good state is between 0.75 to 1 ms and the time spent in the bad state is between 0 and 0.25 ms.

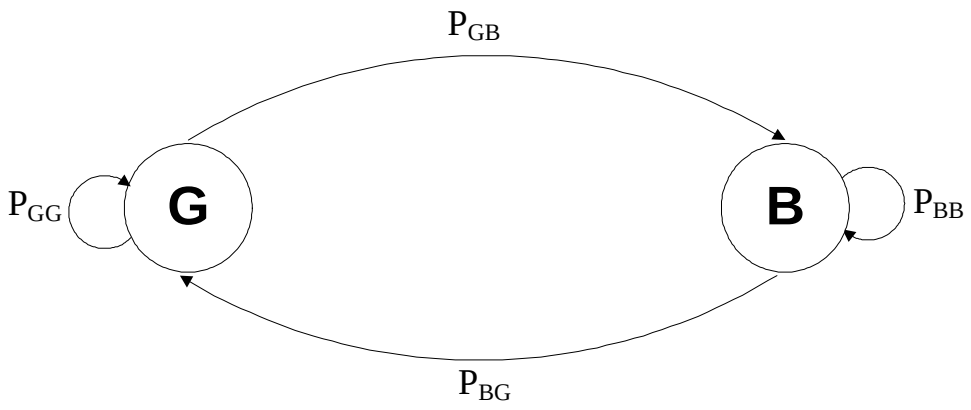


Figure 4.8: Gilbert model state transition diagram

4.2.6 Experimental Design

Experiments are performed in two phases. The first set of experiments considers the quality of the delivered video in the presence of packet loss and congestion. The objective of these experiments is to quantify the effect of propagation errors on video quality when unreliable and mixed protocols are used.

The quality of the video is evaluated using the PSNR. The quality is assessed after some packets are lost in the simulated network. For each video clip, three transmission schemes will be tested; reliable transmission using TCP, unreliable and mixed transmission. Three different packet loss levels are considered: 5%, 15% and 25%. The selection of these packet loss rates is guided by measurement studies [9, 47], which have shown that wireless networks have a wider range of packet loss rates than wired networks. Ten runs are conducted for each transmission scheme at specified packet loss levels.

For the experiments involving congestion, three levels of congestion are used, UDP congestion, TCP congestion and TCP/UDP congestion. Ten experiments are conducted for each congestion level at predefined packet loss rates. A summary of the experiments that are conducted for the video quality is shown in Table 4.4.

It is expected that video data which is delivered using transmission schemes implementing some reliability will have a higher quality compared to those with less

Table 4.4: Video quality experimental configurations

Video	Congestion	Packet loss levels	Delivery levels
Santana	no congestion	no loss	TCP/ UDP/MIXED
	TCP congestion	no loss	
	UDP congestion	no loss	
	Mixed congestion	5,15,25	
	no congestion	5,15,25	
Matrix	no congestion	no loss	TCP/ UDP/MIXED
	TCP congestion	no loss	
	UDP congestion	no loss	
	Mixed congestion	5,15,25	
	no congestion	5,15,25	
Bridge	no congestion	no loss	TCP/UDP/MIXED
	TCP congestion	no loss	
	UDP congestion	no loss	
	Mixed congestion	5,15,25	
	no congestion	5,15,25	

reliability. As a result, TCP transmission will have the best quality and mixed transmission will have the next best quality. UDP is expected to have the lowest quality. Each experiment is replicated 10 times except the first experiment with no loss and no congestion. This is because in a clean channel there is not a difference between runs.

The second set of experiments examines the delay. The delay is measured using the start-up delay. The experiments are first conducted without any loss. Next they are conducted in the presence of packet loss and congestion. The experiment is replicated 10 times for TCP and mixed. Only mixed and TCP are considered for the delay experiments because UDP imposes very minimal delay as a result it is not necessary to consider it. The experimental configuration of the delay experiments are shown in Table 4.5.

The mixed streaming approach is expected to have a shorter start-up delay because most of its packets are transmitted according to the transmission schedule. The only delay that mixed streaming is likely to experience is from transmitting the reliable portion of the stream. Since the mixed packets are transmitted before they are needed, they can accommodate some delay without missing the playout

Table 4.5: Video delay experimental configurations

Video	Congestion	Packet loss levels
Santana	no congestion	no loss
	Mixed congestion	5,15,25
	no congestion	5,15,25
Matrix	no congestion	no loss
	Mixed congestion	5,15,25
	no congestion	5,15,25
Bridge	no congestion	no loss
	Mixed congestion	5,15,25
	no congestion	5,15,25

deadline. On the other hand, TCP is expected to have the largest start-up delay because its transmission schedule is likely to be extended beyond the streaming time when there is significant packet loss because TCP will invoke its congestion and flow control algorithms which will prevent addition packets from entering the network until the assumed congestion is over.

Chapter 5

Video Quality

The goal of this chapter is to evaluate the video quality delivered by mixed streaming and UDP. As mentioned in the previous chapter, the video quality is evaluated using two metrics: Peak Signal to Noise Ratio (PSNR) and frame rate. For the PSNR evaluation, three sets of experiments are conducted. First, the quality is observed on a clean channel, next packet loss is introduced, and finally congestion is introduced. The results in this chapter use three video clips having high action, alternating action and low action levels respectively.

5.1 Video Quality on a Clean Channel

Prior to studying the transmitted video quality in the presence of congestion and packet loss, the delivered video quality is examined without any congestion or packet loss. This is to establish the baseline video quality for all video clips used. The PSNR values are obtained by transmitting each of the video clips over the simulated network, and comparing the delivered video with the original. The network used in the study has the common dumbbell topology. The network has a one-way propagation delay of 104 ms and a bottleneck bandwidth of 0.7 mbps as indicated in Section 4.2.4.

The three types of video clips mentioned above were transmitted using UDP, mixed and TCP. Each video clip yielded the same PSNR value when transmitted with each of the transmission methods because there is no congestion or packet loss introduced into the experiment.

Table 5.1 shows the PSNR of each video clip in a clean channel. Since mea-

Table 5.1: Baseline PSNR values

Video clip	Average PSNR
Santana	36.6
The Matrix	44.9
Bridge	38.1

surements for each video clip are the same for each transmission method, only one PSNR value is shown. The measurements shown are from single runs for each clip. Observe that individual video clips have different PSNR values even though there is no packet loss or congestion. This can be attributed to MPEG-4's lossy compression algorithm, which removes fine details when encoding raw video clips [85].

5.2 Video Quality with Packet Loss

Data loss has a negative impact on the quality of the received video because it leads to impairments, such as pixelation effects and black spots in the video. Errors can occur on both dependent frames and independent frames. Errors that occur on dependent frames (P and B) last for a few frames while errors on independent frames are often propagated to dependent frames. In a case where the first independent frame is lost, the whole GOP will experience some errors.

In this section, the impact of packet loss on video clips transmitted by UDP and mixed streaming will be evaluated. Before the results of the experiments are discussed, the characteristics of the experiment are outlined. The bottleneck capacity, round trip time and the topology are kept as described in the previous section. TCP streaming sends video packets as fast as possible. For UDP streaming, video packets are delivered when they are needed (i.e according to the time specified by the encoder). The mixed streaming approach sends TCP packets as fast as possible and sends UDP packets when they are needed. The TCP component of mixed streaming utilizes TCP Newreno, because it is the TCP flavor currently in wide use [83]. As indicated in Section 4.2.6, packet loss is introduced into the experiment at 5%, 15% and 25% using the two-state Markov error model which represents packet

losses on the wireless network.

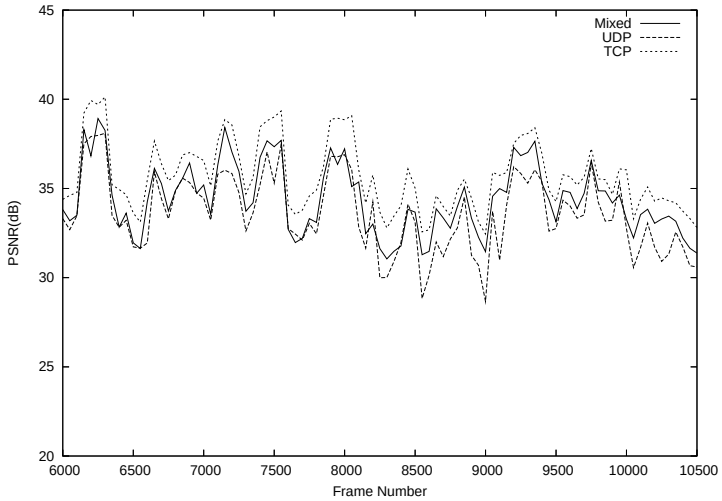
All the video clips are encoded at 200 kbps as specified in the previous chapter. Video clips are decoded at the receiver after being transmitted with default MPEG-4 error resilience methods enabled. Measurement of the video quality statistics start after a warm-up time of 20 seconds. TCP measurement experiments have shown that it takes about 20 seconds for TCP to be become stable [2].

Figures 5.1, 5.2 and 5.3 show the quality of three types of video clips delivered with UDP and mixed at various packet loss rates. The y-axis shows the PSNR of each frame. Since the PSNR is logarithmic and all the values are higher than 20 dB, the y-axis scale starts from 20 dB for Figures 5.1 and 25 dB for Figure 5.2. Figure 5.3 starts at 34 dB because the values have a much smaller range. For the purpose of clarity, only 4500 frames are shown for each plot. Each plot point is an average of 50 frames, this is to make the graphs legible. Each plot shows the average of 10 runs.

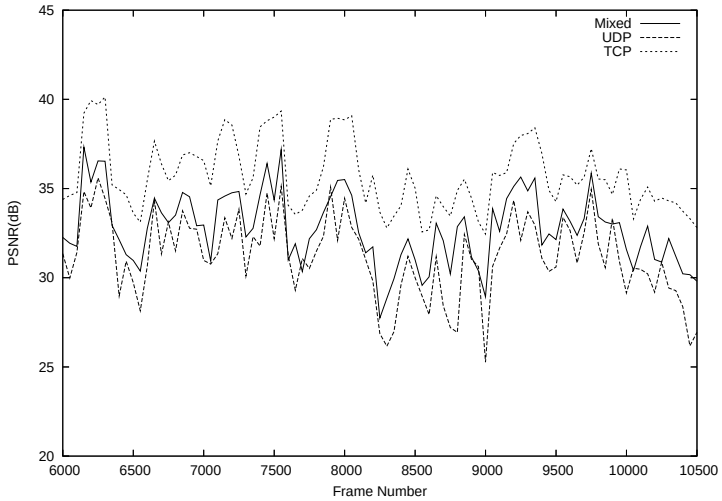
To give a detailed view of the impact of packet loss on videos with different action levels, each video clip is presented in its own graph. The results in Figures 5.1(a) and 5.2(a) show that 5% packet loss reduces the video quality delivered by mixed and UDP. Despite MPEG-4 error control methods being enabled, UDP and mixed delivered video clips show some data loss. MPEG-4 is only effective in concealing errors at very small loss rates.

TCP delivers the highest video quality because it restores all the lost data. Mixed delivers the second highest video quality and UDP delivers the lowest quality. Mixed delivers video quality higher than UDP because it retransmits the lost I-frames. Retransmitting I-frames prevents errors from propagating for a long time in the video sequence, thereby improving the video quality.

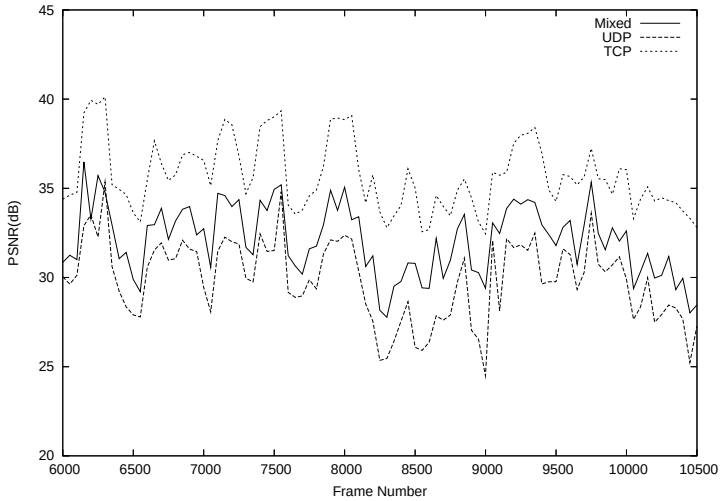
It can be observed from Figure 5.1(a) that there are instances where UDP delivers better video quality than mixed for some frames. At first, this seems impossible, but due to compression and the error model used such behavior is not unreasonable to expect. Another contributing factor is that each plot point is the average of 50 frames, meaning it may include more than 1 GOP.



(a) 5% packet loss

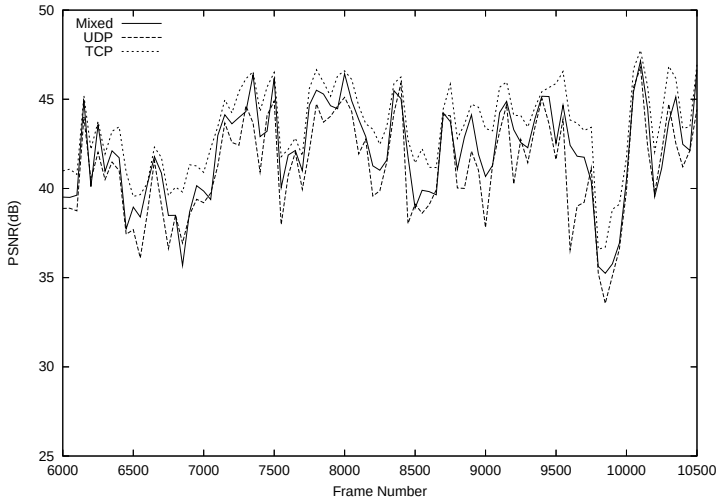


(b) 15% packet loss

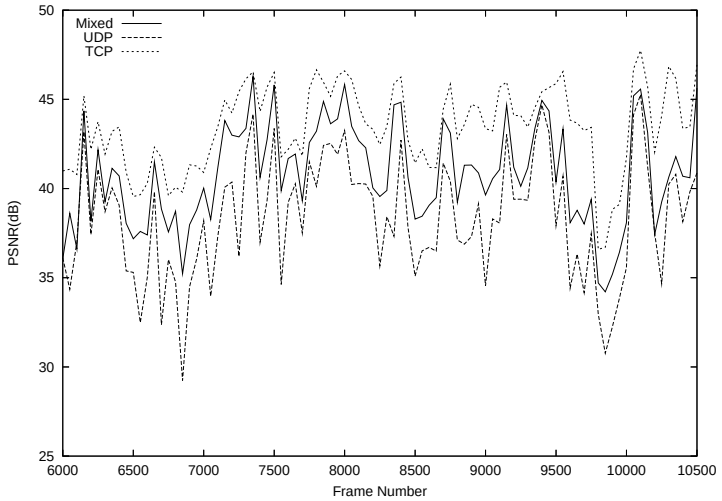


(c) 25% packet loss

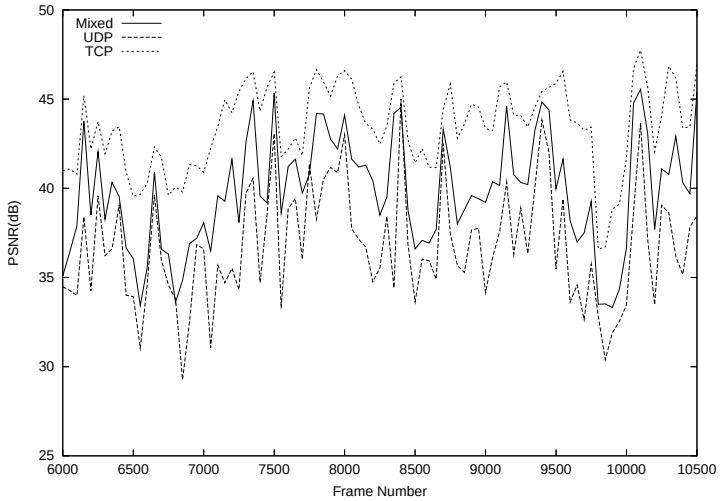
Figure 5.1: Delivered video quality of the Santana clip for various loss rates



(a) 5% packet loss

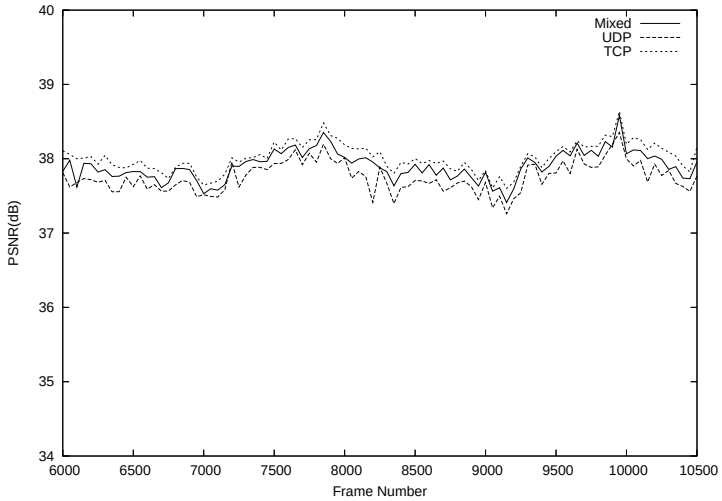


(b) 15% packet loss

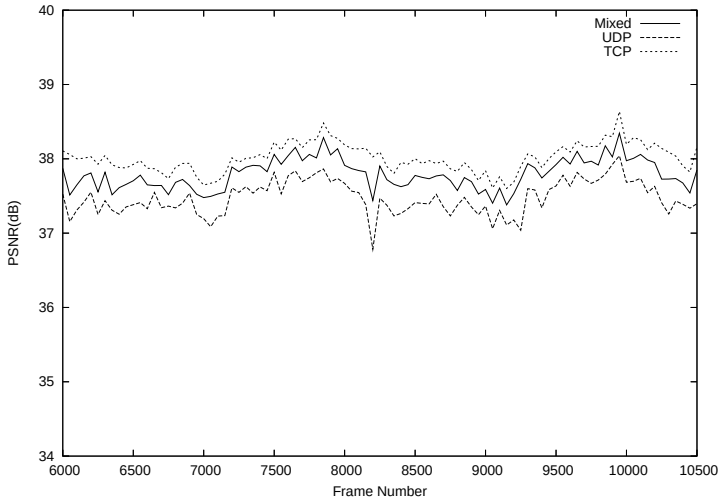


(c) 25% packet loss

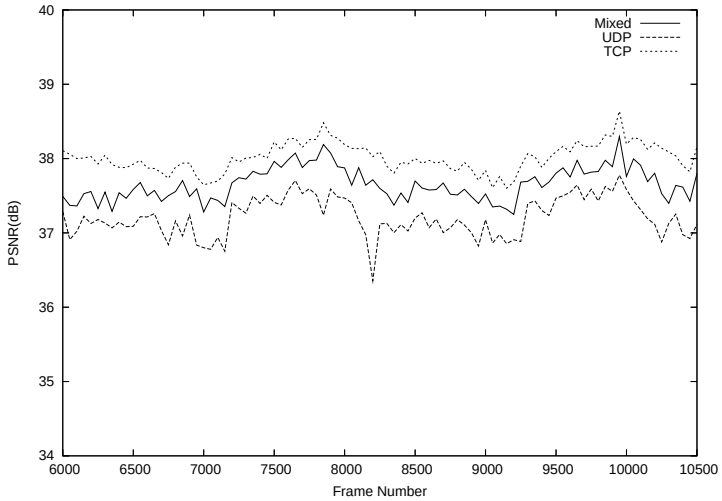
Figure 5.2: Delivered video quality for "The Matrix" clip at various loss rates



(a) 5% packet loss



(b) 15% packet loss



(c) 25% packet loss

Figure 5.3: Delivered video quality for the Bridge clip at various loss rates

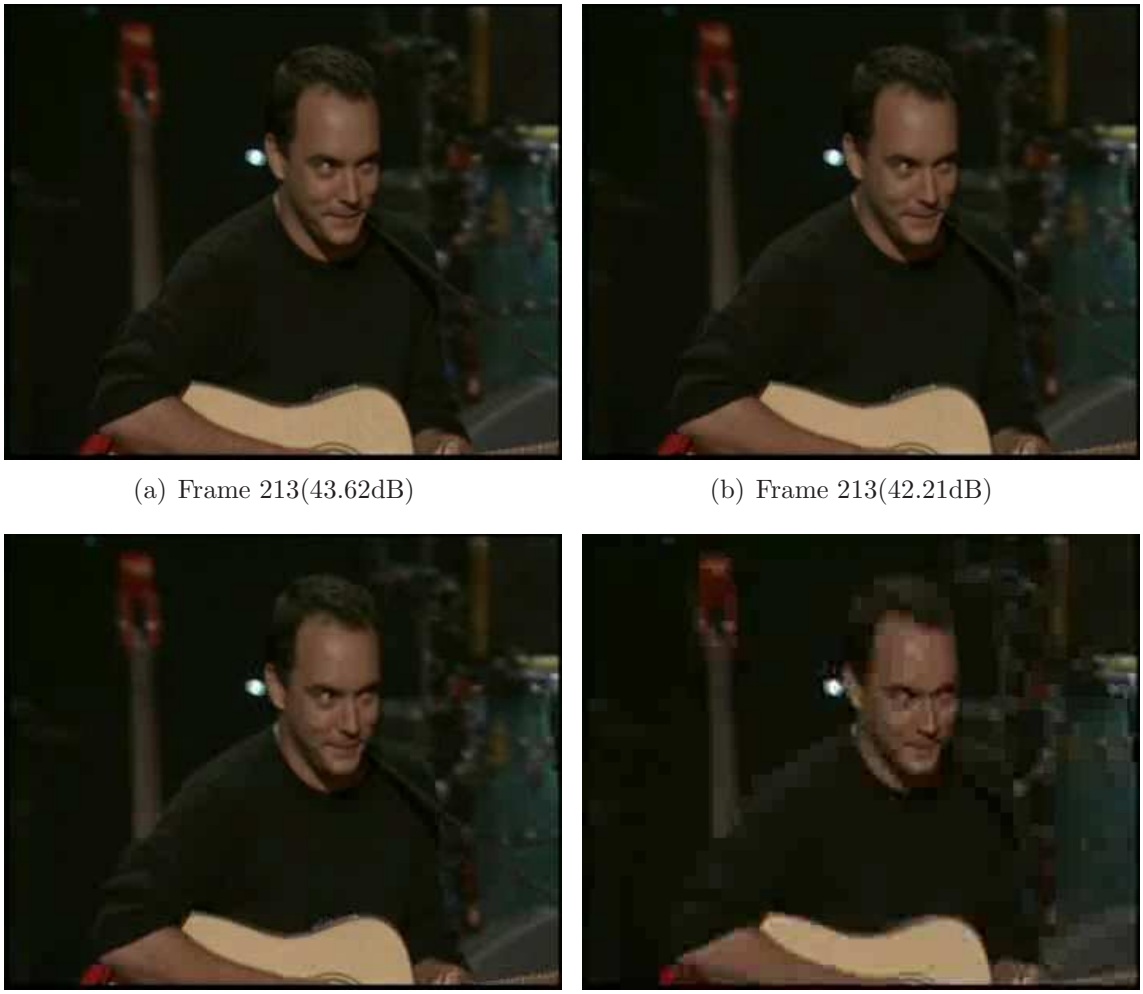
To further investigate why UDP delivers better quality than mixed for some frames, traces from the *ns-2* simulator and the Evalvid evaluation tool were examined. The traces from the simulator indicate that each individual run has a different loss profile. As a result, different frames are lost in individual runs. When the video is decoded, the lost frames will have a low PSNR. For instance, in Figure 5.1(a), mixed loses a P-frame between frames 7750 and 7800 in most of the runs, while UDP only loses the same P-frame in a few runs. Therefore, UDP has a higher PSNR average for the plot point than mixed.

The results indicate that the low quality periods extend for more than one frame due to the MPEG-4 compression algorithm. Losing data on an independent I-frame or P-frame will cause an error to propagate to other dependent frames. In Figure 5.1(a), the mixed stream loses P-frames between frames 7925 and 7950. This causes errors to propagate lowering the PSNR values of subsequent dependent frames.

When the packet loss rate is raised to 15%, video clips for all action levels show a drop in quality for UDP and mixed due to packet loss, while TCP remains the same. In Figure 5.1(b) and 5.2(b), it can be seen that the Santana and "The Matrix" video clips show the highest degradation in quality in comparison to Figure 5.3(b). This suggests that high action and alternating action clips are more susceptible to packet losses than low action video clips. The difference in PSNR values between mixed and UDP is higher in the Santana and "The Matrix" video clips than in the Bridge. A difference of more than 5 dB in some cases is recorded in Figure 5.2(b) while Figure 5.3(b) only records a difference of less than 1 dB. A difference of 1 dB may be visible, and consequently the MPEG committee utilized an improvement threshold of 0.5 dB to determine whether a coding feature was necessary to be included in the standard [33]. In order to give an estimate of the significance of dB differences, images with corresponding dB values are given in Figure 5.4.

At 25% packet loss, mixed delivers a higher quality video than UDP for all action levels. As the error rate increases, mixed delivers video quality higher than UDP.

The results shown in Figures 5.1, 5.2 and 5.3 only show a limited number of frames for each video clip as a result they do not give the overall picture of the



(a) Frame 213(43.62dB)

(b) Frame 213(42.21dB)

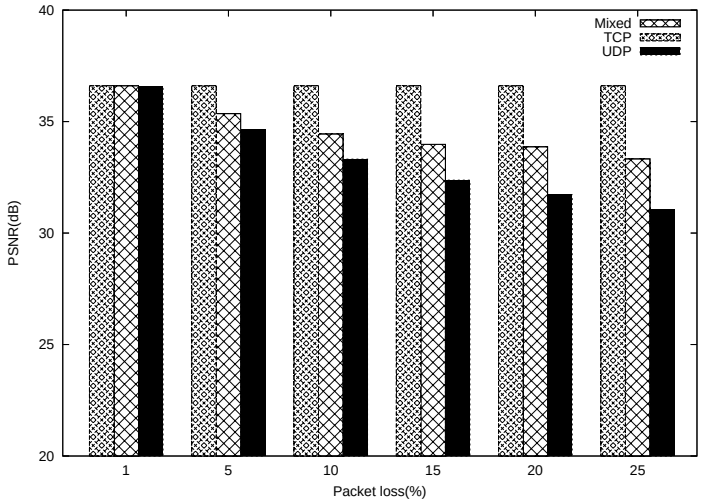
(c) Frame 213(40.45dB)

(d) Frame 213(35.32dB)

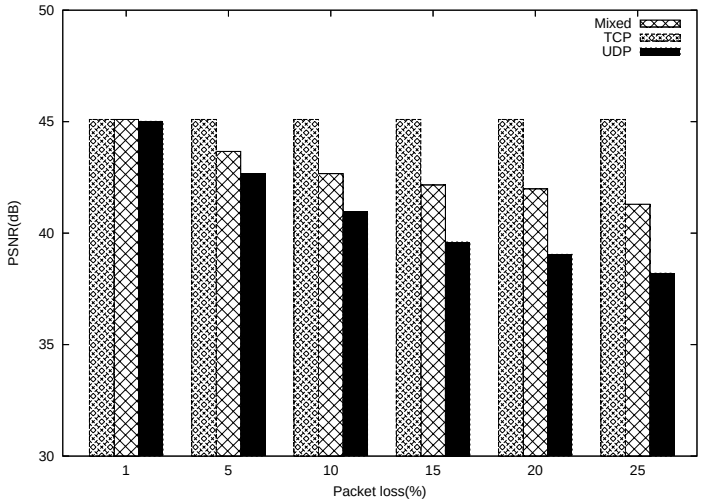
Figure 5.4: Video frame at different PSNR levels

quality. To study the overall delivered video quality in the presence of packet errors, the average PSNR is shown in Figure 5.5. The averages are obtained from 10 runs for each clip.

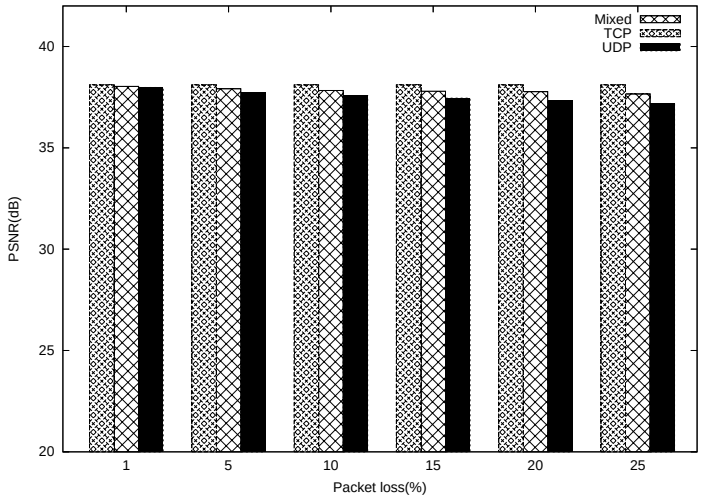
The findings presented in Figure 5.5 are consistent with those shown in Figures 5.1 to 5.3. It can also be observed that the difference in delivered quality between UDP and mixed transmission increases as the packet loss rate increases for all action levels. Mixed transmission shows a larger improvement for high action clips than for low action clips. For example, in the Bridge clip, an improvement of about 0.6 dB is recorded at 25% of packet loss while the "The Matrix" and Santana videos



(a) Santana video clip with packet loss



(b) "The Matrix" video clip with packet loss



(c) Bridge video clip with packet loss

Figure 5.5: Average quality at various packet loss rates (No congestion)

show an average improvement over 1.5 dB. This suggests that low action videos are less affected by packet losses compared to high action videos. This is because there is a lot of temporal redundancy between frames in low action videos. Since there is very little or no difference between successive frames, replacing damaged frames with previous frames is highly effective.

Figure 5.6 shows the percentage of frames that experience impairments and the magnitude of the impairments experienced. The experimental configurations are kept as described above earlier in this section. Each plot line shown in the graph is an average of ten runs. The y-axis is shown in logarithmic scale in order to capture a wide range of the values. The values on the x-axis are obtained by subtracting frame PSNR values of the original encoded video from those of a video with impairments. The graph only shows mixed streaming and UDP streaming because TCP streaming does not experience any impairments due to packet loss.

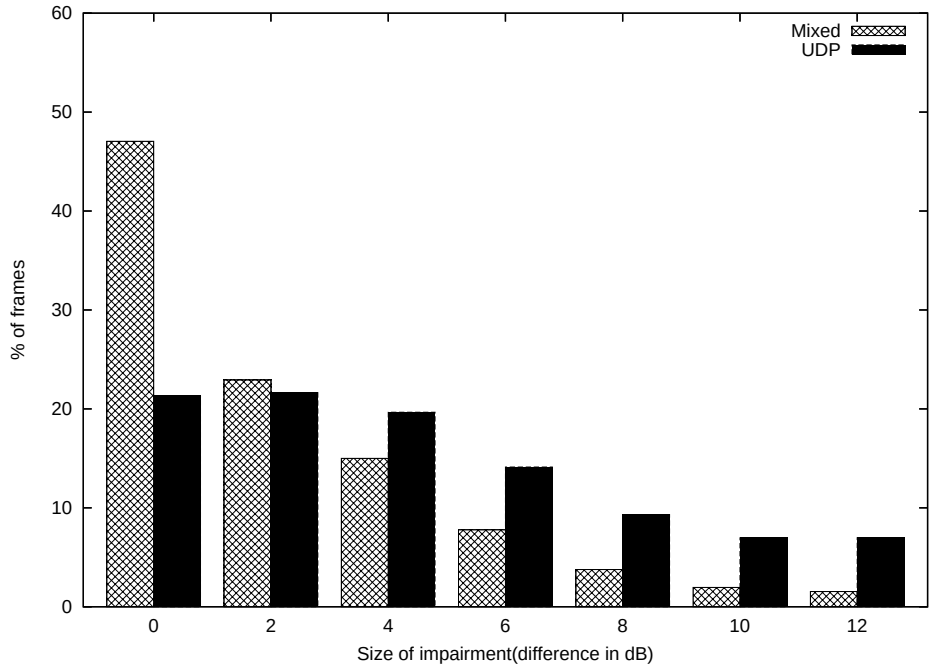


Figure 5.6: Distribution of impairments(15% packet loss)

The results in Figure 5.6 show that the mixed approach delivers a higher number of video frames without any impairments. Specifically, the mixed approach delivered

47% of frames without any impairments while UDP only delivered 21% of frames without any impairments. It can also be seen that UDP has a higher occurrence of frames with impairments greater than 4 dB. When the loss rate is increased, a similar pattern can be noticed, although the percentage of frames delivered without error reduces. The findings in this experiment complements the results that have already been presented and suggest that selective data recovery is beneficial for video data.

5.3 Video Quality with Congestion

In the previous section, video quality delivered by the mixed approach, TCP and UDP in the presence of packet losses was studied. Results suggested that TCP delivers the best quality while mixed streaming delivers a higher quality video than UDP when there is packet loss. However, these results did not incorporate any congestion losses. In this section, the performance is studied in the presence of congestion. Two types of congestion cross traffic are used in the experiments: UDP and TCP.

5.3.1 TCP Congestion

The experiments in this section illustrate how mixed streaming is affected by TCP cross traffic. An FTP bulk transfer is transmitted simultaneously to provide the TCP cross traffic. The TCP cross traffic uses Newreno with a packet size of 1000 bytes as specified in Section 4.2.4. The maximum bandwidth that can be taken by the TCP competing traffic is limited by the maximum window, although TCP may not reach the full window size if the flow experiences packet loss. The competing traffic is composed of 3 TCP flows with varying RTT times. The cross traffic is started 2 seconds after the video transfer is started. The video transmission and the competing traffic are transmitted through a bottleneck link of 0.7 mbps as specified in Section 4.2.4.

Figure 5.7 shows the video quality obtained with competing TCP traffic. The

graphs are presented using different scales on the y-axis to enhance readability. Mixed and UDP are both affected by congestion losses with UDP experiencing the most degradation for high action videos in Figures 5.7(a) and 5.7(b). The degradation in quality is low for the low action video in Figure 5.7(c).

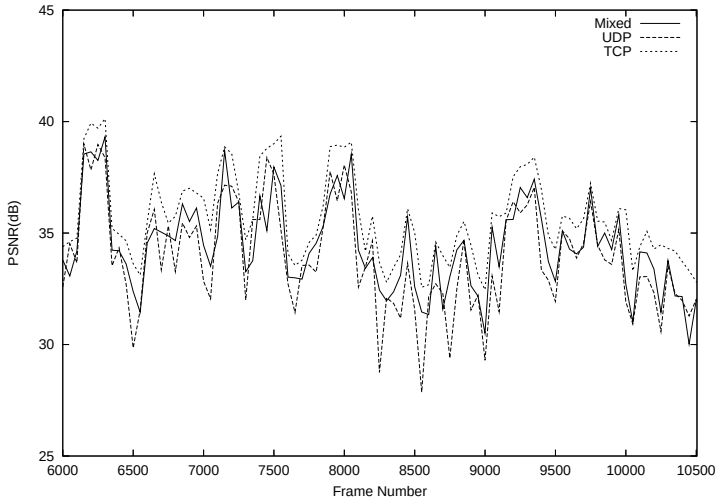
The results show that the video transmitted by UDP is more prone to congestion caused by TCP. As a result, it delivers lower quality video than TCP and mixed. In Figures 5.7(a), 5.7(b) and 5.7(c) UDP exhibits periods of low quality and periods of good quality. This behavior is due to the fact that TCP starts transmitting data at a low rate and increases its sending rate until it consumes all the bandwidth thereby causing congestion. After experiencing congestion, it loses data and cuts its sending rate. During the times when there is sufficient bandwidth, UDP does not lose packets due to congestion, as a result, it has a higher quality. When TCP uses up the bandwidth by increasing its sending rate, it creates congestion, which results in UDP losing packets.

The behavior exhibited by UDP is also observed when streaming with mixed. However, the effect of packet loss caused by congestion is reduced because mixed retransmits packets containing reference video data. In the case of TCP, all the lost packets are retransmitted; as a result, the effect of TCP congestion on quality is fully masked.

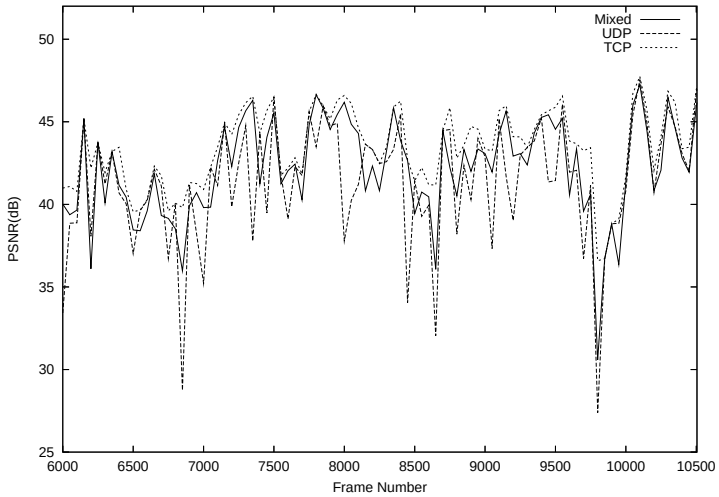
The results suggest that TCP and mixed streaming can offer better quality in environments that have high congestion resulting from TCP traffic such as the Internet. Although it should be noted that TCP will add a significant delay to the streaming because of its retransmissions and ordered delivery algorithm. A fuller treatment of the delay is given in the next chapter.

5.3.2 UDP Congestion

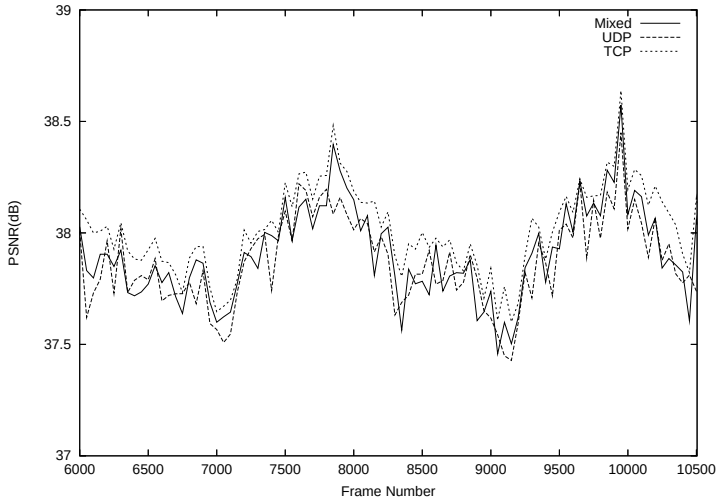
The results presented previously show the interaction between the video transmitted by mixed streaming and UDP with TCP cross traffic. The following experiment shows the reaction of mixed streaming and UDP streaming to UDP competing



(a) Santana video clip with TCP congestion



(b) "The Matrix" video clip with TCP congestion



(c) Bridge video clip with TCP congestion

Figure 5.7: Video quality with TCP cross traffic

traffic.

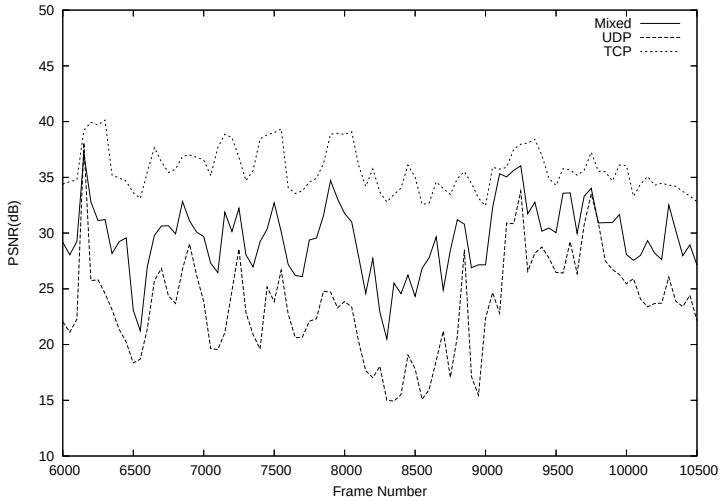
In order to evaluate the quality of delivered video in the presence of UDP cross traffic, experiments were conducted on a network with a bottleneck link of 0.7 mbps. The round trip time of the UDP flow is 104 milliseconds and it starts at the same time as the video transmission. The UDP traffic has a bit rate of 535 kbps and a packet size of 500 bytes. The packet size and bit rate are selected by slowly increasing their values until the network becomes congested. For purpose of this experiment, the network becomes congested when the competing traffic takes additional bandwidth causing video packets to be dropped.

Figure 5.8 shows the delivered quality of the video in the presence of competing UDP traffic. The figure shows that TCP delivers the best quality followed by mixed and then UDP when the link is congested with a UDP flow.

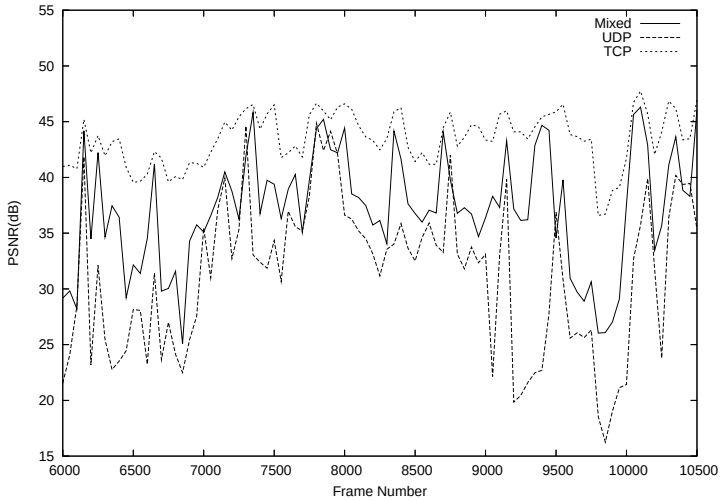
In the results presented, it can be observed that when the action level is high, the effect of the competing UDP cross traffic is also high. This is because errors are difficult to conceal in high action videos. In these types of videos, packet retransmission is more effective than using in-built error concealment methods. As a result, mixed delivers higher quality video compared to UDP in these scenarios. For low activity video clips, the difference in quality between mixed and UDP is small. This is because the gain in quality offered by restoring I-frames is minimal and closer to the gain produced by in-built concealment methods. This means that either mixed or UDP can be used to stream low activity videos in environments that are bandwidth limited by unresponsive UDP traffic. Experiments in this section have not taken into account the delay experienced by the various protocols. The effect of the delay on the video transmission is discussed in Chapter 6.

5.3.3 TCP Cross Traffic and Packet Loss

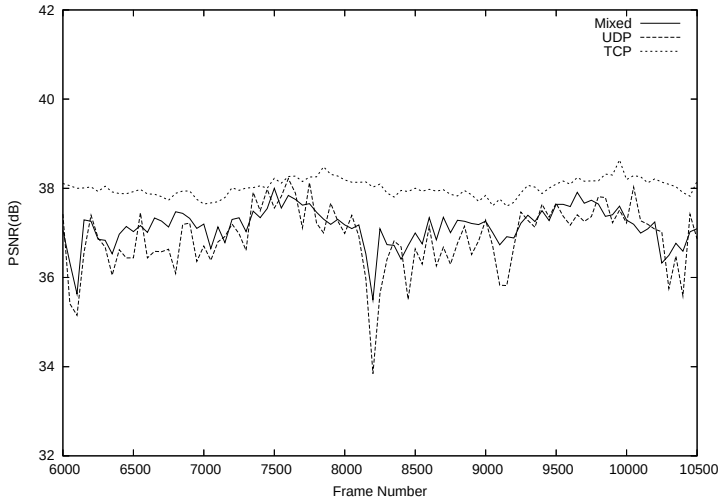
This section discusses the impact of both congestion and packet loss on video quality. The experimental environment is kept the same as in Section 5.3.1 unless stated differently. Only TCP cross traffic is used in this experiment because TCP is more



(a) Santana video clip with UDP congestion



(b) "The Matrix" video clip with UDP congestion



(c) Bridge video clip with UDP congestion

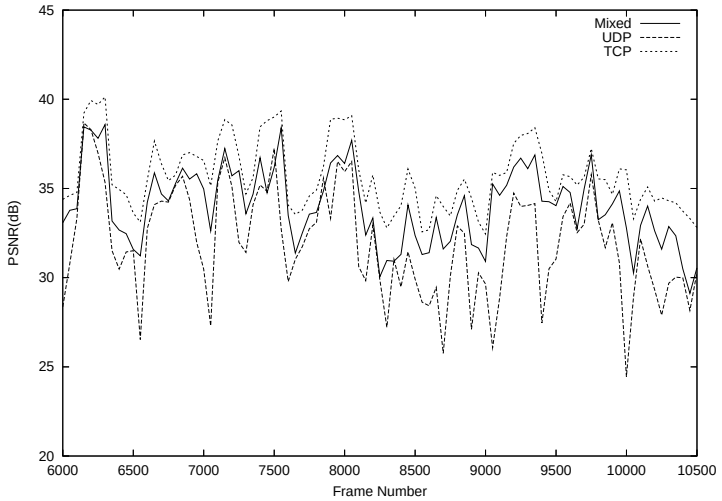
Figure 5.8: Video quality with UDP cross traffic

prevalent on the Internet than UDP [31]. The TCP cross traffic has an FTP source with a propagation delay of 104 ms. TCP Newreno with a packet size of 1000 bytes is utilized in the TCP flow. The bottleneck router can queue 30 packets and the bottleneck bandwidth is set to 0.7 mbps. Results are presented using the PSNR values for all three video activity categories. Results are also presented using the difference in PSNR due to congestion and packet loss.

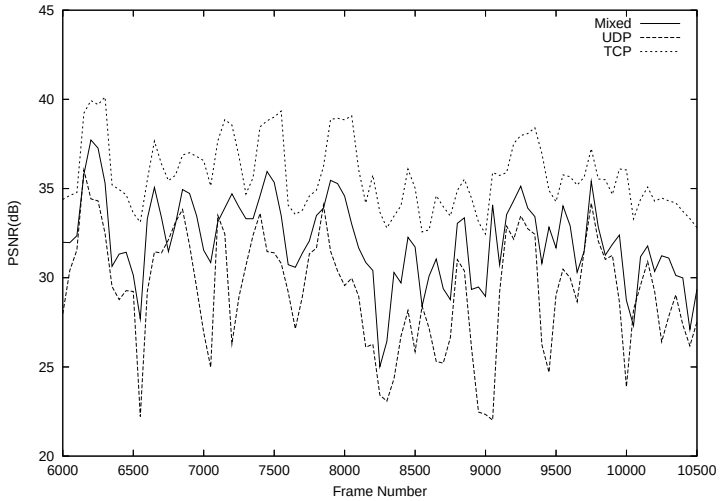
Figure 5.9 shows the effect of a combination of congestion and packet loss in a high action video clip. It can be seen from the graph that even when there is a combination of congestion and channel losses, TCP gives the best PSNR followed by mixed and then UDP. The results for the alternating action video clip in Figure 5.10 and for the low action video clip in Figure 5.11 show a similar behavior.

The results suggest that the recovery of lost key frames in mixed improves the quality of the video in the presence of both congestion and wireless losses. At 5%, 15% and 25% packet loss with congestion, the quality delivered by UDP is generally lower than mixed, because UDP loses some key frames in some runs causing errors to propagate to other dependent frames.

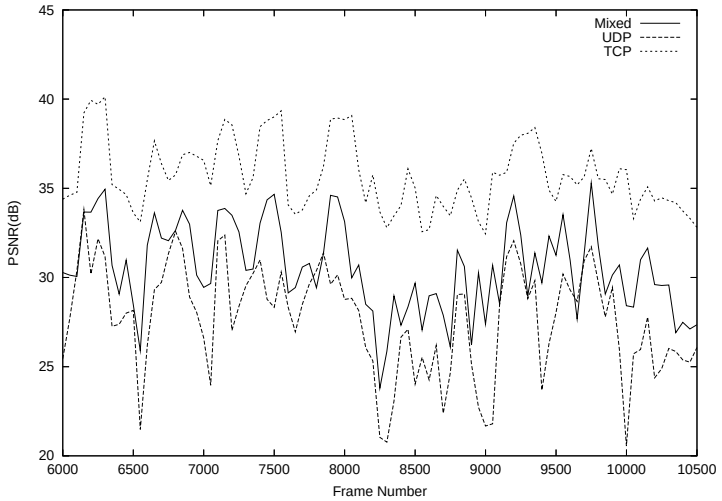
Figure 5.12 shows the overall impact of congestion and packet loss on each of the three classes of video clips. The results are presented using the difference in PSNR. The difference values are obtained by subtracting the PSNR with packet loss and congestion from the PSNR with packet loss. For all the three classes of videos, UDP generally has a higher difference than mixed; showing that the video clip delivered by UDP is more affected by congestion than the clip delivered by mixed for loss rates less than 25%. When the data loss rate is high however, mixed is more affected by the congestion than UDP. This is because the TCP component of mixed keeps reducing its sending rate due to the high packet loss rate. These results are consistent with the measurements shown in Figure 5.10.



(a) 5% packet loss with congestion

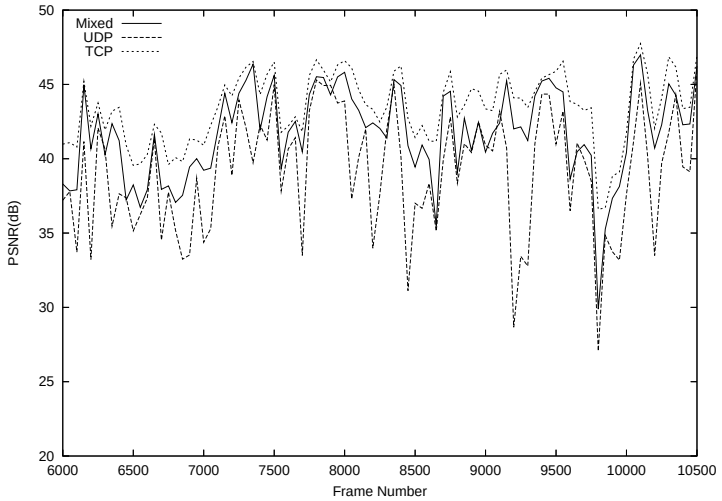


(b) 15% packet loss with congestion

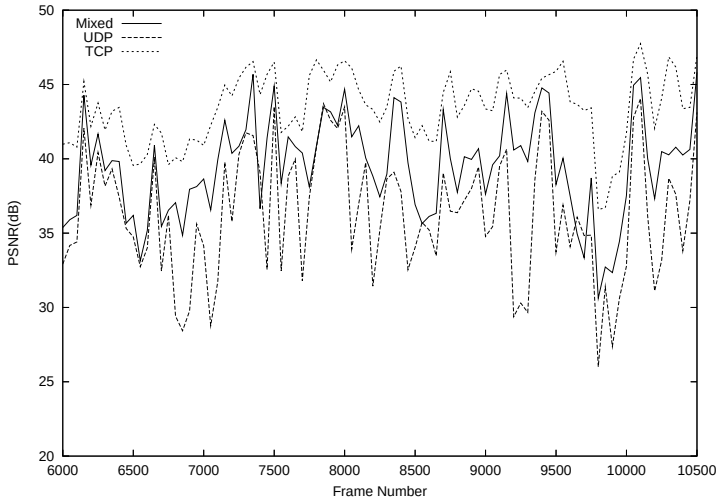


(c) 25% packet loss with congestion

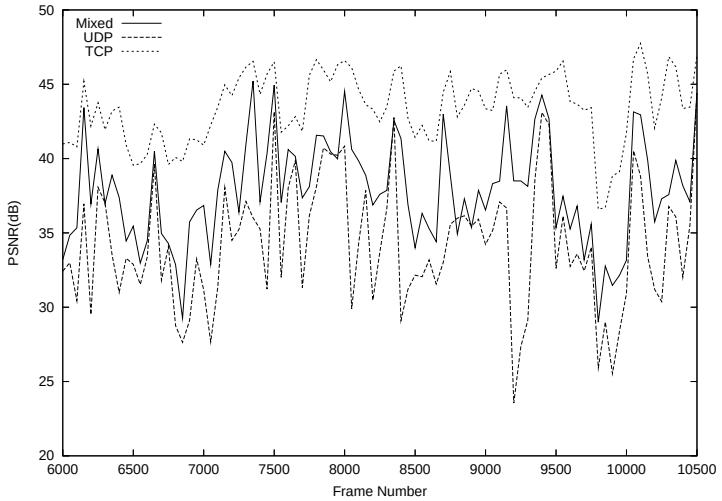
Figure 5.9: Video quality of the Santana clip with congestion and packet loss



(a) 5% packet loss with congestion

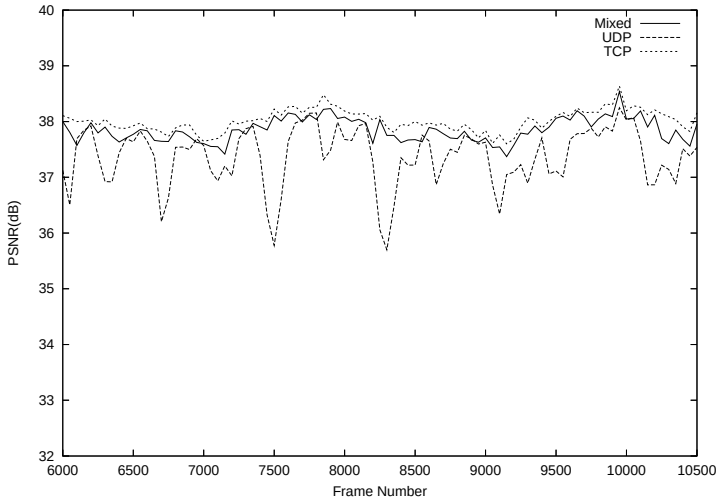


(b) 15% packet loss with congestion

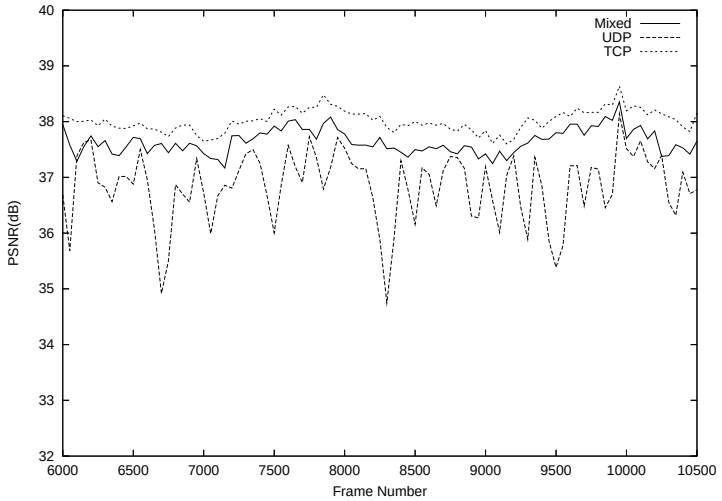


(c) 25% packet loss with congestion

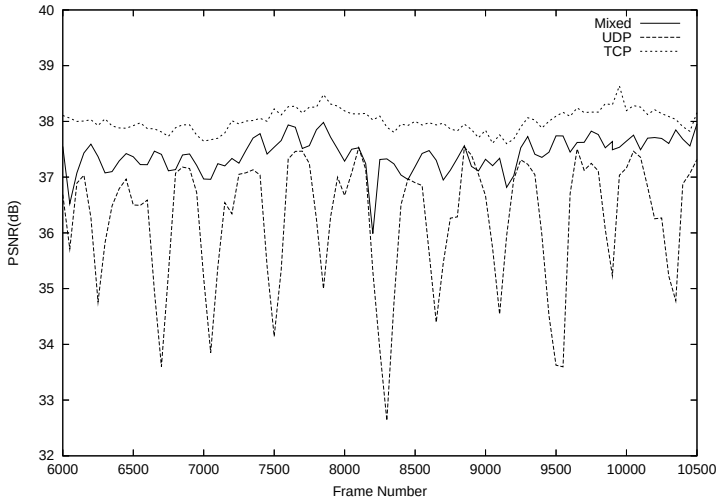
Figure 5.10: Video quality of the "The Matrix" clip with congestion and packet loss



(a) 5% packet loss with congestion

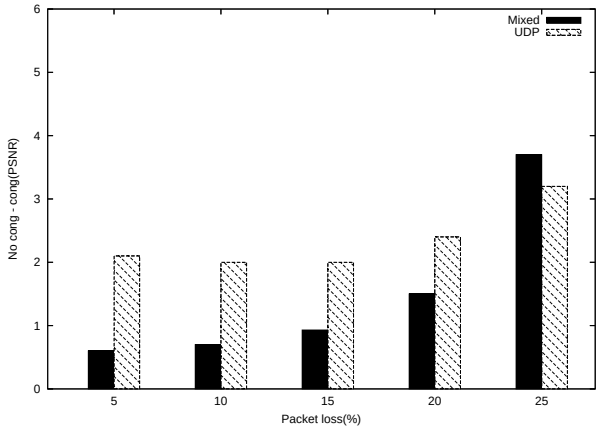


(b) 15% packet loss with congestion

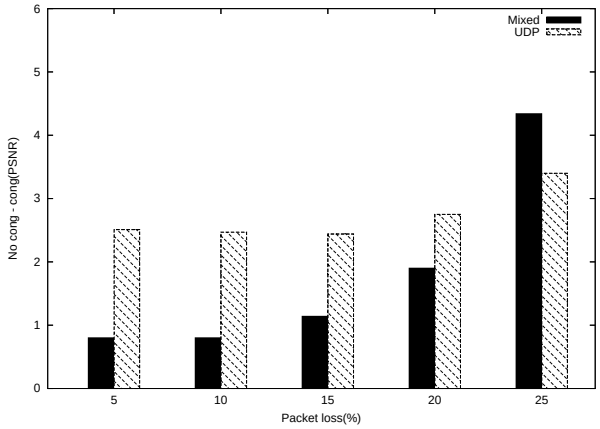


(c) 25% packet loss with congestion

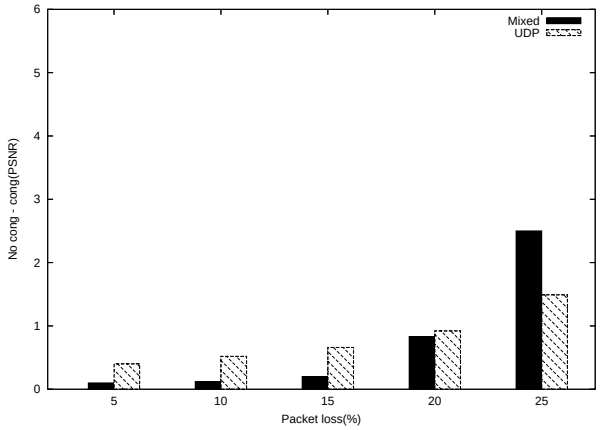
Figure 5.11: Video quality of the Bridge clip with congestion and packet loss



(a) Difference in delivered quality for the Santana video clip



(b) Difference in delivered quality for the "The Matrix" video clip



(c) Difference in delivered quality for the Bridge video

Figure 5.12: Overall difference in quality caused by congestion

5.4 Frame Rate

In the previous section, the video quality was examined in terms of the PSNR. The PSNR indicates the degradation between the original video frame and the transmitted frame but it does not give any information about the video smoothness. The frame rate represents how smooth the video plays [56]. A higher frame rate leads to a smoother video depending on the contents of the video. A video with a lot of action requires a higher frame rate than a low action video clip to play out smoothly. For instance, a high action video clip might require 24 to 30 frames per second to play smoothly while a low action clip might play smoothly at 20 frames per second. Since the smoothness of the video depends on frame rate, the smoothness will be affected if the player cannot play a particular frame at the scheduled time due to packet loss.

In this section, the video quality will be examined in terms of the frame rate. To evaluate the frame rates, an experiment with the same characteristics as described in Section 5.2 was conducted. The frame rate is calculated by dividing the number of correctly decoded frames by the duration of the clip.

Table 5.2 shows results obtained. The frame rate values are averages of ten runs and do not incorporate any congestion. The results show that the frame rate is generally affected by packet loss because the frame rate has dropped from the original 25 frames per second. It can also be observed that the "The Matrix" video clip with alternating action is more affected by packet losses recording the lowest frame rate of 15.13 frames per second at 25% packet loss.

Table 5.2: Frame rates at various packet loss rates

	Santana		**The Matrix**		**Bridge**	
	Mixed	**UDP**	**Mixed**	**UDP**	**Mixed**	**UDP**
5%	23.87	23.70	23.6	21.4	24.94	24.72
10%	23.49	22.46	21.36	18.24	24.62	24.03
15%	23.43	21.72	21.03	17.67	24.57	23.92
20%	22.6	19.20	20.30	17.02	24.11	23.44
25%	21.58	17.72	19.7	15.13	23.79	22.97

Another observation is that mixed streaming generally has a higher frame rate than UDP at all error rates and for all action levels. This suggests that mixed delivers a smoother video than UDP, indicating that the retransmission of I-frames in mixed has a positive impact on the frame rate. The quality difference between UDP-delivered and mixed-delivered clips increases as the error rate increases. This means that the video delivered by UDP will be choppy when the packet loss rate is high because it will not have all the frames to playout smoothly. On the other hand, the mixed-delivered video will have more data to playout than UDP.

The results presented in this section signify that mixed streaming can be used to deliver a smoother video, instead of UDP streaming, for high and alternating action clips in high packet loss environments. For low action videos, the choice of streaming with mixed or UDP has little difference in smoothness.

5.5 Summary

This chapter evaluated the quality of the delivered video using the PSNR. The quality was first studied on a clean channel and then packet loss and congestion were added. The results showed that mixed and TCP generally had better quality than UDP due to the fact that the two schemes recover some or all of the important video data that is lost. This means that TCP and mixed streaming can be used in an error prone environment when the waiting time is unlimited. However using TCP in such an environment may result in a long waiting time before the clip can be played. In such cases, mixed is the best streaming alternative. The findings in this chapter also showed that using mixed streaming delivers a higher frame rate than UDP streaming.

Chapter 6

Delay

Video data can experience delay when it is transmitted over the network due to network delays, packetization delays and protocol overheads. A small waiting period also called *start-up* delay (SD) is usually introduced to fill a buffer before playback commences [57]. Reliable protocols, such as TCP, experience big delays when delivering video data in a lossy or congested environment. As a result, a big start-up delay is needed before the video file can be played. The delay in TCP is from retransmitting all lost packets and from the fact that TCP will only deliver future packets after outstanding packets in the current window of data are delivered. Another source of delay for TCP is from its use of the Additive Increase Multiplicative Decrease (AIMD) rate control. It is expected that mixed streaming will reduce the delays observed in TCP because it only retransmits the most important video frames, instead of retransmitting all lost data. Mixed streaming also eliminates the problem of previously sent packets delaying new packets (head of line blocking) by using two separate connections for streaming. Section 6.1 discusses the start-up delay required to stream a smooth video and Section 6.2 discusses the video quality delivered with limited buffering.

6.1 Required Start-Up Delay

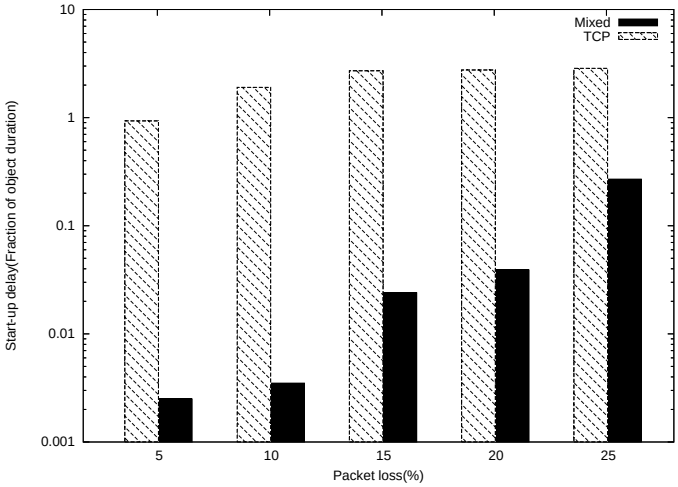
The aim of the first set of experiments in this section is to investigate the average start-up delay that is required to stream a video clip smoothly with TCP and mixed respectively. Only the mixed streaming and TCP streaming approaches are considered because UDP introduces negligible delay during transmission. Therefore, it is

not necessary to evaluate UDP delay in detail.

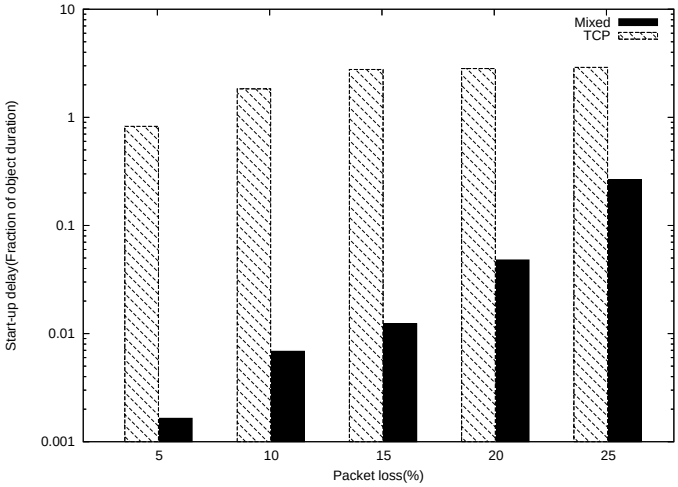
Figure 6.1 shows the start-up delay required for smooth display of the Santana, the Matrix and Bridge video clips. The experiments were conducted on the network topography shown in Figure 4.7. The bandwidth of the bottleneck link was set to 0.7 mbps and the network end-to-end propagation delay is 104 ms. For the mixed streaming approach, the TCP packets are transmitted back to back with a receiver window of 25 packets, while the UDP packets are sent as required. For the TCP streaming technique, video packets are sent as fast as possible. To calculate the required start up delay, a video clip is transmitted over a network link with a specified packet error rate. An average delay of all the frames that miss their playback deadlines is obtained. Each plot line represents ten runs. The y-axis is plotted using a log scale in order to cover the wide range of the required start-up delay times.

The experiment is first conducted on a network without packet loss in order to determine the baseline start-up delay. The characteristics of the video sequences used are shown in Table 4.2. It was observed that there was a very small start-up delay required even when there was no packet loss and the bandwidth was available. The delay could not be eliminated even when the bandwidth of the bottleneck was increased to 1.5 mbps. This small start-up delay is not indicated in the graph because it is negligible. For instance, for a video clip of 12 minutes the start-up delay with no packet loss would be 0.13 seconds.

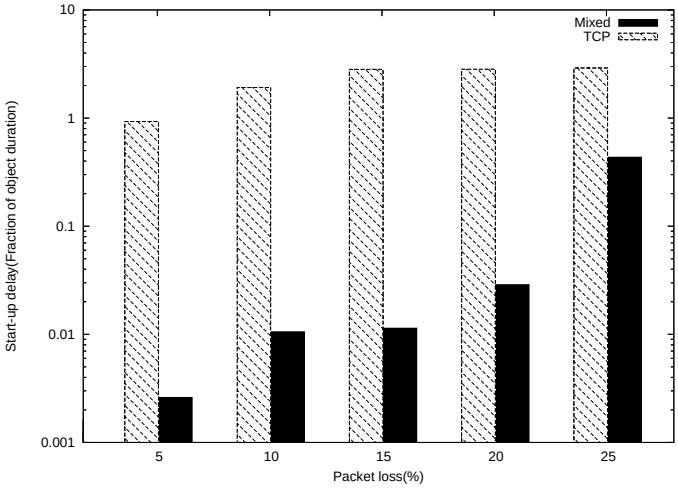
When packet loss is present, TCP requires the largest start-up delay. TCP requires a start-up delay that is more than the duration of the clip when the packet loss rate is greater than 5%. This behavior is similar for all the three video clips with different activity levels. The reason for this is because of TCP's error recovery algorithm that retransmits lost packets and attempts to deliver packets in order. TCP streams the media file sequentially and will not deliver future packets until lost packets have been retransmitted. Consequently, successive packets will be delayed and will result in high start-up delay. In mixed, the effect of TCP's error control algorithm is reduced because it is only limited to I-frames which account for a small



(a) Santana



(b) The Matrix



(c) Bridge

Figure 6.1: Required start-up delay at various packet loss rates

number of frames. The remainder of the frames are not affected because they are streamed with UDP.

The results from the experiment suggest that the mixed streaming approach has an advantage over TCP streaming in terms of the start-up delay. It is able to reduce the start-up delay between 1 and 2 orders of magnitude in high loss environments. For instance, in Figure 6.1(b) a client streaming a 10 minute clip with TCP at 5% packet loss will have to wait for more than 10 minutes to view a smooth video. For the mixed streaming approach, they will have to only wait for about 30 seconds. This suggests that the mixed streaming approach could be useful in a wireless environment with high losses.

Krasic *et al.* [46] proposed reducing the data transmitted by dropping packets when the network network is congested. Sending less data reduces or even stops the congestion there by reducing the number of lost packets and the required start-up delay. This approach is unlikely to reduce the start-up delay on a wireless network, where packet losses are mostly caused by channel conditions. One of the questions that remain is how the mixed approach will perform in regard to the delivered quality when the start-up delay is limited. This is discussed in the next section.

An important issue associated with the start-up delay is the buffer space, since packets have to be stored before playback. Table 6.1 shows the estimation for the buffer size needed to transmit a video file using TCP and mixed respectively. The estimation of the buffer size is obtained by multiplying the required start-up delay by streaming rate. The number of frames is calculated by dividing the estimated required buffer size by the average frame size of the video file.

The results show that as the packet loss rate is increased, TCP needs a much larger buffer size than mixed to accommodate delayed packets. For instance, when the packet loss rate is over 15%, TCP needs the buffer size that is the same size as the video file itself. This also means that at very high packet loss rates it takes longer to deliver the clip via TCP than the duration of the clip. The results also imply that mixed might be useful for streaming video to small devices which have low storage capabilities.

Table 6.1: Estimated buffer space and corresponding number of frames at various packet loss rates

	TCP			Mixed		
	Buffer	**% of File**	**Frames**	**Buffer**	**% of File**	**Frames**
0%	0.024 Mb	0.1	25	0.029 Mb	0.15	30
5%	10.8 Mb	54	11335	0.169 Mb	0.85	177
10%	14.2 Mb	71	14904	0.460 Mb	2	428
15%	18.9 Mb	100	19858	0.508 Mb	2.56	533
20%	18.9 Mb	100	19858	0.702 Mb	3.54	736
25%	18.9 Mb	100	19858	4.638 Mb	23.4	4859

6.2 Video Quality with Limited Start-Up Delay

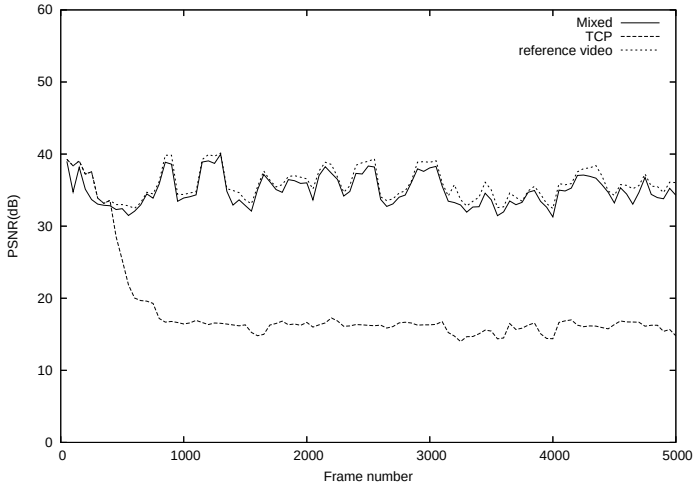
The next set of experiments examine the video quality delivered by TCP streaming and mixed streaming when the start-up delay is limited. In Section 6.2.1, the video quality is evaluated in the presence of packet loss. The delivered quality is then evaluated in the presence of packet loss and congestion in Section 6.2.2.

6.2.1 Delivered Quality with Packet Loss

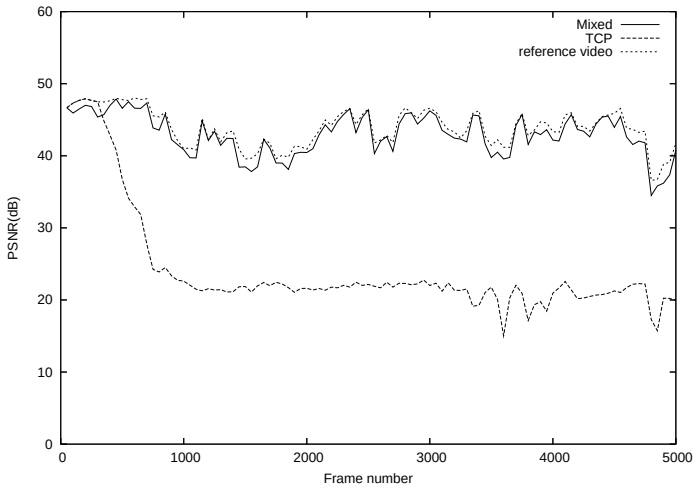
Prior to presenting the results, the parameters used in the experiments are given. The simulations are conducted on a network topology indicated in Figure 4.7 with a 0.7 mbps bottleneck bandwidth and a network propagation delay of 104 ms. The mixed approach delivers the video in the manner specified in Section 6.1. The buffering delay is set to 30 seconds. Thirty seconds buffering time is selected because of its use in popular media players. For instance, it is the default buffering time in the RealNetworks Media Player[1] and it is also recommended for the Windows Media Player in times of network difficulty[2].

Figures 6.2, 6.3 and 6.4 show the video quality delivered by mixed streaming and TCP respectively for various video clips when the start-up delay is limited. The PSNR values shown in the graphs are obtained after transmitting the video over the simulated network link with a fixed data loss rate and fixed start-up delay.

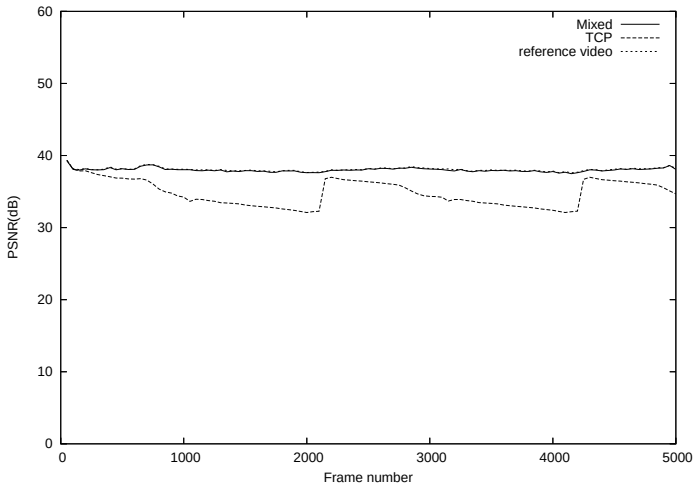
[1] http://home.real.com/product/help/rp10v8_ts/en/Pref_Playback_Settings.htm
[2] http://support.microsoft.com/kb/843509#16



(a) Santana video clip at 5% packet loss

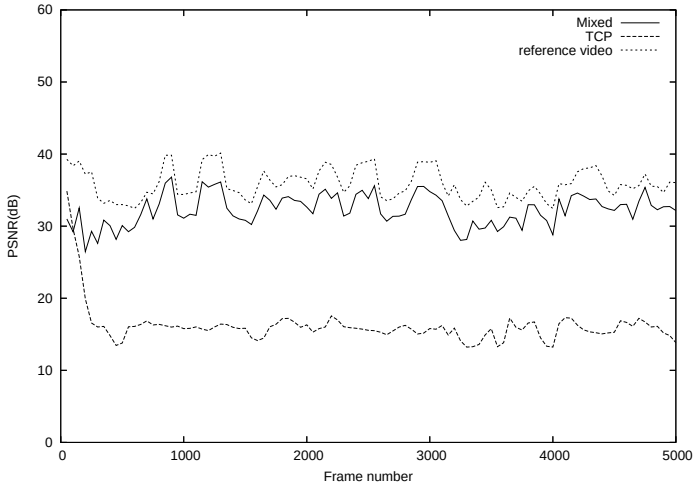


(b) "The Matrix" video clip at 5% packet loss

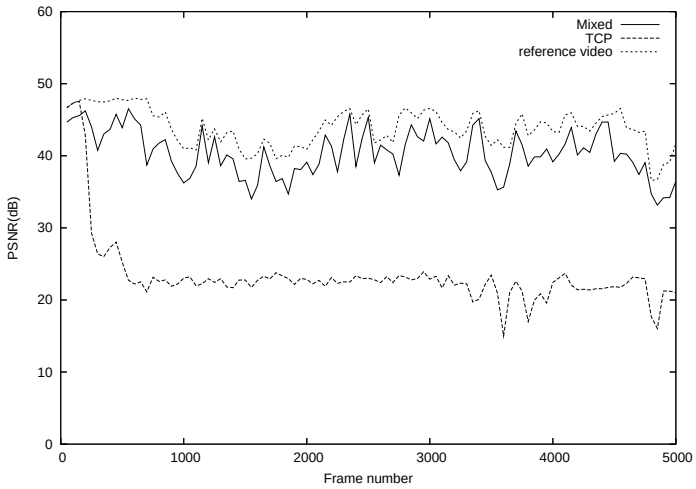


(c) Bridge video clip at 5% packet loss

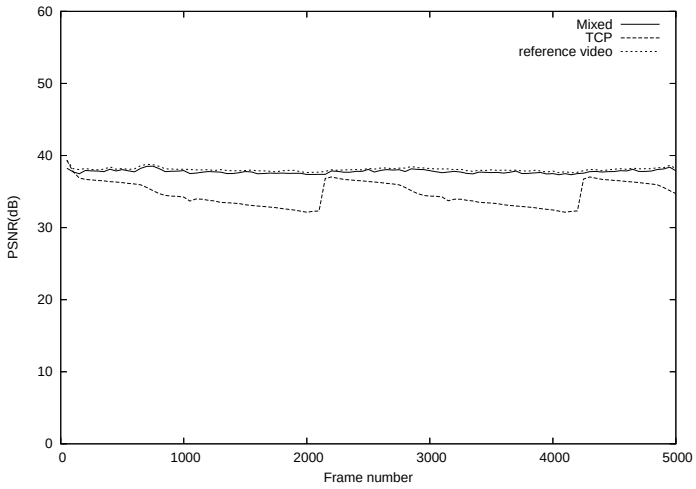
Figure 6.2: Video quality with a limited start-up delay (SD = 30 sec)



(a) Santana video clip at 15% packet loss

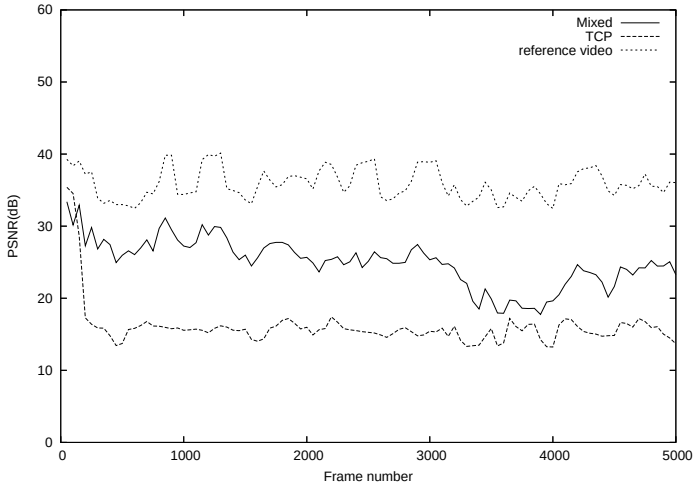


(b) "The Matrix" video clip at 15% packet loss

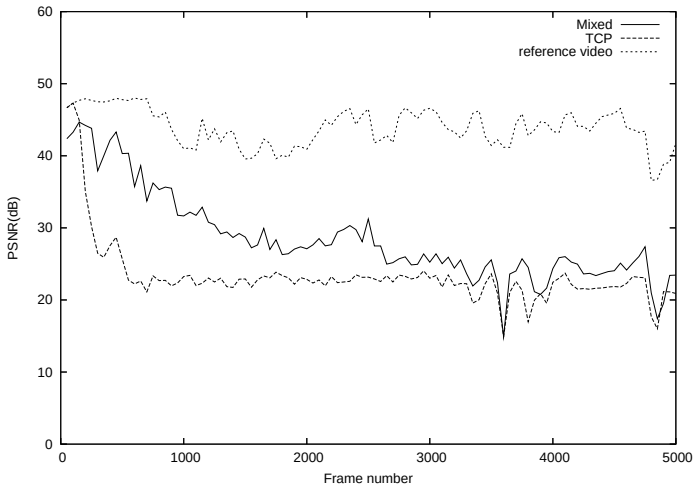


(c) Bridge video clip at 15% packet loss

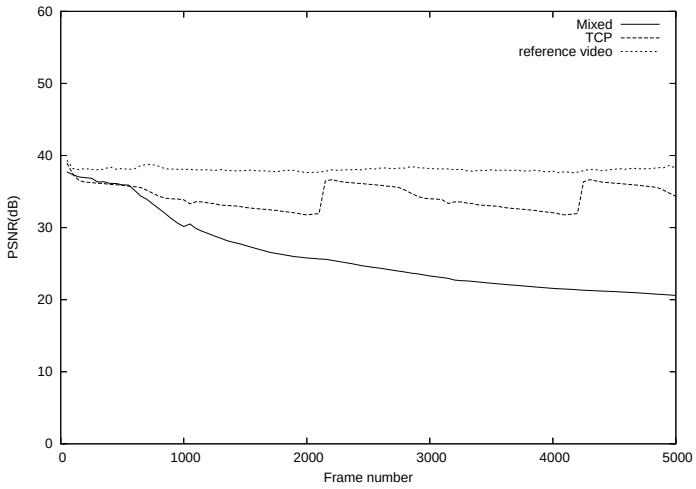
Figure 6.3: Video quality with a limited start-up delay (SD = 30 sec)



(a) Santana video clip at 25% packet loss



(b) "The Matrix" video clip at 25% packet loss



(c) Bridge video clip at 25% packet loss

Figure 6.4: Video quality with a limited start-up delay (SD = 30 sec)

Each plot point is an average of 50 frames. No congestion is introduced into the experiments. Each plotted line in the graph is an average of 10 independent runs.

The results for high, alternating and low action video clips in Figures 6.2 and 6.3 respectively show a similar behavior. For the initial part of the video, TCP streaming delivers the highest quality. As the streaming continues however, the video quality delivered by TCP degrades by more than 10 dB on average. For the mixed streaming approach, it can be observed that at the very beginning of the clip, the quality is less than that delivered by TCP, but remains stable for the remainder of the streaming session.

The reason for the behavior exhibited by TCP streaming is that it is unable to keep up with retransmissions in a timely manner during the latter parts of the clip. Due to TCP's ordered delivery, it will not deliver future data until the data in the current transmission window has been transmitted. This adds additional delay to successive TCP packets causing them to miss their playout deadline. Even the 30 second start-up delay is not enough to mitigate these delays. Experimenting with a 50 second start-up delay also showed the same pattern for TCP. The only difference was that it took a bit longer before TCP's performance degraded. Also observe that the TCP stream in Figure 6.2(c) shows a repetitive behavior for every 2000 frames. This is because the video was created by repeating the first 2000 frames continuously to increase the duration of the clip. When a frame is lost, the decoder plays previous frame. This results in the frames having a similar pattern every 2000 frames.

Figure 6.4 shows the video quality received when the packet loss rate is 25%. Mixed shows a rapid decline in quality for all action levels. The degradation in quality is due to the fact that the data loss rate is too high for packets to be delivered successfully with the given start-up delay. A bigger start-up delay is required to improve the video quality.

Figure 6.4(c) shows an unexpected plot, in which mixed streaming has a lower PSNR than TCP. The reason for this occurrence is that mixed only delivers a few I-frames at the beginning of the streaming session. The remainder of the I-frames

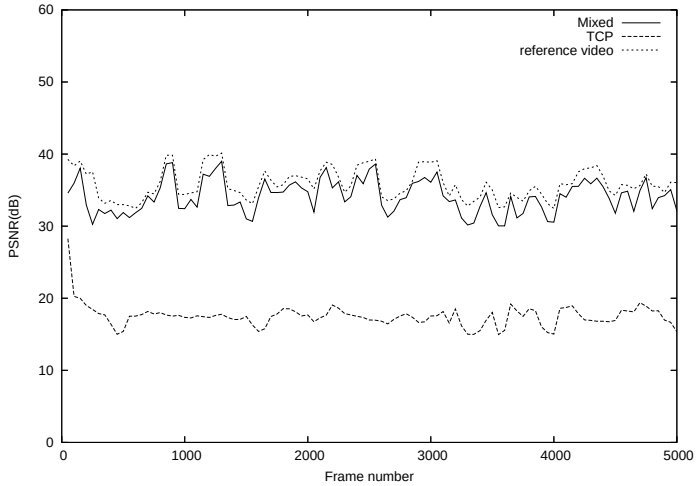
are delayed due to the high data loss. This causes some errors when decoding the video. Since some of the P and B-frames are received, the decoder uses motion vectors from B and P-frames and applies them to the last decoded frame even if it has an error. Since there are no more new I-frames to limit the propagation errors, future frames are predicted based on previous frames with errors causing the video quality to continuously degrade.

The results in this section show that the mixed streaming approach generally produced a higher PSNR than TCP on average, when the packet loss rate is around 15% or less. This means that mixed can be advantageous for delivering video clips in lossy environments, where there is limited buffering. When the packet loss becomes too high (greater than 25%), mixed also start showing some degradation. This experiment also revealed that TCP still has a higher PSNR during the initial parts of the video. This means that TCP is still attractive for streaming very short video clips because only a very small amount of data needs to be retransmitted. The results also suggest that the 30 second start-up delay is not sufficient when the packet loss rate is 25% or greater.

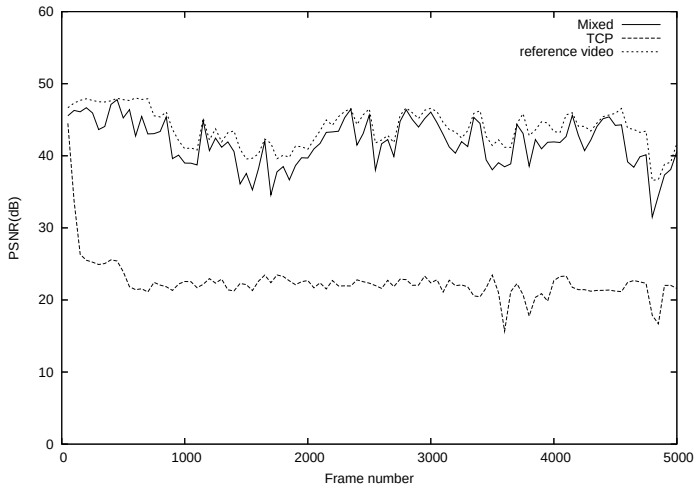
6.2.2 Delivered Quality with Congestion and Packet Loss

This section examines the delivered video quality in the presence of packet loss and congestion when the start-up delay is limited. Congestion is introduced into the simulation using an FTP source, that transmits TCP packets. Competing TCP traffic is used to provide congestion because it has been shown that the majority of data on the Internet is transmitted using TCP [31]. Therefore, videos transmitted over the Internet are expected to encounter this type of congestion. The simulation environment is kept as specified in Section 6.2.1. The FTP source starts at the beginning of the simulation and has a propagation delay of 110 ms. The mixed streaming approach is configured as specified in the previous section. The start-up delay is also set to 30 seconds.

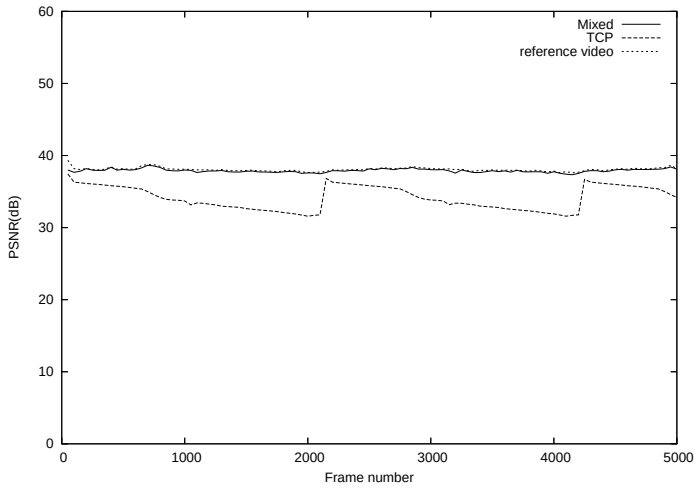
Figures 6.5, 6.6 and 6.7 show the delivered video quality when the network link is



(a) Santana video clip at 5% packet loss with congestion

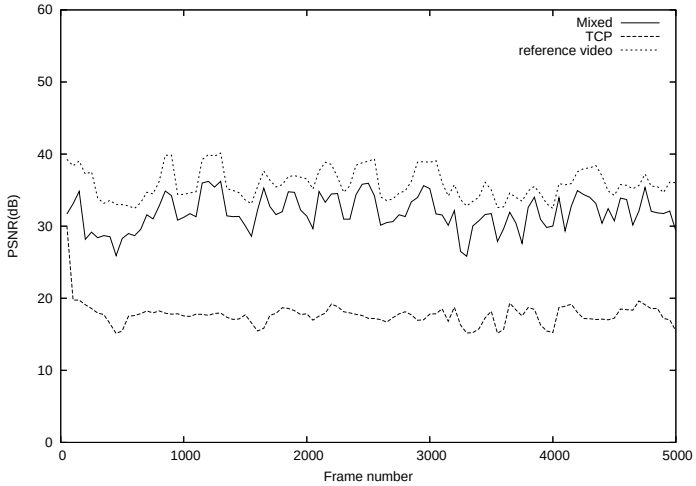


(b) "The Matrix" clip at 5% packet loss with congestion

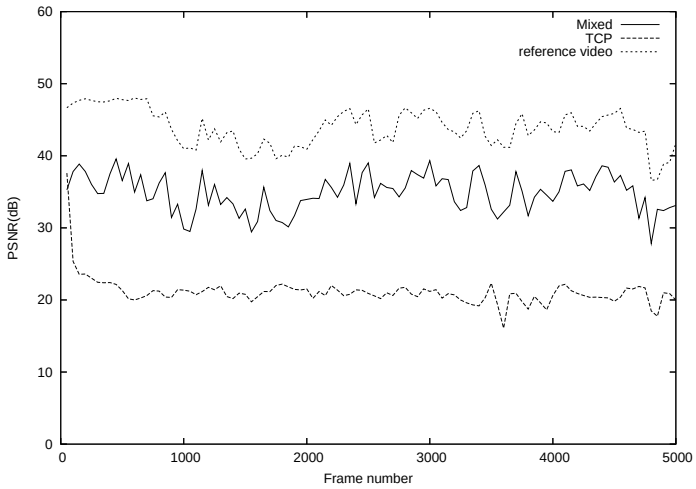


(c) Bridge video clip at 5% packet loss with congestion

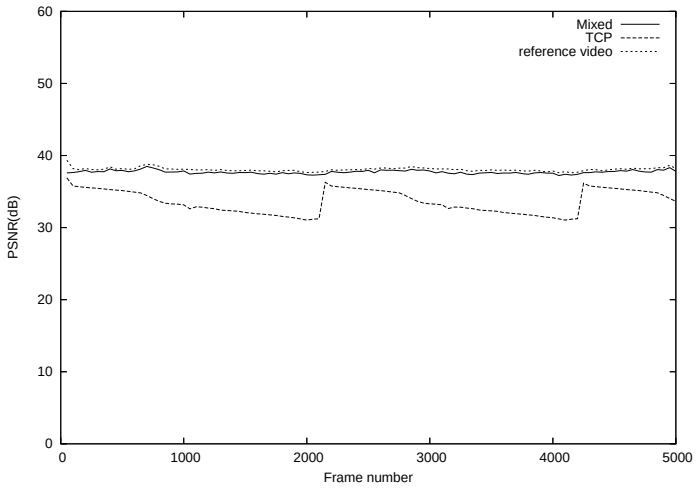
Figure 6.5: Video quality with packet loss and TCP congestion (SD = 30 sec)



(a) Santana video clip at 15% packet loss with congestion

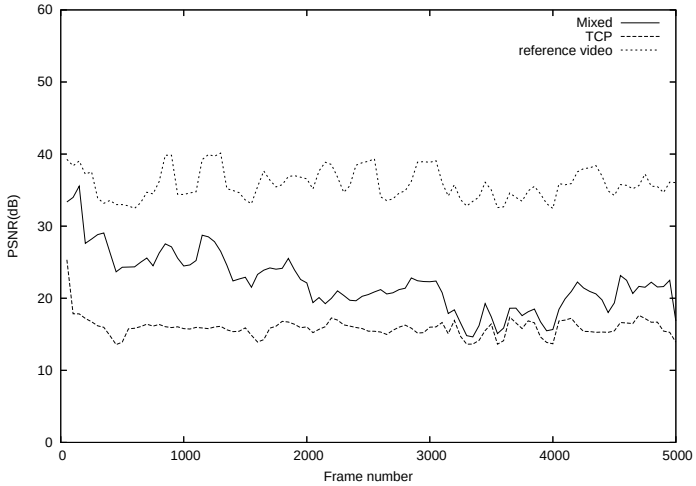


(b) "The Matrix" clip at 15% packet loss with congestion

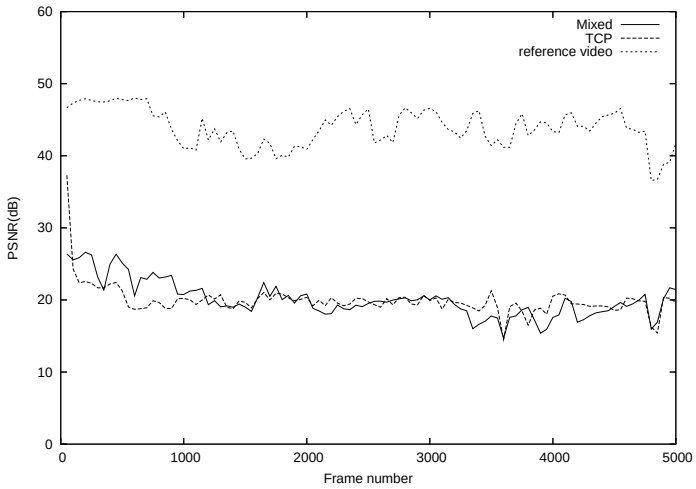


(c) Bridge video clip at 15% packet loss with congestion

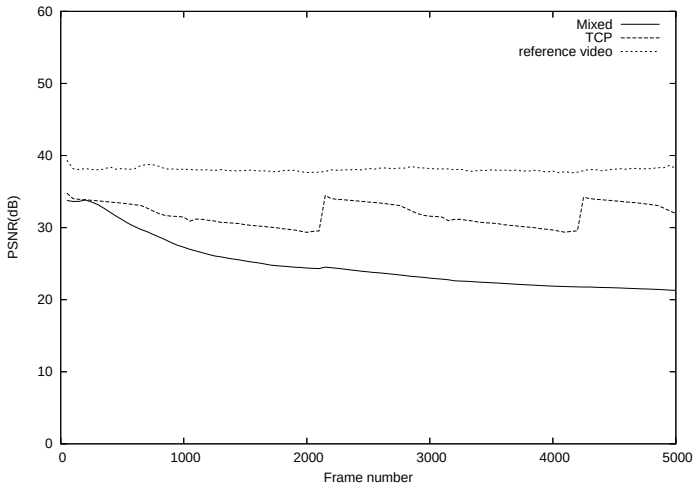
Figure 6.6: Video quality with packet loss and TCP congestion (SD = 30 sec)



(a) Santana video clip at 25% packet loss with congestion



(b) "The Matrix" clip at 25% packet loss with congestion



(c) Bridge video clip at 25% packet loss with congestion

Figure 6.7: Video quality with packet loss and TCP congestion (SD = 30 sec)

congested and there is limited buffering. At 5% and 15% of packet loss, the behavior observed is similar to that in Figure 6.2, in that mixed generally shows higher PSNR values than TCP. The difference is that TCP performs bad even in the initial parts of the videos.

The reason for the observed behavior is that when TCP starts to stream the video, the link becomes congested which leads to packet loss. This causes TCP to reduce its sending rate and begin retransmission of lost data. In addition, TCP will experience packet loss from the wireless channel which will cause additional retransmissions and delays. Due to TCP's ordered delivery, successive packets will also be late, resulting in a low PSNR.

Figure 6.7 gives the video quality when the packet loss rate is 25%. The graph indicates that the mixed streaming approach struggles to deliver a higher quality video. In some cases, it delivers video quality that is very close to that delivered by TCP.

The results show that the mixed streaming approach records a higher PSNR in the presence of packet loss and congestion when the loss rate is less than or equal to 15%. For high and alternating action video clips in Figures 6.7(a) and 6.7(b) respectively, it records PSNR values which are 10 dB more than TCP. For the low action video, PSNR differences are smaller because most of the frames are similar. The results in this experiment imply that the mixed streaming approach can be a feasible streaming alternative to TCP in wired-cum-wireless scenarios which might experience high congestion losses on the wired network and high packet loss due to channel conditions on the wireless network. The results also illustrate that when the packet loss is very high (25%) the delivered video will be bad regardless of the transmission method used.

6.3 Summary

This chapter presented results outlining the impact of the delay on the delivered video. The results show that mixed streaming requires a much smaller start-up

delay and buffer size to hide the effect of the delay than TCP. The results also indicated that the typical 30 second start-up delay used on wired networks is not enough for lossy environments like wireless networks. Experiments also revealed that in spite of the high start-up delays experienced by TCP, it is still attractive for streaming very short high quality objects.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Streaming media over the Internet is slowly becoming a major Internet application. Most of the streaming media delivered on the Internet is delivered over TCP. In spite of the increase in the amount of streaming media, streaming over wireless is still a challenging problem. This is due to channel errors that are associated with the wireless network.

The main goal of this thesis was to develop the concept of mixed streaming and study its performance over wireless networks. Another goal was to investigate if it reduces the impact of errors caused by the wireless channel on the video. To measure the performance, mixed streaming was compared with common streaming techniques namely, UDP streaming (*true streaming*) and TCP streaming (*progressive download*).

The findings on the video quality show that when there is unlimited waiting time, TCP streaming delivered better quality than mixed streaming and true streaming in the presence of wireless errors. However, having an unlimited start-up delay is not practical because it would take a long time before the client can view the video clip. As a result a limited start-up delay is usually used.

When a start-up delay (30 seconds) was introduced, mixed streaming offered the better quality than TCP in the presence of wireless errors, especially when the packet loss rate is less that 25%. When the error rate is too high, (i.e. greater than 25%) the performance gain offered by mixed streaming is reduced, as a result the quality of the video offered is similar to TCP.

For the delay experiments, the results showed that mixed streaming requires a small start-up delay in order to play the video clip smoothly while TCP requires a start up delay between 0 and 1 orders of magnitude. The results also should that it is not unusual for the client streaming with TCP to wait for more than the duration of the clip when there is significant packet loss. Another issue that makes mixed advantageous is that it requires a smaller a small start-up delay buffer than TCP to mask network delays.

These results are useful because the give insight on common streaming methods, this will help network managers to plan resource usage appropriately. For instance, an E-commerce site serving very short video clips does not require a dedicated server to improve video quality. Any available web server could be used for streaming the clips without buying new equipment. Secondly, this study demonstrates that video quality can be improved using well tested protocols such as TCP and UDP. Unlike, data from traditional Internet applications that only have one class of reliability; video data need multiple levels of reliability. As a result protocols designed for streaming applications should take reliability into account. Designing protocols which focus on data reliability could help protocol designers improve current streaming products.

7.2 Future Work

This section presents some directions for future work. Possible avenues in which this work can be extended include the following.

- *Real world experimentation* - The experiments conducted in this study were carried out on a simulated network. It would be interesting to carry out the experiments on a real wired/wireless network and see how the mixed streaming protocol performs.

- *Network-Friendliness* - Another possible area of future work would be to improve the mixed streaming protocol so that it becomes friendly to other flows

sharing the link. Currently, only the reliable component is network friendly because it uses TCP. Since the unreliable component of the mixed streaming approach uses UDP, it will be non-responsive to other flows sharing the network. One way to transform the mixed streaming approach to make it TCP-friendly is to use the Datagram Congestion Control Protocol (DCCP)[1] protocol. The DCCP is a recently developed unreliable protocol which implements a TCP-like congestion control algorithm. Since the DCCP protocol has network-friendly characteristics, replacing UDP with DCCP would make the mixed streaming approach responsive to other flows. Replacing the UDP component of mixed streaming with DCCP however might result in longer delays because DCCP will cut its sending rate in response to high packet losses.

- The scenarios considered for the experiments in this thesis only used a single mixed streaming session. It would be interesting see how multiple mixed streaming sessions would perform when they are competing with each other.

[1] http://tools.ietf.org/rfc/rfc4340.txt

References

[1] S. Acharya and B. Smith. An Experiment to Characterize Videos Stored on the Web. In *Proceedings of the ACM/SPIE Multimedia Computing and Networking*, pages 166–178, San Jose, CA, 1998.

[2] M. Allman and A. Falk. On the Effective Evaluation of TCP. *SIGCOMM Computer Communication Review*, 29(5):59–70, 1999.

[3] A. Balachandran, G. M. Voelker, P. Bahl, and P. Venkat Rangan. Characterizing User Behavior and Network Performance in a Public Wireless LAN. In *Proceedings of the 2002 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 195–205, Marina Del Rey, CA, 2002.

[4] H. Balakrishnan and R. Katz. Explicit Loss Notification and Wireless Web Performance. In *Proceedings of the IEEE GLOBECOM Global Internet Mini-Conference*, Sydney, Australia, 1998.

[5] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R.H. Katz. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. *IEEE/ACM Transactions on Networking*, 5(6):756–769, 1997.

[6] J. Bolot. End-to-End Packet Delay and Loss Behavior in the Internet. In *SIGCOMM '93: Conference on Communications Architectures, Protocols and Applications*, pages 289–298, San Francisco, CA, 1993.

[7] J. C. Bolot and T. Turletti. Adaptive Error Control for Packet Video in the Internet. In *Proceedings of International Conference on Internet Protocols*, pages 25–33, Lausanne, Switzerland, 1996.

[8] J. C. Bolot and T. Turletti. Experience with Rate Control Mechanisms for Packet Video in the Internet. *SIGCOMM Computer Communication Review*, 28(1):4–15, 1998.

[9] J. M. Boyce and R. D. Gaglianello. Packet Loss Effects on MPEG Video Sent Over the Public Internet. In *Proceedings of the 6th ACM International Conference on Multimedia*, pages 181–190, Bristol, United Kingdom, 1998.

[10] M. Chen and A. Zakhor. Rate Control for Streaming Video over Wireless. In *Proceedings of Infocom 2004*, pages 1181–1190, Hong Kong, China, 2004.

[11] C. Chien, M. B. Srivastava, R. Jain, P. Lettieri, V. Aggarwal, and R. Sternowski. Adaptive Radio for Multimedia Wireless Links. *IEEE Journal on Selected Areas in Communications*, 17(5):793–813, 1999.

[12] N. Cranley and M. Davis. Performance Evaluation of Video Streaming with Background Traffic over IEEE 802.11 WLAN Networks. In *Proceedings of the 1st ACM Workshop on Wireless Multimedia Networking and Performance Modeling*, pages 131–139, Montreal, Quebec, Canada, 2005.

[13] C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk. Gigascope: A Stream Database for Network Applications. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 647–651, San Diego, California, 2003.

[14] B. J. Dempsey, J. Liebeherr, and A. C. Weaver. On Retransmission-Based Error Control for Continuous Media Traffic in Packet-Switching Networks. *Computer Networks and ISDN Systems*, 28(5):719–736, 1996.

[15] D. A. Eckhardt and P. Steenkiste. Measurement and Analysis of the Error Characteristics of an In-Building Wireless Network. In *Proceedings of the ACM SIGCOMM '96: Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 243–254, Palo Alto, CA, 1996.

[16] D. A. Eckhardt and P. Steenkiste. A Trace-based Evaluation of Adaptive Error Correction for a Wireless Local Area Network. *Mobile Networks and Applications*, 4(4):273–287, 1999.

[17] N. Feamster and H. Balakrishnan. Packet Loss Recovery for Streaming Video. In *12th International Packet Video Workshop*, Pittsburgh, PA, 2002.

[18] FFMPEG. [Online] http://ffmpeg.sourceforge.net/index.php. Retrieved June 2004.

[19] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol (HTTP 1.1). http://www.ietf.org/rfc/rfc2616.txt, 1999.

[20] F. H. P. Fitzek and M. Reisslein. Wireless Video Streaming with TCP and Simultaneous MAC Packet Transmission (SMPT): Research Articles. *International Journal of Communication Systems*, 17(5):421–435, 2004.

[21] S. Floyd and K. Fall. Promoting the use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, 1999.

[22] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based Congestion Control for Unicast Applications. In *Proceedings of the ACM SIGCOMM '00: Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 43–56, Stockholm, Sweden, 2000.

[23] S. Floyd and V. Paxson. Difficulties in Simulating the Internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, 2001.

[24] B. Furht and M. Ilyas. *Wireless Internet Handbook: Technologies, Standards, and Applications*. CRC Press, Boca Raton, FL, 2003.

[25] D. Le Gall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, 34(4):46–58, 1991.

[26] M. Ghanbari. *Video Coding: An Introduction to Standard Codecs*. Institution of Electrical Engineers, London, Great Britain, 1999.

[27] M. Ghanbari. *Standard Codecs-Image Compression to Advanced Video Coding*. Institution of Electrical Engineers, London, Great Britain, 2003.

[28] E. N. Gilbert. Capacity of a Burst Noise Channel. *Bell Systems Technical Journal*, 39(9):1253–1265, 1960.

[29] S. Gringeri, R. Egorov, K. Shuaib, A. Lewis, and B. Basch. Robust Compression and Transmission of MPEG-4 Video. In *Proceedings of the 7th ACM International Conference on Multimedia*, pages 113–120, Orlando, FL, 1999.

[30] K. Grinnemo and A. Brunstrom. Evaluation of the QoS Offered by PRTP-ECN - A TCP-Compliant Partially Reliable Transport Protocol. In *Proceedings of the 9th International Workshop on Quality of Service*, pages 217–234, Karlsruhe, Germany, 2001.

[31] L. Guo, S. Chen, Z. Xiao, and X. Zhang. Analysis of Multimedia Workloads with Implications for Internet Streaming. In *Proceedings of the 14th International Conference on World Wide Web*, pages 519–528, Chiba, Japan, 2005.

[32] L. Guo, E. Tan, S. Chen, Z. Xiao, O. Spatscheck, and X. Zhang. Delving into Internet Streaming Media Delivery: A Quality and Resource Utilization Perspective. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, pages 217–230, Rio de Janeriro, Brazil, 2006.

[33] M. Hairuo and M. El Zarki. Broadcast/Multicast MPEG-2 Video over Broadband Fixed Wireless Access Networks. *IEEE Network*, 12(6):80–93, 1998.

[34] R. Han and D. Messerschmitt. A Progressively Reliable Transport Protocol for Interactive Wireless Multimedia. *Multimedia Systems* , 7(2):141–156, 1999.

[35] B. G. Haskell, A. Puri, and A. N. Netravali. *Digital Video: An Introduction to MPEG-2*. Kluwer Academic Publishers, Norwell, MA, 1997.

[36] O. I. Hillestad, A. Perkis, V. Genc, S. Murphy, and J. Murphy. Delivery of On-demand Video Services in Rural Areas via IEEE 802.16 Broadband Wireless Access Networks. In *Proceedings of the 2nd ACM International Workshop on Wireless Multimedia Networking and Performance Modeling*, pages 43–52, Terromolinos, Spain, 2006.

[37] F. Hou, P. Ho, and Y. Zhang. Performance Analysis of Differentiated ARQ Scheme for Video Transmission over Wireless Networks. In *Proceedings of the 1st ACM Workshop on Wireless Multimedia Networking and Performance Modeling*, pages 1–7, Montreal, Quebec, Canada, 2005.

[38] L. Huang, U. Horn, F. Hartung, and M. Kampmann. Proxy-based TCP-Friendly Streaming over Mobile Networks. In *Proceedings of the 5th ACM International Workshop on Wireless Mobile Multimedia*, pages 17–24, Atlanta, GA, 2002.

[39] Y. Iraqi and R. Boutaba. Supporting MPEG Video VBR Traffic in Wireless Networks. *International Computer Communication Journal*, 24(12):1188–1201, 2001.

[40] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, New York, NY, 1991.

[41] R. Jain and K. Ramakrishnan. Congestion Avoidance in Computer Networks with a Connectionless Network Layer. In *Proceedings of the Computer Networking Symposium*, pages 134–143, Washington, D.C, 1988.

[42] I. Joe. An Adaptive Hybrid ARQ Scheme with concatenated FEC Codes for Wireless ATM. In *Proceedings of the 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 131–138, Budapest, Hungary, 1997.

[43] J. Klaue, B. Rathke, and A. Wolisz. EvalVid - A Framework for Video Transmission and Quality Evaluation. In *Proceedings of Computer Performance Evaluation* , pages 255–272, Urbana, IL, 2003.

[44] E. Kohler, M. Handley, and S. Floyd. Designing DCCP: Congestion Control Without Reliability. In *SIGCOMM '06: Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 27–38, Pisa, Italy, 2006.

[45] A. Konrad, Ben Y. Zhao, A. D. Joseph, and Reiner Ludwig. A Markov-Based Channel Model Algorithm for Wireless Networks. *Wireless Networks*, 9(3):189–199, 2003.

[46] C. Krasic, J. Walpole, and W. Feng. Quality-adaptive Media Streaming by Priority Drop. In *Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 112–121, Monterey, CA, 2003.

[47] T. Kuang and C. L. Williamson. Hierarchical Analysis of RealMedia Streaming Traffic on an IEEE 802.11b Wireless LAN. *Computer Communications*, 27(6):538–548, 2004.

[48] T. Kuang, F. Xiao, and C. Williamson. Diagnosing Wireless TCP Performance Problems: A Case Study. In *Proceedings of SCS Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, Montreal, Canada, 2003.

[49] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison Wesley, Boston, MA, 2001.

[50] L. Larzon, M. Degermark, and S. Pink. Efficient use of Wireless Bandwidth for Multimedia Applications. In *Proceedings of the IEEE International Conference of Communications*, pages 187–193, San Diego, CA, 1999.

[51] M. Li, M. Claypool, R. Kinicki, and J. Nichols. Characteristics of Streaming Media Stored on the Web. *ACM Transactions on Internet Technology*, 5(4):601–626, 2005.

[52] H. Liu and M. El Zarki. Adaptive Source Rate Control for Real-time Wireless Video Transmission. *Mobile Networks and Applications*, 3(1):49–60, 1998.

[53] Y. Liu and M. Claypool. Using Redundancy to Repair Video Damaged by Network Data Loss. In *Proceedings of IS&T/SPIE/ACM Multimedia Computing and Networking* , pages 73–84, San Jose, CA, 2000.

[54] D. Loguinov and H. Radha. Measurement Study of Low-bitrate Internet Video Streaming. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 281–293, New York, NY, 2001.

[55] R. Ludwig, A. Konrad, and A. D. Joseph. Optimizing the End-to-End Performance of Reliable Flows over Wireless Links. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 113–119, Seattle, WA, 1999.

[56] S. Mack. *Streaming Media Bible*. Hungry Minds, New York, NY, 2002.

[57] A. Mahanti. *Scalable Reliable On-Demand Media Streaming Protocols*. PhD thesis, University of Saskatchewan, 2004.

[58] A. Mahanti, D. L. Eager, M. K. Vernon, and D. J. Sundaram-Stukel. Scalable On-Demand Media Streaming with Packet Loss Recovery. *IEEE/ACM Transactions on Networking*, 11(2):195–209, 2003.

[59] R. Mantiuk, G. Krawczyk, K. Myszkowski, and H. Seidel. Perception-Motivated High Dynamic Range Video Encoding. *ACM Transactions on Graphics*, 23(3):733–741, 2004.

[60] S. McCanne and V. Jacobson. Vic: A Flexible Framework for Packet Video. In *Proceedings of the 3rd ACM International Conference on Multimedia*, pages 511–522, San Francisco, CA, 1995.

[61] J. McDougall and S. Miller. Sensitivity of Wireless Network Simulations to a Two-State Markov Model Channel Approximation. In *Proceedings of the IEEE Global Telecommunications Conference*, pages 697–701, San Francisco, CA, 2003.

[62] B. McFarland and M. Wong. The Family Dynamics of 802.11. *Queue*, 1(3):28–38, 2003.

[63] T. Numanoglu, B. Tavli, and W. Heinzelman. The Effects of Channel Errors on Coordinated and Non-Coordinated Medium Access Control Protocols. In *Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, pages 58–65, Montreal, Canada, 2005.

[64] J. Padhye, J. Kurose, D. Towsley, and R. Koodli. A Model Based TCP Friendly Rate Control Protocol. In *Proceedings of the 9th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 137–151, Basking Ridge, NJ, 1999.

[65] C. Papadopoulos and G. Parulkar. Retransmission-based Error Control for Continuous Media Applications. In *Proceedings of the 6th International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 5–12, Zushi, Japan, 1996.

[66] C. Partridge and S. Pink. A Faster UDP. *IEEE/ACM Transactions on Networking*, 1(4):429–440, 1993.

[67] J. Postel. RFC 768: User Datagram Protocol. http://www.ietf.org/rfc/rfc3448.txt, 1980.

[68] R. Rejaie, M. Handley, and D. Estrin. Quality Adaptation for Congestion Controlled Video Playback over the Internet. In *Proceedings of the ACM SIGCOMM '99: Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 189–200, Cambridge, MA, 1999.

[69] I. Rhee. Error Control Techniques for Interactive Low-bit Rate Video Transmission over the Internet. In *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 290–301, Vancouver, BC, Canada, 1998.

[70] I. Rhee and S. Joshi. Error Recovery for Interactive Video Transmission over the Internet. *IEEE Journal on Selected Areas in Communications*, 18(6):1033–1049, 2000.

[71] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC 1889: A Transport Protocol for Real-Time Applications(RTP). http://www.ietf.org/rfc/rfc1889.txt, 1996.

[72] H. Schulzrinne, A. Rao, and R. Lanphier. RFC 2326: Real Time Streaming Protocol (RTSP). http://www.ietf.org/rfc/rfc2326.txt, 1998.

[73] A. Singh, A. Konrad, and A. D. Joseph. Performance Evaluation of UDP Lite for Cellular Video. In *Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 117–124, Port Jefferson, NY, 2001.

[74] R. Sinha and C. Papadopoulos. An Adaptive Multiple Retransmission Technique for Continuous Media Streams. In *Proceedings of the 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 16–21, Cork, Ireland, 2004.

[75] K. Sripanidkulchai, B. Maggs, and H. Zhang. An Analysis of Live Streaming Workloads on the Internet. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, pages 41–54, Taormina, Sicily, Italy, 2004.

[76] W. Stevens. *TCP/IP Illustrated*. Addison-Wesley, New York, NY, 1994.

[77] A. Tripathi and M. Claypool. Improving Multimedia Streaming with Content-Aware Video Scaling. In *Proceedings of the 6th Joint Conference on Information Science*, pages 1021–1024, Research Triangle Park, NC, 2002.

[78] J. van der Merwe, S. Sen, and C. Kalmanek. Streaming Video Traffic : Characterization and Network Impact. In *Proceedings of the 7th International Workshop on Web Content Caching and Distribution (WCW)*, Boulder, CO, 2002.

[79] B. W. Wah, X. Su, and D. Lin. A Survey of Error Concealment Schemes for Realtime Audio and Video Transmission over the Internet. In *Proceedings of the International Symposium on Multimedia Software Engineering*, pages 17–24, Taipei, Taiwan, 2000.

[80] N. Wakamiya, M. Miyabayashi, M. Murata, and H. Miyahara. MPEG-4 Video Transfer with TCP-Friendly Rate Control. In *Proceedings of the 4th IFIP/IEEE International Conference on Management of Multimedia Networks and Services*, pages 29–42, Chicago, IL, 2001.

[81] B. Wang, J. Kurose, P. Shenoy, and D. Towsley. Multimedia Streaming via TCP: An Analytic Performance Study. In *Proceedings of the 12th Annual ACM International Conference on Multimedia*, pages 908–915, New York, NY, 2004.

[82] J. Wang, A. Tang, and S. H. Low. Maximum and Asymptotic UDP Throughput under CHOKe. In *Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 82–90, San Diego, CA, 2003.

[83] R. Wang, M. Valla, M.Y. Sanadidi, and M. Gerla. Adaptive Bandwidth Share Estimation in TCP Westwood. In *Proceedings of the IEEE Global Telecommunications Conference*, pages 2604–2608, Taipei, Taiwan, 2002.

[84] Y. Wang and Q. Zhu. Error Control and Concealment for Video Communication: A Review. *Proceedings of the IEEE*, 86(5):974–997, 1998.

[85] J. Watkinson. *MPEG Handbook: MPEG-1, MPEG-2, MPEG-4*. Focal Press, Oxford, Great Britain, 2001.

[86] S. Wolf and M. Pinson. Video Quality Measurement Techniques. Technical Report TR-02-392, U.S. Department of Commerce, NTIA, 2002.

[87] D. Wu, T. Hou, W. Zhu, Y. Q. Zhang, and J. Peha. Streaming Video over the Internet: Approaches and Directions. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):282–300, 2001.

[88] D. Wu, Y. T. Hou, and Y. Zhang. Transporting Real Time Video over the Internet: Challenges and Approaches. *IEEE*, 88(12):1855–187, 2000.

[89] H. Wu and P. Chuang. Dynamic QOS address for Multimedia Ad Hoc Wireless Networks. *Mobile Networks and Applications*, 6(4):377–384, 2001.

[90] B. Yan and K. W. Ng. Techniques for the Transport of MPEG-4 Video over Wireless Networks. In *Proceedings of the 1st International Conference on Information Technology & Applications*, pages 235–240, Bathurst, Australia, 2002.

[91] J. Zhao, B. Li, C. Kok, and I. Ahmad. MPEG-4 Video Transmission over Wireless Networks: a Link Level Performance Study. *Wireless Networks*, 10(2):133–146, 2004.