TECHNIQUES FOR IMPROVING MULTIMEDIA APPLICATIONS IN HETEROGENEOUS NETWORKS

by

AHMED ABD EL AL

A dissertation submitted to the Graduate Faculty in Engineering
in partial fulfillment of the requirements for the degree of Doctor of Philosophy,
The City University of New York

2005

UMI Number: 3187423

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirements for the degree of Doctor of Philosophy.

9/1/2005	Prof. Tarek Saadawi
Date	Chair of Examining Committee
9/1/2005	Dean. Mumtaz Kassir
Date	Executive Officer

Prof. Myung Lee (EE Dept. CCNY,CUNY)

Prof. Umit Uyar (EE Dept. CCNY,CUNY)

Prof. Kaliappa Ravindran (CS Dept. CCNY,CUNY)

Prof. Mohamed Zahran (EE Dept. CCNY,CUNY)

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

TECHNIQUES FOR IMPROVING MULTIMEDIA APPLICATIONS IN HETEROGENEOUS NETWORKS

by

Ahmed Abd El Al

Advisor: Professor Tarek Saadawi

Universal access to multimedia information is now the principal motivation behind the design of next-generation communication networks. However, there are many challenges that need to be solved before transporting high quality multimedia in heterogeneous packet-based networks becomes a reality.

Low channel bandwidth is common in many wireless networks as well as on many Internet paths today. We present the design and evaluation of a transport layer solution for aggregating the bandwidth of multiple channels to form one logical wide channel for the application. The mechanism is scalable with the number of transmission channels. Also it is robust to the asymmetric channel characteristics as well as channel failures.

The high error rate and the rapidly changing characteristics of wireless channels pose a challenge for the transport of compressed video in mobile wireless networks. In motion compensated coding, errors due to packet losses are propagated from reference frames to dependant frames causing lasting visual effects. In addition, the bounded playout delay for interactive video limits the effectiveness of retransmission-based error control. We present a mechanism that combines retransmission-based error control with path diversity

in wireless networks, to provide different levels of protection to packets according to their importance to the reconstructed video quality. The new multi-path retransmission mechanism maintains the video quality under different loss rates and mobility speeds, with less overhead compared to error control techniques that depend on reference frame updates.

Although Additive-Increase Multiplicative-Decrease (AIMD) based transport protocols, such as Transmission Control Protocol (TCP), are the dominant transport layer protocols in current Internet, streaming video over such protocols is a challenging problem. As the available bandwidth to the connection changes significantly over medium and long time scales, it is desirable that the application adapts the video quality as a function of the available bandwidth. We present and evaluate adaptation strategies for real-time video, where we switch among several versions of the coded video to match the available network bandwidth accurately and meet client delay constraints.

Acknowledgments

When I sit back and think about the number of people who influenced me and helped me complete this dissertation, since I started my graduate education at City University of New York, I am overwhelmed!

There is no doubt that this would have been impossible without their help. I hope that I can acknowledge everyone who helped me through this difficult yet rewarding process.

First and foremost, I would like to thank my advisor, Professor Tarek Saadawi, for taking me on as a student, even though he knew little about me at the time. It was an extraordinary piece of good fortune that led to my becoming his student. He has been an ideal advisor in every respect, both in terms of technical advice on my research and in terms of professional advice. My choice of career path has been greatly influenced by Prof. Saadawi and I hope that I can live up to his high standards.

I am also thankful to Professors Myung Lee, Umit Uyar, Kaliappa Ravindran and Mohamed Zahran for serving on my dissertation committee. I have benefited greatly from their valuable comments on both research and writing. Prof. Lee taught one of the most interesting courses that I took, on Wireless Communications, which broadened my outlook and motivated me to work on wireless networking research. I had a great time working on the STARs project under Prof. Uyar's supervision and learnt a lot from my extended association with STARs.

I want to thank Dr. Kenneth Young for his efforts as the program manager for the Army Research Laboratory (ARL) Collaborative Technology Alliance in Communications and Networks.

The media delivery architecture group at the IBM T. J. Watson Research Center, Lisa Amini, Chitra Venkatramani and Olivier Verscheure, provided me numerous technical and career advice during my research internship. I want to thank all of them and looking forward to interact with them in the future.

I am grateful to Dean Mumtaz Kassir and his office for continuously providing me with advice and guidance through my graduate study.

I also appreciate the tremendous efforts of Victoria Okay. Her friendly attitude, approachability and sense of humor made it easy to get all administrative problems solved.

I owe a special debt of gratitude to my parents and family. They have, more than anyone else, been the reason I have been able to get this far. Words cannot express my gratitude to my parents, my sisters who give me their support and love from across the seas. They instilled in me the value of hard work and taught me how to overcome life's disappointments. My wife gives me her selfless support and love that make me want to excel. I am grateful to her for enriching my life.

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction and Preview

1.1 Motivation

Among the several trends in networking and communication, two stand out prominently: (i) the rapid global expansion and use of the multimedia communications and (ii) the massive growth in the use of wireless communication. The main reason for the first trend is the meteoric growth of the World Wide Web (WWW) and the large amounts of multimedia contents that one can find on it. The second trend, towards wireless connectivity, has been fueled by advances in communication hardware and technologies that have enabled large scale wireless networks.

One would therefore expect that the combination of these two growing technology trends—*multimedia access* on the one hand and *wireless access* on the other—would result in a potent combination and that wireless multimedia networks and services would dominate the marketplace. However, a survey of the commercial world shows that wireless multimedia networks and services are floundering, with a lack of widespread proliferation of such systems.

Why is this the case? Could it be that the inherent user demand for wireless multimedia systems is nonexistent? On the contrary, several user surveys show that this is not so, and that users do prefer the convenience offered by wireless multimedia access [1]. Wireless systems enable a plethora of new applications, especially for users on the move.

We believe that the problem is divided between the wired and the wireless sections of the network and the overall effect hampers the realization of widely deployed wireless multimedia systems. We briefly state the basic problems as:

• *Poor wireless networks performance:* Unfortunately, the performance of *multimedia* transport over different wireless networks is unacceptably low today. First, multimedia communications requires high bandwidth to guarantee an acceptable quality, which is still expensive to provide in wireless networks. Second, the high loss rate in wireless networks affects the perception quality of video and can lead to unacceptable viewing experience. Therefore, the degraded performance of wireless transport strongly discourages widespread use of wireless access networks, since the resulting price/performance ratio becomes too large compared to alternative wired access technologies.

• *Best effort nature of the Internet:* Multimedia transport across a heterogeneous best effort network, such as the Internet, while achieving acceptable quality at the client, is still a major challenge. As most networks today have no Quality of Service (QoS) provision, there has been a significant interest in developing adaptive multimedia transport protocols in best effort networks. However, these protocols should provide the

maximum possible throughput to the clients, while being friendly to the exiting traffic in the network.

Thus the vision of reliable and ubiquitous information access poses several challenges. In the first part of this thesis, we identify and solve problems related to improving the throughput and reliability of wireless multimedia networks. The end result is an adaptive reliable transport protocol architecture that provides significantly improved end-to-end multimedia performance in heterogeneous wireless networks.

In the second part of the thesis, we identify the challenges behind transmitting real-time video over Additive-Increase Multiplicative-Decrease (AIMD) transport protocols, including Transport Control Protocol (TCP) [2] and Stream Control Transmission Protocol (SCTP) [3], across a heterogeneous best effort network such as the Internet, while achieving acceptable perceptual quality at the client. Then we propose server adaptation mechanisms, for real-time multimedia over AIMD transport protocols.

Section 1.2 discusses the three fundamental challenges to efficient multimedia transport in heterogeneous networks that we identify and overcome in this thesis. Section 1.3 presents an overview of our contributions and describes the organization of the rest of the thesis.

1.2 Preview: Three Fundamental Challenges

In this section, we present the problems that we tackle in this thesis for improving multimedia performance in both wired and wireless networks and outline our solutions to them.

1.2.1 Challenge #1: Low Wireless Channel Bandwidth

For decades, only alphanumeric information was stored on computers. Since the early 1990s, however, a rapid drop in the cost of mass storage media and high-speed processors has opened the door to widespread use of multimedia applications in computer systems. Today, information is increasingly stored in the form of image, video, or audio files, which take up many times the amount of space required by text files. A parallel development in the field of data communications has been the shift from text-oriented user interfaces, such as FTP or Telnet, to more advanced multimedia communications infrastructures, including the World Wide Web, video conferencing, IP telephony, and multimedia email, to name just a few examples. This has meant an increase both in the amounts of data transmitted and in the sensitivity of that data to transmission delay. To meet today's requirements, networks must have extensive bandwidth available, and use the latest transport mechanisms.

Of the different kinds of multimedia contents, video sequences place the heaviest load on network infrastructures. Thanks to the advanced data compression techniques developed over the past few years, however, data speed requirements for transmitting video data have dropped significantly. Table 1.1 shows the average bandwidth requirements for various video applications. Depending on the screen content, however, the amount of bandwidth required can vary significantly. Video sequences that contain only slow or very little movement generate far less data than fast-changing images.

User Interface	Resolution	Bandwidth (1 channel, half duplex)	Bandwidth (Video Conference 4 users)
Video (MPEG-1 compressed)	352 x 288	1.15 Mbit/s	13.8 Mbit/s
Video (MPEG-2 compressed)	720 x 576	4 Mbit/s	48 Mbit/s
Video (MPEG-3 compressed) (HDTV)	1920 x 1080	20 Mbit/s	240 Mbit/s
Video (MPEG-4 compressed) (Videophone)	176 x 144	0.064 Mbit/s	0.768 Mbit/s

Table 1.1 Comparison between bandwidths requirements of different video formats [4].

To deploy multimedia applications to new types of networks, including those employing relatively low bit rates such as mobile wireless networks, all the available resources should be utilized to improve the viewer perception.

Many mobile devices are currently built with multiple wireless communication technologies and the capacity for expansion. The coverage areas for these wireless technologies overlap so as to provide uninterruptible communication. However, the support is still limited to the use of one communication channel per application data stream. The key contribution of our research is the implementation and performance evaluation of a transport layer mechanism for aggregation of the resources across multiple heterogeneous channels in a mobile environment. The proposed mechanism provides the mobile applications with a logical channel that has a bandwidth equal to the sum of bandwidths of all the available channels.

We implemented the bandwidth aggregation mechanism as an extension to Stream Control Transmission Protocol (SCTP) [3]. SCTP is a reliable, message-oriented data transport protocol that supports multiple streams within a single transport layer connection, an "association" in SCTP terminology, and hosts with multiple network interfaces (multi-homed hosts). These properties make SCTP more suitable for signaling transport, as well as for providing transport benefits to other applications requiring additional performance and reliability. SCTP features will be described in detail in Section 2.1.

SCTP support for multi-homed hosts is intended to provide communication reliability for the hosts engaged in the association. Initially, two interfaces, one at each host, are chosen to form the primary path that is used for transmission of the data units, "data chunks" in SCTP terminology. The other interfaces, which form the secondary paths, are only used for retransmission of lost data chunks or as a backup for the primary path. This means that although these paths exist, they are only utilized for retransmission or for failure recovery. The extend SCTP, referred to as Load Sharing-SCTP (LS-SCTP), utilizes the available paths for simultaneous transmission of data chunks, i.e., load sharing, while maintaining the SCTP congestion control on each path, in order to ensure fair integration with other traffic in the network. LS-SCTP aggregates the available bandwidth, taking in account the differences in the characteristics of the paths, in terms of bandwidth, latency and loss rates. In addition, LS-SCTP employs a new retransmission mechanism that accelerates the delivery of missing data to the receiver. We believe that this form of bandwidth aggregation is extremely beneficial for networks with limited bandwidth and high loss rates.

In Chapter 3, we present the details of the LS-SCTP protocol; compare it to several other schemes as well as evaluating its performance under different network conditions.

1.2.2 Challenge #2: High Wireless Bit-Error Rate

The second challenge arises due to the wireless channels high bit error rate. In such networks there is no end-to-end guaranteed Quality of Service (QoS) and packets may be discarded due to bit errors. Wireless channels provide error rates that are typically around 10^{-2}, which range from single bit errors to burst errors or even intermittent loss of the connection. The high error rates are due to multi-path fading, which characterizes mobile radio channels, while the loss of the connection can be due to the mobility in such networks. In addition, designing the wireless communication system to mitigate these effects can be complicated by the rapidly changing quality of the radio channel.

The effect of the high error rates in wireless channels can be devastating for the transport of compressed video. Video standards, such as MPEG [5] and H.263 [6], use motion-compensated coding to reduce the temporal and statistical redundancy between the video frames. Although motion-compensated coding can achieve high compression efficiency, it is not designed for transmission over lossy channels [7]. In this coding scheme the video sequence consists of two types of video frames: *intra-frames* (I-frames) and *inter-frames* (P- or B-frames). I-frame is encoded by only removing spatial redundancy present in the frame. P-frame is encoded through motion estimation using preceding I- or P-frame as a reference frame. B-frame is encoded bi-directionally using the preceding and succeeding reference frames. For each image block in an inter-frame, motion estimation finds a closely matching block within its reference frame, and generates the displacement

between the two matching blocks as a motion vector. The pixel value differences between the original inter-frame and its motion-predicted frame are encoded along with the motion vectors. This poses a severe problem, namely error propagation (or error spread), where errors due to packet loss in a reference frame propagate to all of the dependent frames leading to visual artifacts that can be long lasting and annoying [8].

Different approaches have been proposed to tackle the error propagation problem. One approach is to reduce the time between intra-coded frames, in the extreme case to a single frame. Unfortunately, I-frames typically require several times more bits than P- or B-frames. While this is acceptable for high bit-rate applications, or even necessary for broadcasting, where many receivers need to resynchronize at random times, the use of the intra-coding mode should be restricted as much as possible in low bit rate point-to-point transmission, as typical for mobile wireless networks. The widely varying error conditions in wireless channels limit the effectiveness of classic Forward Error Correction (FEC), since a worst-case design would lead to a prohibitive amount of redundancy. Closed-loop error control techniques like retransmission have been shown to be more effective than FEC and successfully applied to wireless video transmission. But for interactive video applications, the playout delay at the receiver is limited, which limits the number of admissible retransmissions [9].

We propose a mechanism to provide error resilience to interactive video applications in mobile wireless networks. The mechanism extends retransmission-based error control with redundant retransmissions on diverse paths between the sender and receiver. As different paths can have independent loss characteristics, sending multiple copies of the retransmitted packet on different paths can increase the probability that the packet get

received in less number of retransmissions. The priority for each data unit in the stream is determined by the application. Thus in the context of motion compensated coding, the application can assign higher priority for I-frames data than P- or B- frames data. Also P-frames might be assigned varying priority levels, since P-frames that are closer to the preceding I-frame are more valuable for preserving picture quality than later P-frames in the Group of Pictures (GOP). This prioritization scheme can also be applied on the macro-block basis in MPEG-4 [5], which provides the encoder with the flexibility to select the coding mode, i.e., intra or inter coding, on the macro-block level. To ensure in-time delivery of retransmitted packets, and to prevent re-transmitting expired packets, the retransmission is controlled by the packet lifetime, as well as estimate(s) of the path(s) delay.

As the interactivity of the video session can be hurt by the continuous failure of the transmission paths, which can be due to mobility, the mechanism monitors the paths Round Trip Time (*RTT*) and accordingly it switches between them seamlessly.

We implemented the proposed mechanism as a sub-layer above Real Time Protocol (RTP) [10]. We refer to this sub-layer as Multiple Path-RTP (MP-RTP). MP-RTP is responsible for: 1) Maintaining the priority level and the lifetime for each packet, as well as implementing a delay constrained retransmission. The priority level and the lifetime for each frame are specified by the application as it hands the frame for transmission, 2) Monitoring the status of the available paths, and switching between them.

In Chapter 4, we present the design of MP-RTP as well as its performance evaluation under different loss rates and mobility speeds.

1.2.3 Challenge #3: Unsuitability of AIMD Transport Protocols for Video Transport

Transmitting real-time video across a heterogeneous best effort network such as the Internet, while achieving acceptable perceptual quality at the client, is still a major challenge. There has been significant interest in developing adaptive streaming media mechanisms for best-effort networks for many reasons, (i) The cost of a non-QoS connections is low; (ii) Although some networks provide the option for a QoS connection, most networks today have no such provision; and (iii) Due to the long connection duration of a streaming session, performing admission control only at the beginning of a session, may lead to high call-rejection rates or unnecessarily poor media quality.

While the majority of traffic on the Internet today is comprised of Additive-Increase Multiplicative-Decrease (AIMD) transport protocols, such as TCP/SCTP flows, conventional wisdom holds that such protocols are unsuitable for “real-time” traffic due to its lack of throughput guarantees and insistence on reliability [11]. For these reasons, there have been many proposals for new transport protocols for the purpose of solving the video transport problem over the Internet. These protocols need to be TCP-friendly to ensure that they will not cause network collapse. However, proving a new transport protocol to be TCP-friendly can be difficult, because the dynamics of TCP congestion control is extremely complex [12]. Thus an advantage of using TCP/SCTP in transporting video is that the transport is ensured to be friendly to other flows sharing the same network. Also in many situations streaming over TCP/SCTP is unavoidable, such as

when the client machines are located behind network firewalls permitting only inbound HTTP traffic.

The TCP/SCTP flow of an application experiences rate variations for two distinct reasons -- the first being the flow's own congestion control behavior, i.e., the window-based congestion control algorithm of TCP/SCTP introduces saw-tooth fluctuation in the streaming rate, and the second being competing traffic in the network. Client-side buffers can be used for smoothing out the saw-tooth fluctuation of a TCP/SCTP flow. However, despite any amount of buffering, competing traffic can have persistent effects on the streaming rate, and consequently on the viewing quality. Also in the case of real-time video, buffering is limited by end-to-end latency limit. Thus streaming video applications must deal with persistent rate changes, before the client-side buffers are overwhelmed. The usual way is to employ quality-adaptation control, adjusting the basic quality-rate trade off of the video.

The primary design goal of quality-adaptation control mechanisms is to adapt the outgoing video stream so that, in times of network congestion, less video data is sent into the network and consequently fewer packets are lost and fewer frames are discarded. This rests on the underlying assumption that the smooth and timely play out of consecutive frames is central to a human observer's perception of video quality. Although a decrease in the video bit rate noticeably produces images of coarser resolution, it is not nearly as detrimental to the perceived video quality as inconsistent, start-stop playout. By switching between different quality levels during the stream, the mechanism makes a fundamental trade-off by increasing the video compression in an effort to preserve a consistent frame rate at the client.

Previous work in literature introduces different quality adaptation mechanisms for stored video over TCP [13][14][15], where future portions of the video stream are prefetched into client storage when the available bandwidth exceeds the consumption rate. For real-time video prefetching is not possible, and the server application transmits the video stream at the consumption rate, unless the available bandwidth is less than the consumption, at which time the stream is transmitted at the available bandwidth rate. If the prefetch buffer becomes empty, the client can partially receive the stream by reading directly from the network while incurring some loss of data. High fractions of lost data occur if the scheme fails to follow the variations in available bandwidth.

We introduce a sender-driven quality adaptation to minimize any overheads at the client. In particular, we focus on stream switching as quality adaptation using stream switching was shown to provide better viewing quality than adding/dropping layers, due to the layering overhead. The adaptive stream switching mechanism for real-time video does not require modifications to the network transport protocol at the sender or at the receiver, or need support from the network infrastructure. It detects the variations of the network bandwidth through monitoring the application buffer occupancy, and accordingly adapts the video quality to ensure that the client buffer does not underflow and that the adaptation affects the perceptual quality at the client minimally. We also show that by adapting the TCP/SCTP sender buffer according to the available bandwidth delay product, the accuracy of the adaptation can be improved significantly.

Chapter 5 presents the design and implementation details of the adaptation mechanisms and the results of performance experiments.

1.3 Structure of Thesis

The rest of this thesis is organized as follows. In Chapter 2, we discuss background material in the area of multimedia transport and reliable data delivery over heterogeneous networks and describe related work in improving multimedia transport performance.

Chapters 3 through 5 form the core of this thesis. Chapter 3 describes Load Sharing-SCTP, our extension to standard SCTP for aggregating the bandwidth of the available channels regardless of their different characteristics in terms of data rate, delay and loss rate.

In Chapter 4, we address the challenges posed by the high error rate in mobile wireless networks and the rapidly changing quality of the radio channels and their devastating effects on the transport of compressed video. After discussing these problems, we describe the details of our solutions to them, which depend on extending retransmission-based error control to provide different levels of protection to video packets through redundant retransmission on diverse paths.

In Chapter 5, we describe different multimedia adaptation strategies for real-time video streams over AIMD transport protocols. In addition, we present a sender side transport buffer adaptation mechanism that improves the adaptation accuracy. We present our simulation and testbed implementation for the adaptation strategies, and verify their performance, in terms of the network fidelity and received video quality, using real and simulated network traces.

Finally, in Chapter 6, we present a summary of our work and contributions. We also outline some directions for future research.

Chapter 2
Background and Related Work

The purpose of this chapter is to introduce the reader to related work in multimedia transport and reliable data delivery over heterogeneous networks. As several mechanisms in this thesis are based on Stream Control Transmission Protocol SCTP [3], we present, in Section 2.1, an overview of SCTP, highlighting its features and comparing it to Transmission Control Protocol, TCP [2]. In Section 2.2, we discuss conventional approaches to tackling the problems caused by limited bandwidth, concentrating on transport level solutions, and highlighting their weakness. In Section 2.3, we review different proposed techniques to tackle the effect of wireless bit-errors on video. Section 2.4 describes related work in Internet multimedia adaptation. We conclude this chapter with a summary in Section 2.5.

2.1 Stream Control Transmission Protocol Overview

For many years SS7 has been the dominant bearer of signaling traffic for telecommunication networks, but recently many proprietary solutions for transporting signaling traffic over IP have appeared. This approach promises tighter integration with Voice over IP (VoIP) solutions and ultimately the possibility of a common core network

transporting signaling and media traffic. Driven by industry interest, and general agreement on the unsuitability of either TCP or UDP, the IETF Signaling Transport (SIGTRAN) group was formed in 1999 to standardize a suitable transport protocol for signaling traffic over IP, Stream Control Transmission Protocol (SCTP) is the result of this work, and recently published as RFC 2960 [3] by the Internet Society.

SCTP is the fundamental member of a family of protocols [16] being designed by the SIGTRAN group to allow SS7 messages to be transported over an unreliable IP infrastructure. Furthermore, the enhanced capabilities of SCTP when compared with traditional Internet transport protocols such as TCP and UDP may make it attractive as a transport protocol for a wide range of traditional Internet services such as those based on HTTP. A comparison between SCTP, TCP and UDP features is included in appendix A.

SCTP is characterized in a number of current Internet drafts that discuss its capabilities and applicability to both signaling domain [17] and more general applications. Essentially, SCTP is a reliable, message-oriented data transport protocol that supports multiple streams within a single transport layer connection (an “association” in SCTP terminology) and hosts with multiple network addresses (called multi-homing in SCTP). All of these properties make SCTP more suitable for signaling transport, as well as for providing transport benefits to other applications requiring additional performance and reliability. An illustration of SCTP association is shown in Figure 2.1.

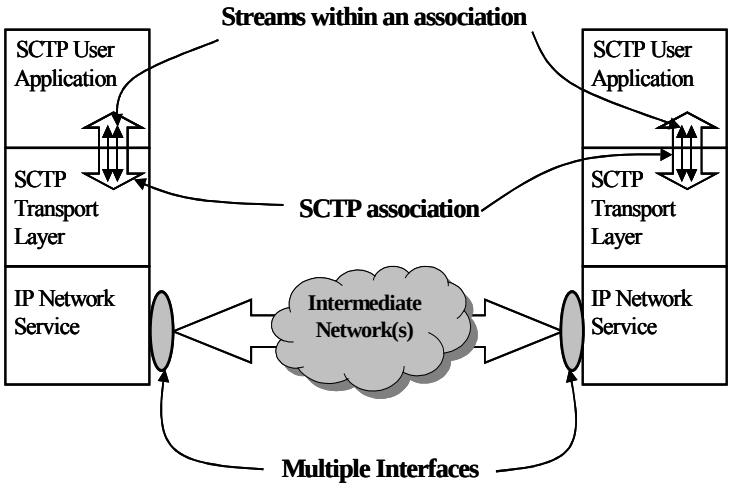


Figure 2.1 SCTP association.

2.1.1 SCTP Associations

Data transfer between two hosts in SCTP takes place in the context of an association. All transfer of data between hosts is encapsulated in the SCTP packet, shown in Figure 2.2, which contains a common header and sequence of structures called data chunks. These chunks may be either SCTP control information, such as a Selective Acknowledgement (SACK) chunk, or user data carried in a data chunk.

The common header, as shown in Figure 2.3, has source and destination port numbers to allow multiplexing of different SCTP associations at the same address, a 32-bit verification tag that guards against insertion of an out-of-date or false message into the SCTP association, and a 32-bit checksum for end-to-end error detection, which is more robust than the 16 bit checksum of TCP and UDP.

As illustrated in Figure 2.4, each chunk includes chunk type, flag field, length and value. Control chunks incorporate different flags and parameters depending on the chunk type.

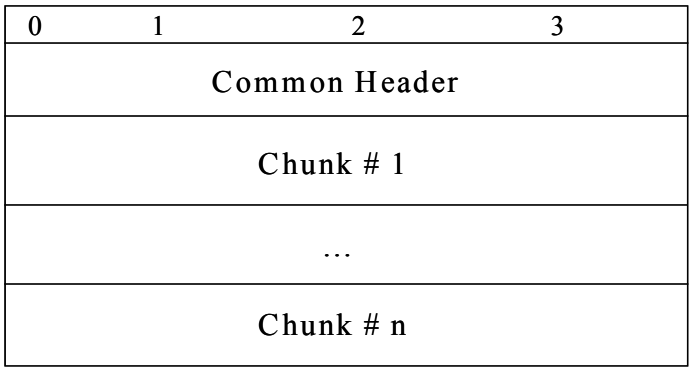


Figure 2.2 A high-level view of an SCTP packet.

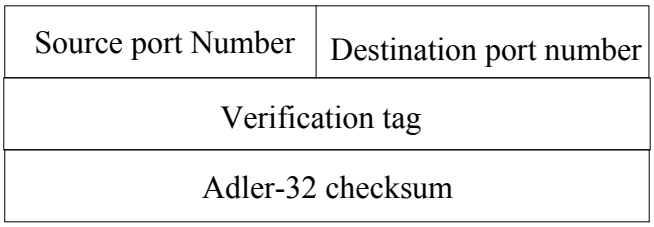


Figure 2.3 The SCTP common header.

The Data chunk is the container for all the user data transferred in SCTP. Its format is shown in Figure 2.5.

Data chunks in particular incorporate flags for control of segmentation and reassembly, and parameters for the Transmission Sequence Number (TSN), Stream ID and Stream Sequence Number (SSN), and a Payload Protocol Identifier. The Payload Protocol identifier has been included for future flexibility. Initially, values will reflect the adaptation layer protocol running over SCTP, and will match the registered port number assignments for adaptation layers. In the future, it is envisioned that the functions of protocol identification and port number multiplexing will not be as closely linked so that the port number significance is not so much overloaded as it is now.

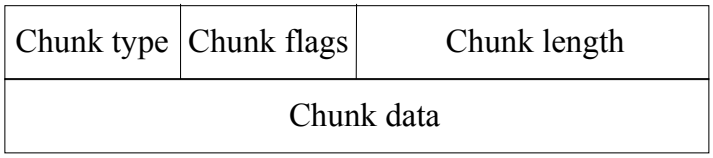


Figure 2.4 The layout of a chunk.

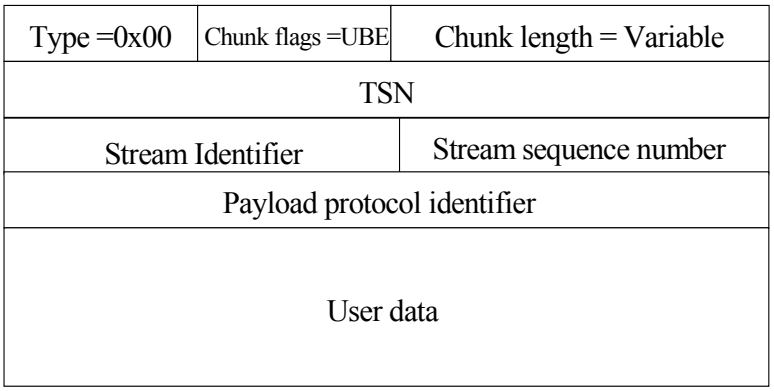


Figure 2.5 The Data chunk.

Data chunks arriving at the SCTP receiver are acknowledged by transmitting a SCTP packet with a SACK control chunk. The format of SACK chunk is illustrated in Figure 2.6.

The SCTP message format naturally allows support of bundling of multiple data and control chunks in a single message, to improve transport efficiency. Use of bundling is controllable by the application, so that bundling of initial transmission can be prohibited. Bundling will naturally occur on retransmission of DATA chunks, to further reduce any chance of congestion.

Within the association DATA chunks belong to a specific stream of the data. Normally data is ordered within a stream, although transfer of unordered data is also supported, and ordered data that has arrived out of sequence is buffered until the missing data chunks arrive.

Type=0x03	Chunk flags =0	Chunk length = Variable
Cumulative TSN acknowledgement		
Advertised receiver window credit (a_rwnd)		
Number of Gap ACK Blocks = N	Number of duplicates = X	
Gap Ack block # 1 start TSN offset	Gap Ack block # 1 end TSN offset	
.....		
Gap Ack block # N start TSN offset	Gap Ack block # N end TSN offset	
Duplicate TSN 1		
.....		
Duplicate TSN X		

Figure 2.6 The SACK chunk.

During association initialization the SCTP end points exchange the size of their receiver window (an indication of the space available in their inbound buffer) and the initial TSN of the user DATA chunks to be exchanged during the association. All DATA chunks transmitted receive a unique (monotonically increasing) TSN from a 32 bit namespace. Ordered data within a stream also receive a unique SSN from a 16 bit name space. User data is not limited in size as it may be fragmented over multiple data chunks. Individual DATA chunks may be up to 65,536 bytes in length. SCTP should ensure that each SCTP datagram including any chunks fits within one IP datagram and fragments user data into multiple DATA chunks as necessary.

2.1.2 SCTP Core Features

2.1.2.1 Multi-streaming

The name *Stream Control Transmission Protocol* is derived from the multi-streaming function provided by SCTP. This feature allows data to be partitioned into multiple streams that have the property of being delivered independently, so that message loss in

any of the streams will only affect delivery within that stream, and not in other streams (i.e., prevent head-of-line blocking).

In contrast, TCP provides a single stream of data and ensures that delivery of that stream takes place with perfect sequence preservation "strictly ordered service". While this is desirable for delivery of a file or record, it causes additional delay when message loss or sequence error occurs within the network. When this happens, TCP must delay delivery of data until the correct sequencing is restored, either by receipt of an out-of-sequence message, or by retransmission of a lost message.

For a number of applications, this characteristic of strict sequence preservation is not truly necessary. In signaling, for example, it is only necessary to maintain sequencing of messages that affect the same resource (e.g., the same call, or the same channel). Other messages are only loosely correlated and can be delivered without having to maintain overall sequence integrity, i.e., partially ordered service.

Another application having this property is delivery of web page objects, if done over a single session. It is generally not necessary to maintain sequence between the presentation of objects, and in some cases (especially some types of graphic images) it may be possible to present parts of a single object out of sequence. Eventually, the goal is to deliver all objects, however the ability to deliver objects out of sequence may result in at least better perceived performance, as parts of the web page can be displayed rather than waiting for all of the information to be received before displaying it. Figure 2.7 illustrates SCTP multi-streaming.

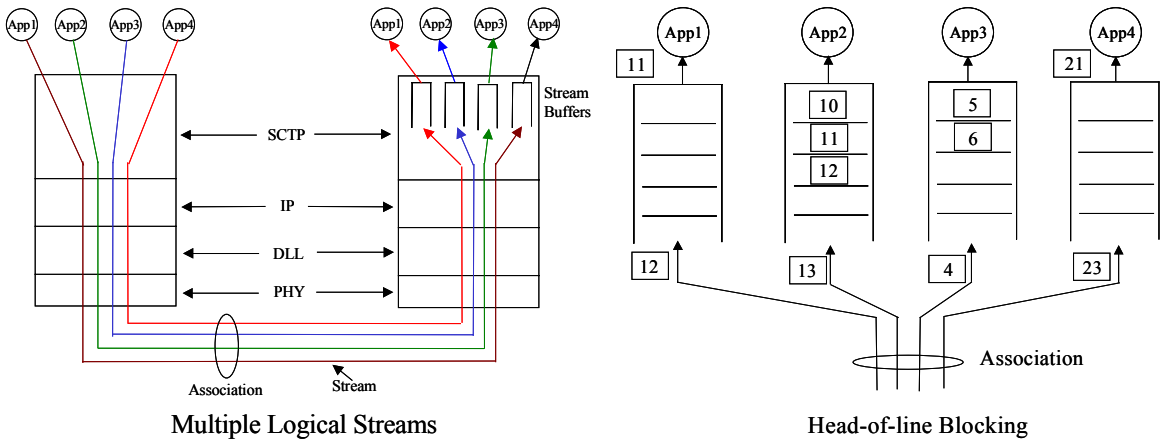


Figure 2.7 SCTP multi-streaming.

SCTP accomplishes multi-streaming by creating independence between data transmission and data delivery. In particular, each data chunk in the protocol uses two sets of sequence numbers, a Transmission Sequence Number (TSN) that governs the transmission of messages and the detection of message loss, and the Stream ID/Stream Sequence Number (SSN) pair, which is used to determine the sequence of delivery of received data.

This independence of mechanisms allows the receiver to determine immediately when a gap in the transmission sequence occurs (e.g., due to message loss), and also whether or not messages received following the gap, are within an affected stream or not. If a message is received within the affected stream, there will be a corresponding gap in the SSN, while messages from other streams will not show a gap. The receiver can therefore continue to deliver messages to the unaffected streams while buffering messages in the affected stream until retransmission occurs.

2.1.2.2 Multi-homing

The other core feature of SCTP is multi-homing, or the ability for a single SCTP endpoint to support multiple IP addresses. The benefit of multi-homing is potentially greater survivability of the session in the presence of network failures. In a conventional single-homed session, the failure of a local LAN access can isolate the end system, while failures within the core network can cause temporary unavailability of transport until the IP routing protocols can re-converge around the point of failure. Using multi-homed SCTP, redundant LANs can be used to reinforce the local access, while various options are possible in the core network to reduce the dependency of failures for different addresses. Use of addresses with different prefixes can force routing to go through different carriers, for example, while route-pinning techniques or even redundant core networks can also be used if there is control over the network architecture and protocols. Figure 2.8 illustrates SCTP multi-homing.

In its current form, multi-homing is used for redundancy purposes only. A single address is chosen as the "primary" address and is used as the destination for all data chunks during normal transmission. Retransmitted data chunks use the alternate address(es) to improve the probability of reaching the remote endpoint, while continued failure to send to the primary address ultimately results in the decision to transmit all data chunks to an alternate destination address until the primary address become reachable again.

To support multi-homing, SCTP endpoints can exchange lists of addresses during initiation of an association. Each endpoint must be able to receive messages from any of the addresses associated with the remote endpoint. A single port number is used across the entire address list at an endpoint for a specific session.

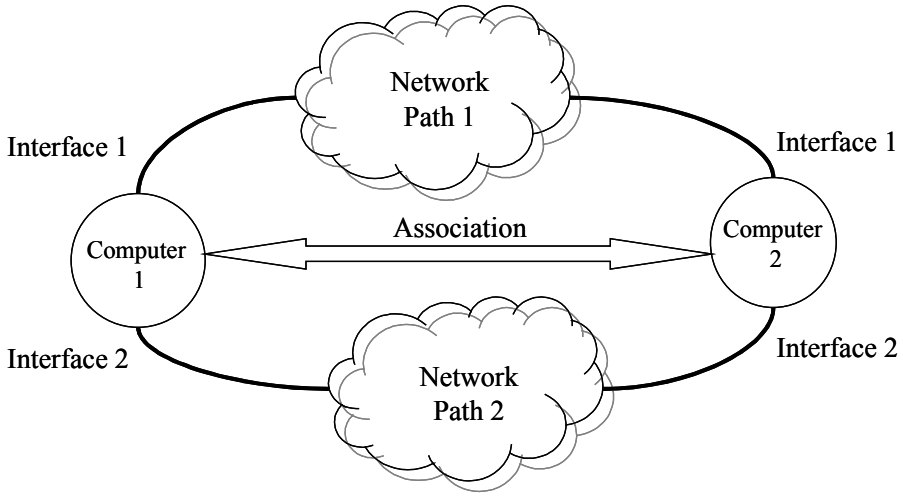


Figure 2.8 SCTP multi-homing.

In order to reduce the potential for security problems, it is required that some response messages be sent specifically to the source address in the message that caused the response. For example, when the server receives an INIT chunk from a client to initiate an SCTP association, the server always sends the response INIT ACK chunk to the source address that was in the IP header of the INIT.
For path failure detection, a count is maintained of the number of retransmissions to a particular destination address without successful acknowledgement. When this count exceeds a configured maximum, the address is declared inactive, notification is given to the application, and the SCTP begins to use an alternate address for sending data chunks. Also, Heartbeat chunks are sent periodically to all idle destinations (i.e., alternate addresses), and a counter is maintained on the number of Heartbeats sent to an idle destination without receipt of a corresponding Heartbeat ACK. When this counter exceeds a configured maximum, that destination address is also declared inactive. Heartbeats continue to be sent to inactive destination addresses until an ACK is received, at which point the address can be made active again. The rate of sending Heartbeats is

tied to the Retransmission Time Out (*RTO*) estimate plus an additional delay, to allow Heartbeat traffic to be tailored to the needs of the application.

For Endpoint failure detection, a count is maintained across all destination addresses on the number of retransmits or Heartbeats sent to the remote endpoint without successful ACK. When this count exceeds a configured maximum, the endpoint is declared unreachable, and the SCTP association is closed.

2.1.2.3 Other SCTP Features

- SCTP is a unicast protocol, and supports data exchange between exactly two endpoints, although these endpoints may be represented by multiple IP addresses. SCTP provides *reliable* transmission, detecting when data is discarded, reordered, duplicated or corrupted, and retransmitting damaged data as necessary. Newer versions of SCTP have also introduced a third variation called "Partially Reliable", which offers a service resembling UDP [18].

- SCTP is *message oriented* and supports framing of individual message boundaries. In comparison, TCP is *stream oriented* and does not preserve any implicit structure within a transmitted byte stream, which means that applications are responsible for tracking message boundaries and using the *push* mechanism to ensure that messages are transferred in reasonable time.

- SCTP allows *unordered* data delivery. By default, delivery is ordered, i.e., delivery is postponed until packets are re-ordered. If the U-bit is set in the data chunk flags, the receiver must bypass the normal ordering mechanism and deliver the packet

immediately. This can be effective method of delivering "out-of-band" data. When sending, SCTP may place an unordered packet at the head of the outbound transmission queue.

- SCTP is *rate adaptive* similar to TCP, and will scale back data transfer to the prevailing load conditions in the network. It is designed to behave cooperatively with TCP sessions attempting to use the same bandwidth.

- Resistance to Denial Of Service (DOS) and masquerading attacks, by using the cookie concept and verification tags.

2.1.3 Monitoring Data Delivery

In order to track the amount of data currently within the network the SCTP sender must associate TSNs and the byte size of the corresponding data chunks. Data chunks arriving at the SCTP receiver are acknowledged by transmitting a SCTP packet with a Selective Acknowledgements (SACK) control chunk, shown previously in Figure 2.6.

The SACK chunk carries several pieces of information.

- The Cumulative TSN Acknowledgement, which records the highest sequential TSN received.

- The Advertised Receiver Window Credit (*a_rwnd*), which reflects the current capacity of the receiver inbound buffer.

- A sequence of Gap Acknowledgement Blocks, which record any out of sequence TSNs received.

- A sequence of Duplicate TSNs, which record any TSNs for which duplicates have been recorded.

Data is not considered fully delivered until the cumulative TSN Acknowledgement point advances past its TSN. Thus the information in the Gap Acknowledgement Blocks corresponds to TCP SACK blocks. Under normal conditions SCTP uses a delayed acknowledgment scheme that sends one SACK for every second incoming packet that contains one or more new data chunks.

In order to perform congestion control, SCTP uses a number of internal variables to control the rate at which data is injected into the network. These are as follows (all are recorded in bytes):

- Receiver window size (*rwnd*): This corresponds to a sender's view of the receiver's incoming buffer space.

- Congestion control window (*cwnd*): this corresponds to the sender's view of the network conditions. The initial value of *cwnd* is less than or equal to twice the Path Maximum Transfer Unit (PMTU).

- Slow-start threshold (*ssthresh*): the sender uses this variable to distinguish between slow-start and congestion avoidance phases.

- Partial bytes acknowledged (*pba*): the sender uses this variable to limit *cwnd* adjustment to once per Round Trip Time (*RTT*) during congestion avoidance phases.

- Flight Size (*flightsize*): Some implementations use this variable as a counter for how many bytes of data are outstanding on a destination address.

Current SCTP specification always keeps the receiver window size on a per association basis, in multi-homing environment the other values must be recorded on a per destination address basis.

2.1.4 SCTP Congestion Control

The basis of SCTP congestion control is an amalgamation of current best practices for TCP implementations with extensions to deal with multi-homing aspect of SCTP and modifications due to the message rather than the stream-based nature of the protocol [19]. The standard specifies an adaptive sliding window control with adapted versions of the well known TCP slow-start, congestion avoidance, fast retransmit and fast recovery mechanisms. In addition recent work on TCP, such as congestion window validation [20], is also incorporated. One currently optional mechanism for TCP, the use of Selective Acknowledgements (SACKs) [21] to report out of sequence data arriving at the receiver, has been incorporated into SCTP as the baseline for congestion control implementation. This is due to the demonstrated superiority of this mechanism to earlier TCP congestion control options [22]. Although the possibility to support IP Explicit Congestion Notification (ECN) [23] has been incorporated into SCTP, this mechanism is optional and (as with TCP) a packet loss is the normal method of congestion indication.

2.1.4.1 SCTP Congestion Control Behavior

Normal operation of SCTP follows TCP in allowing a sender to inject Min (*rwnd*, *cwnd*) data into the network. Although in order to protect against lost SACKs when *rwnd* is zero, it is permitted to have one packet in flight (if allowed by *cwnd*) to probe for changes in the advertised receiver window credit.

SCTP congestion control, similar to TCP, uses a single control law called an Additive-Increase Multiplicative-Decrease (AIMD) to control both congestion and fairness [24]. During slow-start (at the start of an association, after a transmission time out or after a long idle period) *cwnd* is increased exponentially to probe the network for capacity. This occurs while $cwnd \leq ssthresh$. During this phase *cwnd* may only be increased by Min (total size of new data acknowledged by a SACK, PMTU) if, and only if, two conditions are true: the cumulative TSN ACK point must have been advanced by the SACK, and the full *cwnd* must have been used on the destination address(es). To tell if the full congestion window was used, an implementation will compare *flightsize* (before processing the SACK) to *cwnd*. This means that duplicate SACKs (ones which do not advance the cumulative TSN Acknowledgment point) can only enable new data transmission by reducing the sender's view of how much data is currently in the network (flightsize).

During the congestion avoidance phase ($cwnd > ssthresh$) *cwnd* may be incremented by more than one PMTU per *RTT*, if and only if both conditions stated in the previous paragraph are satisfied. This produces a linear increase in *cwnd*, up to the limit of the initial *rwnd* exchanged at the association initialization.

Whenever a SCTP sender is not transmitting to a receiver address (both during slow start and congestion avoidance), the *cwnd* is adjusted to Max(*cwnd*/2, 2*PMTU) per Retransmission Time Out (*RTO*). This protective measure decreases *cwnd,* as network conditions are unknown for a destination to which the sender is not transmitting, and is similar to the approach for TCP outlined in [25].

There are two methods of packet loss detection (taken as a congestion notification) defined in SCTP:

1. Detection of gaps in the received TSNs through Gap Acknowledgement reports in a SACK. Normally a sender will wait for four consecutive reports before reacting as packet re-ordering could occur in the network. In this case the path variables are reset as follows:

$$ssthresh = \text{Max}(cwnd/2, 2*\text{PMTU}) \quad (2.1)$$

$$cwnd = ssthresh \quad (2.2)$$

   This will have the effect of putting the sender in slow-start with a reduced *cwnd*. This will force the sender to wait before transmitting new data for either the SACK advancing the cumulative TSN ACK point beyond the missing packet (Normally retransmitted through fast Retransmit, described below) or until enough duplicate SACKs arrive to reduce the *flightsize* below the modified *cwnd*.

2. Timeout of the retransmission timer. In this case the path variables are reset as follows:

$$ssthresh = \text{Max}(cwnd/2, 2*\text{PMTU}) \quad (2.3)$$

$$cwnd = 1*\text{PMTU} \quad (2.4)$$

   This will have the effect of putting the sender in slow start and assure that no more than one packet is in flight until it receives the SACK advancing the cumulative TSN ACK point beyond the TSN of the retransmitted packet.

Whenever SCTP retransmission occurs, the sender bundles as many as possible of the earliest TSNs marked as missing into a single packet and retransmits them. Note that in the case of a Fast Retransmit triggered due to gap reports, this will correspond to the earliest TSNs that are reported as missing at least four times. For a timer expiry, it will be the earliest TSNs which have not yet been acknowledged by the cumulative TSN ACK point irrespective of gap ACK reports.

The *RTO* is adjusted based on estimates of the round trip delay and backs off exponentially as message loss increases. When the receiver is multi-homed, senders compute the *RTO* and maintain the retransmission timers on destination address basis.

SCTP also requires the use of path MTU discovery techniques.

2.1.4.2 Differences between SCTP and TCP Congestion Control

For the purpose of this section we treat the standard TCP congestion control mechanisms as these laid out in the most recent RFC defining TCP behavior [25].

There are probably two changes to the control scheme, which represent the most significant departure from standard TCP congestion control mechanism.

- The direct dependence on the number of bytes acknowledged rather than the number of acknowledgements received to increase the congestion window.
- The implicit dependence on selective acknowledgement (SCTP Gap ACK block) messages for reporting in the case of packet losses.

The first change should in theory produce slight more sophisticated control mechanism and is similar to experimental proposals for TCP proposed in [26]. It is also eliminates the need for additional control variables such as the pipe variables suggested for SACK TCP [22] to control the amount of data injected into the network during fast recovery as *cwnd* now also controls this. The second major change is in itself a welcome improvement over earlier TCP implementations (pre-SACK TCP) and should ensure greater resilience to packet loss than pre-SACK TCP implementations.

Minor changes to the control mechanism include:

- No explicit fast recovery phase is required, as no artificial inflation of *cwnd* is required for throughput to be maintained. The reason behind this that SCTP can clock out new data when duplicate ACKs are received.

- TCP allows a choice of congestion avoidance or slow-start when *cwnd* equals *ssthresh*. SCTP mandates slow-start in this case.

- During the congestion avoidance phase *cwnd* may only be increased if the full *cwnd* is currently being used. This restriction is not used in TCP, however current TCP research [26] indicates that it is preferable to the "standard" TCP approach of increasing *cwnd* based on incoming non duplicate ACKs. This limit *cwnd* in SCTP to be less than or equal to the receiver's initial advertised window.

- An unlimited number (barring PMTU restrictions) of gap ACK blocks are allowed in SCTP. TCP has a maximum of three SACK blocks. This should make SCTP more resilient to acknowledgement loss and better able to deal with multiple (non-sequential) packet loss than TCP.

- The value of *cwnd* decays when packets are no longer being transmitted. This protects against the use of an invalid congestion window during application limited periods as recommended by current TCP research [20].

- Fast retransmission of packets is controlled by *cwnd*. This reduces the speed with which SCTP reacts to packet loss and hence reduces throughput [27].

- During recovery from a packet loss, certain implementations of the SCTP fast retransmission algorithm are vulnerable to multiple retransmissions of the lost packet and a collapse of *cwnd* down to its minimum value of 2*PMTU with a corresponding decrease in throughput [27].

- Without gap block generation, SCTP has no way of detecting packet loss other than a retransmission time out [27].

2.1.5 SCTP Research Survey

Due to SCTP's attractive features, described in the previous sections, it has received much attention from the network community, in terms of both research and development [28].

Caro et al. [29] compare the performance of multimedia document retrieval over reliable transport services providing unordered, partial ordering, and ordered delivery. Performance analysis show that for all network loss rates > 0%, Partially Ordered/Reliable (PO/R) service provides, on average, better progressive display for parallel GIF images than Ordered/Reliable (O/R) service.

Atiquzzaman and Ivancic [30] study the impact of multi-streaming in increasing the performance of SCTP over error prone satellite network. Their results show that multi-streaming results in higher goodput than single streams when the receiver buffer is constrained as in the case of wireless portable handheld devices, also they demonstrate that multi-steaming feature of SCTP results in reduced buffer requirements at the receiver in the presence of losses in the satellite network.

Jungmaier et al. [31] compare the performance of TCP and SCTP on a network where a high bandwidth satellite link with large delay is used as the primary path, and a low bandwidth terrestrial link with low delay is used as a secondary path. Their results show that with the existence of random transmission errors on the primary path, SCTP can achieve better performance than TCP with respect to the delivery delay of retransmitted packets, as SCTP uses the low delay secondary link for retransmissions. The authors also recommend a modification to the standard SCTP so that the SACK chunks that contain gap reports are sent over the low delay secondary path, this will further decrease the time needed for retransmission.

Noonan et al. [32] study the interaction between SCTP and Mobile IP (M-IP) standard for IPv4. Results show a moderate benefit from SCTP multi-homing feature, and that in most circumstances SCTP will operate well with M-IP.

Ravier et al. [33] investigate the performance characteristics of multi-homed SCTP hosts through experimental studies. Their results show that the ability of SCTP to perform the retransmissions over an alternative path than the primary, could increase the association throughput, especially if the alternate path have better properties than the primary path.

Decoupling failure detection from recovery process is proposed by Caro et al. [34]. Failure detection remains conservative, i.e., the error counter must exceed "Path.Max.Retrans" of that destination address before marking the destination address as inactive, while the recovery process begins during early signs of possible failure. Authors argued that beginning the recovery process before the failure is detected has the advantage of providing improved throughput.

A problem in the current SCTP specification is illustrated by Iyengar et al [35]. This problem results in unnecessary retransmissions and "TCP-Unfriendly" growth of the sender's congestion window during certain changeover conditions. The reason behind this behavior is the inadequacies of SCTP – either (i) the sender is unable to distinguish between SACKs for transmission and SACKs for retransmissions, or (ii) the congestion control algorithm at the sender is unaware of the occurrence of a changeover, and hence is unable to identify reordering introduced due to changeover. A solution was proposed for the first problem through the use of two new chunks called Retransmission Identifier (RTID) chunk and the Retransmission Identifier (RTID) Echo. The RTID chunk is added to every outgoing data packet at the sender, and carries one bit per TSN in the packet. The bit is zero if its respective TSN is a first transmission, and is one if the TSN is a retransmission. The receiver echoes back these bits in the RTID echo chunk in the SACK. By comparing the corresponding bit for each TSN in the RTID echo with the corresponding RTID bit maintained locally, the sender can distinguish between SACKs for transmission and SACKs for retransmission, and accordingly decides to change the corresponding *cwnd* or not.

Ye et al. [36] examine SCTP throughput performance over IEEE 802.11 wireless LAN protocol using different congestion window sizes and number of hops between the source and destination. Results show that the throughput of SCTP degrades when the number of hops increases, due to hidden node and the exposed node problems. Also it is shown that increasing the window size will not help to increase the throughput, on the contrary it amplifies the problem, thus worsening the throughput.

Elsayed et al. [37] introduce a synchronization algorithm for periodic traffic in SCTP networks. Different levels of the algorithm are introduced to compromise between performance improvement and implementation complexity. The highest performance algorithm is called “maximum delay algorithm”. It recovers the Inter-arrival Time (IT) to its original value however it requires larger buffer especially in case of very high jitter. On the other hand, another algorithm, referred to as “average delay algorithm”, solves the buffer problem.

2.2 Bandwidth Aggregation

The idea of resource aggregation in order to obtain higher performance has been used in different areas in the computer and communication fields. Katz et al. [38] introduces a striping technique for the disk subsystem, which is now a key aspect in Redundant Arrays of Inexpensive Disks (RAID) architectures. Brenden et al. [39] provide an overview on the use of resource aggregation in the network subsystem, and introduce an evaluation criteria to judge the benefits provided from the resource aggregation in terms of: latency and buffering requirements, skew tolerance, scalability and complexity and finally the

maximum aggregate bandwidth that can be supported. Using these criteria, they examine and evaluate resource aggregation at each layer of the protocol stack.

Bandwidth aggregation mechanisms can be categorized according to the layer in which the mechanism works.

An example of link layer bandwidth aggregation is Bellcore's (currently Telcordia) effort, for their Aurora testbed [40], to obtain the equivalent bandwidth of an STS-12c (620 Mbps) by using four STS-3cs (155 Mbps) aggregated together into a trunk group. ATM cells are striped across the links in an order determined by the trunk control algorithm. The algorithm places idle and active cells that allows the receiver to determine the order in which the cells were placed on the trunk group. Duncanson [41] and Snoeren [42] introduce *Inverse Multiplexing*, a standard application-transparent method to provide higher end-to-end bandwidth by splitting traffic across multiple physical channels, creating a single logical channel. Most inverse multiplexing implementations assume physical transport mechanisms with constant bit rates and stable channel characteristics, in terms of bandwidth, latency and loss rates, such as found in circuit switched networks. For this reason, they use a *Round Robin* approach to assign data fragments to the available channels. Round robin data striping is simple and can provide perfect fair distribution of fragments on the available channels when the fragments are roughly of the same size, and when the channels characteristics are similar and stable. Variations in the characteristics of any particular channel can have catastrophic impact on the performance of the whole bundle. Magalhaes and Kravets [43] present an adaptive inverse multiplexing scheme, for Wireless Wide Area Networks, referred to as *Link Quality Balancing* that uses relative performance metrics to adjust traffic scheduling across

bundled links. Link Quality Balancing dynamically adjusts the Maximum Transfer Unit (MTU) of each link in proportion to the available bandwidth. By splitting packets into fragments that can be transmitted in roughly the same amount of time by each link, reassembly can proceed without delay. However, their scheme considers only bandwidth aggregation for links that belong to the same access network and thus have comparable characteristics.

On the network layer, Phatak and Goff [44] introduce a mechanism that uses IP-in-IP tunneling to hide the utilization of multiple interfaces from the receiver's TCP stack. The packets sent on each path are fragmented so as to equalize the *RTT* of the different paths. The proposed mechanism considers only the case where the effective bandwidth of the paths is constant throughout the lifetime of the TCP connection, which is not a practical assumption in mobile wireless networks. Papadopouli and Schulzrinne [45] present an architecture that enables network connection sharing in an environment of mobile wireless, collaborating hosts. The central idea is that dual-homed hosts with wireless connections to the Internet can share their connection with other hosts of the collaborating group by acting as temporal gateway. While this mechanism requires the involvement of intermediate nodes that acts as gateways, our proposed mechanism (LS-SCTP) is an end-to-end mechanism.

On the transport layer, Magalhaes and Kravets [46] present Reliable - Multiplexing Transport Protocol (R-MTP), a rate based transport protocol capable of multiplexing application data across multiple network interfaces. R-MTP depends on a packet pair probing mechanism to estimate the current available bandwidth over each channel. Due to the fast variation in the network condition, R-MTP bandwidth estimation is very

sensitive to the rate of probing. Inaccuracies in bandwidth estimation, can lead to out of order arrival of packets at the receiver and in some cases to re-sequencing buffer overflow. Hsieh and Sivakumar [47] present an end-to-end transport layer protocol called parallel TCP (pTCP). pTCP allows striping the application data on multiple TCP connections. The protocol is based on two components the Striping Manager (SM) and one or more TCP-virtual (TCP-v), which is a modified version of TCP that deals only with virtual packets and virtual buffers. The SM manager is responsible for opening and closing the TCP-vs, in addition to striping the application data on the different TCP-vs based on their current congestion status. As oppose to the multiple TCP-vs architecture in pTCP, our proposed bandwidth aggregation mechanism is based on a single LS-SCTP connection, due to SCTP inherent multi-homing support, which reduces the overhead of starting and ending multiple connections. In addition, pTCP does not support dynamic addition/deletion of new paths during the lifetime of the transport connection.

On the application level, Allman et al. [48] propose a modified version of FTP, called XFTP, for large delay-bandwidth product channels, such as satellite channels. As limited receiver window size can throttle the TCP throughput over these channels, XFTP uses multiple parallel TCP connections (sockets) to simulate a virtual TCP connection with a large receiver window size. Hacker and Athey [49] present an analytical model to find the optimal number of parallel sockets that can achieve higher throughput for the applications and at the same time do not lead to network congestion. Implementing the bandwidth aggregation on the application level increases the complexity of the applications, as they are responsible for striping and re-sequencing the data. In addition, application level bandwidth aggregation mechanisms stripe the data on the different

sockets without taking in consideration the differences in the paths characteristics. This can lead to a situation where a slow path can drag down the throughput of the whole bundle.

2.3 Reliable Wireless Video

The increase in the bandwidth of wireless channels and the computing power of mobile devices increase the interest in video communications over mobile wireless networks. Beyond the limited available bit-rate, wireless video offers a number of interesting technical challenges. One of the most difficult issues is due to the fact that mobile networks cannot provide a guaranteed Quality of Service (QoS), and high bit rates occur during fading periods. Transmission errors of a mobile wireless radio channel range from single bit errors to burst errors or even an intermittent loss of the connection. Without special measures, compressed video signals are extremely vulnerable against transmission errors. Basically, every bit counts. Considering specifically low bit-rate video, compression schemes rely on inter-frame coding for high coding efficiency, i.e., they use the previous encoded and reconstructed video frame to predict the next frame. Therefore, the loss of information in one frame has considerable impact on the quality of the following frames. Since some residual transmission errors will inevitably corrupt the video bit-stream, this vulnerability precludes the use of low bit-rate video coding schemes designed for error-free channels without special measures. A great varity of error control and concealment techniques have been proposed during the last 10 – 15 years [50]. These schemes can be divided into source coder-independent and source coder-dependent schemes.

2.3.1 Source Coder-Independent Schemes

Source coder-independent techniques for video transmissions are described similarly in three categories.

Sender-based schemes employ *intelligent packetization* at the sender side to prevent two kinds of propagation losses. First, because of exclusive use of Variable Length Coding (VLC) in compression standards, packet losses often cause the loss of synchronization in a coded bit stream, rendering subsequent packets useless. Turletti and Huitema [51] propose to divide a coded bit stream into packets according to inserted synchronization points. If a synchronization unit (e.g., Group of Blocks (GOB) in MPEG and H.263) cannot fit into a single packet, it is further divided according to smaller syntactic units (e.g., macro-blocks). Second, most coding schemes rely on temporal-difference coding to achieve high coding efficiency, thereby introducing a pervasive dependency structure into a bit stream. To prevent the propagation effects due to difference coding, Chen [52] proposes a dependency tree-based scheme that put all information derived from a common ancestor into a single packet. In this approach, a lot of side information has to be sent to ensure proper assembly of the received packets at the decoder.

Receiver-based techniques are motivated by the insensitivity of human perception to high frequency components. The processing is carried out in spatial domain, temporal domain, frequency domain, or some combinations of the above.

Spatial-domain recovery makes use of the smoothness assumption of video signals through a minimization approach. Zhu et al. [53] propose an approach that recovers a lost block by minimizing the sum of squared differences between the boundary pixels of the

lost block and its surrounding blocks. This smoothness measure often leads to blurred edges in the recovered image. Kwok and Sun [54] propose an approach to minimize variations along edge directions or local geometric structures. They require accurate detection of image structures, and mistakes can yield annoying artifacts in a reconstructed video.

Temporal-domain recovery exploits temporal correlation by replacing a corrupted block by its corresponding block on the motion trajectory in the previous frame [55]. The difficulty with this approach is that it relies on the knowledge of motion information that may not be readily available in all circumstances.

Frequency-domain recovery performs reconstruction by interpolating each lost coefficient in a damaged block from the corresponding coefficients in its four neighboring blocks [56]. Because the correlation of pixels in adjacent blocks is likely to be small, the interpolation does not produce satisfactory results.

Other approaches employ some combinations of the above three techniques to reconstruct lost data. Zhu et al. [57] propose maximum-smoothness recovery that extends the smoothness property to both spatial and temporal neighbors. Lee et al. [58] introduce POCS (Projection onto Convex Sets) that formulates spatial- and temporal-smoothness constraints into convex sets and derives a solution iteratively. Besides computationally expensive, designing post-processing schemes at the receiver, independent of the encoder at the sender, may not result in high-quality reconstruction because the two are usually closely related.

Sender receiver-based schemes involve the cooperation of both the sender and the receiver in concealing errors.

Forward error-correction (FEC) methods have been proposed for video communications in the past [59]. Besides increasing transmission bandwidth, it also introduces long delay in decoding and is difficult to apply in packet networks because hundred of bytes of data may be lost in a burst and need to be recovered.

Retransmissions have generally been considered inappropriate for real-time streaming applications because of delays introduced. To make use of retransmitted data, a naive decoder will wait for requested retransmitted packets before playing subsequent data, leading to long freezes in playback. More complex decoders conceal lost video by a certain recovery method, without waiting for retransmitted packets to arrive [60]. At this point, error concealment introduces certain inaccuracies in the video data that will propagate in playback. Upon arrival of retransmitted data, the affected pixels due to error propagation will be corrected according to a complex relationship. Ghanbari [61] proposes a retransmission-based error control technique without incurring additional latency by rearranging the temporal dependency of frames so that a reference frame is referenced by its dependent frames much later than its display time, thereby masking the delay in recovering lost packets. This approach, however, has difficulty in handling cascade loss scenarios in which packet losses happen again before the arrival of re-transmitted packets.

Interleaving or *scrambling* [62] reorders image pixels to be transmitted in such a way that packet losses cause isolated losses that may be approximately reconstructed using their

surviving neighbors. It is applicable to situations where neighboring pixels are highly correlated, but may not work well when adjacent pixel values are changing rapidly. To overcome this difficulty, Wah and Su [63] propose a similar linear transformation to improve the reconstruction quality when some of the interleaved streams were lost during transmission.

2.3.2 Source Coder-Dependent Schemes

This class of techniques performs error recovery by adding redundant information in the source and/or channel coders.

Source-coding schemes exploiting redundancy. Error recovery at the decoder in a traditional source-coder design is very difficult because redundancy is removed to the largest extent in order to achieve the best compression. Hence, adding redundancy intentionally in the source coder is one way to achieve increased robustness.

Robust entropy coding (REC) decreases the effects of error propagation when packet losses result in wrong decoding states. One way is to periodically insert synchronization codes in the bit stream [64], although the length of inserted code words has to be long enough to prevent false synchronization. Kawahara and Adachi [65] propose a second scheme that distributes code words from individual blocks into slots of equal size. In this approach, the proper slot size is hard to determine. Sikora [67] introduces a the third technique, included in the error-resilient mode of MPEG-4, in which a reversible VLC is employed in such a way that once a synchronization word is found, the coded bit stream can be decoded backwards. This approach achieves improved robustness at reduced coding efficiency.

Restricting prediction domain (RPD) tries to reduce degradations due to temporal-difference coding. Cote et al. [68] introduce video-redundancy coding modes in H.263, *independent segment decodin*g, and *dynamic reference picture selection*. These schemes can only limit adverse effects of error propagation due to prediction coding.

Layered coding (LC) has been an area of active research in the past decade in the context of ATM [69] and wireless networks [70]. Layered coding partitions data into a base layer and a few enhancement layers. The base layer contains visually important video data that can be used to produce output of acceptable quality, whereas the enhancement layers contain complementary information that allows higher-quality video data to be generated. In networks with priority support, the base layer is normally assigned a higher priority so that it has a larger chance to be delivered error free when network conditions worsen. Layered coding has been popular in ATM networks but may not be suitable for Internet transmissions for two reasons. First, the Internet does not provide priority delivery service for different layers. Second, when the packet-loss rate is high and part of the base layer is lost, it is hard to reconstruct the lost bit stream since little redundancy is present. Shunan et al. [71] propose a scheme for reliable transmission of video over bandwidth limited ad-hoc networks. A raw video stream is layer coded and the base layer and the enhancement layer packets are transmitted separately on two disjoint paths. The base layer packets are given higher protection than enhancement layer packets. As will be shown in our performance study, in Section 4.3.5, the quality of the reconstructed video depends on the condition of the path that carries the base-layer packets. This will cause the video quality to fluctuate with the error characteristics of a single path. Even a very small loss rate on the base layer will make the layered coder less desirable.

Multiple-description coding (MDC), in contrast, divides video data into equally important streams such that the de-coding quality using any subset is acceptable, and that better quality is obtained by more descriptions. It is assumed in *MDC* that losses to different descriptions are uncorrelated, and that the probability of losing all the descriptions is small. Apostolopoulos [72] proposes to code a video source into multiple descriptions, using temporal frame sub-sampling, and to transmit them over multiple paths through either IP source routing or relay service. Gogate et al. [73] examine the effectiveness of combining *MDC* and *Multiple Path Transport* (*MPT*) for video and image transmission in a multi-hop mobile radio network. The primary inefficiency in *MDC* is that coding separate independent substreams reduces the compression efficiency since the frames are spaced farther apart, in addition to the redundancy added to each description so that it can be decoded independently of the other descriptions.

Wang et al. [74] and Lee et al. [75] compare the error-resilience capabilities of LC and MDC in case of multiple paths transport under different path environments.

Source coding and channel coding achieves error resilience by adding error correction codes [76]. It differentiates from FEC in coder-independent schemes because its distribution of protection is closely related to the source-coder output. For example, intra-frames are guarded by more protection bits in H.263-alike coders. Again, the use of such schemes in video communications must be based of prudent decisions because video transmissions are already very bandwidth-intensive. In addition, such schemes are stationary and must be implemented to guarantee a certain QoS requirement for the worst-case channel characteristics. Due to the fact that wireless channel is non-stationary, and the channel bit error rate varies over time, these techniques are associated with

unnecessary overhead that reduces the throughput when the channel is relatively error free. Furthermore, in mobile wireless networks, the assumption that burst length is short does not hold as long burst losses can be common due to fading and channel interference.

Joint source channel-coding (JSCC) schemes minimize transmission errors by designing jointly quantizers and channel coders, according to a given channel error model [77]. To cope with noisy channels, they try to optimally partition bandwidth between the source and channel coders, depending on channel-loss status, normally characterized by some parameters. They are, however, hard to apply in the Internet since the Internet does not have a well-defined channel model.

2.4 Adaptive Internet Multimedia

The development and use of distributed multimedia applications are growing rapidly at present. Some common examples are video-conferencing, Internet telephony, and video-on-demand. High-quality delivery of multimedia information requires high network bandwidth. Also, since minimum audio and video quality is required in order to communicate the desired information, video and audio applications require a certain minimum throughput for useful operation. Finally, in order to support interactive conversations, and to ensure synchronization of data belonging to different streams, there should be an upper bound on the end-to-end delay, and on the maximum variation in delay [78].

The special characteristics of multimedia applications place a number of requirements on the network. The requirements can be specified in terms of QoS parameters, such as throughput, packet loss, delay, and jitter. In a network providing undifferentiated, best-

effort service without any QoS support mechanisms, fluctuations in network load can adversely affect multimedia applications. Also, multimedia applications on the Internet commonly employ the UDP transport protocol, which lacks any congestion control mechanism. As a result, applications with high bandwidth can severely overload the Internet, and starve TCP applications (which perform congestion control) of their fair share of bandwidth.

Different approaches may be considered to address these shortcomings. One approach is to enhance the network with mechanisms such as resource reservation, admission control, and special scheduling mechanisms, so that a certain level of QoS can be guaranteed to an application. A certain degree of QoS support can also be provided by allowing differentiated or prioritized service at network switches.

Another approach is to adjust the bandwidth used by an application according to the existing network condition. This approach has the advantage of better utilizing available resources (which change with time), compared to approaches relying on resource reservation. It is also facilitated by the nature of existing multimedia applications, many of which allow the media rate and quality to be adjusted over a wide range. At the same time, the special requirements of multimedia applications mean that strictly TCP/SCTP-like congestion control may not be suitable for these applications. The rate halved (to a first order approximation) for every lost packet in TCP/SCTP congestion control, which may cause corresponding sharp changes in encoder parameters to achieve the desired rate, and unpleasant perceived quality. Too small rate may also violate the minimum throughput requirements of the application.

Multimedia adaptation has been studied for Internet applications, and the adaptive control schemes can be classified into receiver-driven, sender-driven and transcoder-based [78].

Receiver driven schemes allow receivers individually to tune the received transmission according to their needs and capabilities. Mehra and Zakhor [79] modify the TCP protocol at the receiver end to provide video streams a nearly Constant Bit Rate (CBR) connection over a bandwidth limited access link. Hsiao et al. [12] present Receiver-based Delay Control (RDC) in which receivers delay TCP ACK packets based on router feedback to provide constant bit rate for streaming. While receiver buffers can be used for smoothing out rate fluctuations, buffering is limited by the end-to-end latency limit.

A majority of the sender-driven algorithms may be grouped under quality adaptation schemes. Quality adaptation techniques can further be classified into on-the-fly encoding, adding/dropping layers, and switching among multiple encoded versions. Kanakia et al. [80] estimate the buffer occupancy and the service rate received by the connection at the bottleneck queue through periodic feedback messages from the network. These estimates are used to control the transmission rate of each video frame on-the-fly by adjusting the encoder quantization factor. However, in general, on-the-fly encoding is CPU intensive and thus regarded as unsuitable for streaming real-time video. This is especially true when the sender has to service many different clients with different bandwidth requirements. In the adding/dropping layers scheme, the video stream is partitioned into several layers using scalable coding schemes such as MPEG-4 Fine Granularity Scalable (FGS) video or inter-frame wavelet video encoding. Video streaming applications can add or drop enhancement layers to adjust the transmission rate to the available bandwidth. In the switching-versions scheme, the video is encoded at different rates, and therefore

different quality levels, and each of these versions is made available to the streaming server as an independent stream. The server detects changes in available bandwidth and switches among the input streams, in order to adapt the transmission rate to the available bandwidth. Quality adaptation that is based on multiple encoded versions has been shown to provide better viewing quality than adding/dropping layers, due to the layering overhead [13].

Besides quality adaptation, scheduling algorithms may also be used to improve the multimedia streaming adaptation. Saparilla and Ross [14] prefetch portions of the video into the client buffer as bandwidth is available.

Transcoder-based schemes [81] decode and re-encode the video at gateways placed at appropriate locations to deliver different levels of quality to clients with different bandwidths.

2.5 Summary

This chapter started with a description of the Stream Control Transmission Protocol (SCTP) discussing the salient features, and comparing it to TCP. In Section 2.2, we surveyed some different approaches for bandwidth aggregation, concentrating on transport level solutions. In Section 2.3, we discussed and critiqued conventional approaches to tackling the problems caused by wireless bit-errors and user mobility on video transport in mobile wireless networks. In Section 2.4, we described different approaches for adaptive multimedia transport in the Internet.

We observe that current approaches towards handling bandwidth aggregation, wireless bit-errors, and Internet video streaming do not comprehensively solve observed problems. The material described in this chapter sets the stage for our work. The next chapters describe in detail our approaches for improving multimedia transport in bandwidth limited and error prone networks.

Chapter 3
Transport Layer Bandwidth Aggregation

This chapter describes the design and performance analysis of the *Load Sharing SCTP (LS-SCTP)*, an extension to the SCTP for bandwidth aggregation. LS-SCTP significantly improves the end-to-end performance of SCTP in error-prone mobile wireless networks. It achieves this by exploiting the idea of separating between the SCTP flow and congestion control for enhanced end-to-end performance.

3.1 Introduction

As mobile wireless communication become a key factor in our daily life, the demand for reliable connectivity for mobile devices increases and the requirements on that connectivity become more stringent. Connectivity for mobile devices is governed by the communication technology on the device and the coverage area of the technology. To add versatility to the mobile devices, they are built with multiple communication technologies and the capacity for expansion. The coverage areas for these wireless technologies overlap so as to provide uninterruptible communication. In order to support better handoffs and communication quality, the mobile node's operating system can coordinate

the simultaneous use of multiple diverse communication channels, providing seamless connectivity as the mobile node migrates between coverage areas *vertical handoffs* [82].

Currently, many of the mobile communication devices are occupied with multiple wireless interfaces. The main target of the multiple interfaces is to increase the communication reliability, so that if one of the interfaces fails, the other interfaces can still provide a communication path. If the mobile node has access to multiple communication channels, the application can be guided as to which channel currently the most appropriate. Such support is limited to the use of one communication channel per application data stream. Due to the limited and variable bandwidth of the wireless channels, as well as their lossy and break prone nature, the performance of mobile applications can be greatly hurt.

The key contribution of our research is the implementation and performance evaluation of a mechanism for aggregation of the resources across multiple heterogeneous channels in a mobile environment. The proposed mechanism provides the mobile applications with a logical channel that has a bandwidth equal to the bandwidth of all the available channels.

In the network subsystem, load sharing can be performed at different levels of the protocol stack. We chose our bandwidth aggregation mechanism to work at the transport layer, as the transport layer has better knowledge about the characteristics of the communication paths than other layers, so it can aggregate the bandwidth of paths with different characteristics, in terms of bandwidth, delay and loss rate. In addition, the proposed mechanism does not need any involvement from the intermediate nodes in the

network. These properties are beneficial as it provides high throughput and reliability to time sensitive applications.

We present a transport layer solution for aggregating the bandwidth of the available interfaces. Our solution is based on extending Stream Control Transmission Protocol (SCTP) [3]. Our choice to extend SCTP for bandwidth aggregation was motivated by its inherent support for multi-homing, as well as the different useful features, presented in Section 2.1. Currently, SCTP uses multi-homing for redundancy and not for load sharing. Each endpoint chooses a single destination address as the primary destination address for all data units, "data chunks" in SCTP terminology, during normal transmission. Retransmitted data chunks use the alternate address(es), under the assumption that this can improve the probability that the data chunks can reach the peer endpoint. Continues failure to send to the primary address ultimately results in the decision to transmit all the data chunks to an alternate destination until the primary destination become reachable again.

Our proposed SCTP bandwidth aggregation extension, which we refer to as Load Sharing-SCTP (LS-SCTP), utilizes the available paths for simultaneous transmission of data chunks, while maintaining the SCTP congestion control on each path, to ensure fair integration with other traffic in the network [83][84]. In order to achieve the bandwidth aggregation, taking in account the differences in the paths characteristics in terms of bandwidth, loss rate and delay, LS-SCTP separates between the congestion control and flow control, such that the congestion control is performed on a path basis, while the flow control is kept on association basis. In addition, LS-SCTP monitors the status of the available paths and as their status change, i.e., new paths become active or existing paths

break, it updates the list of a*ctive* paths, which includes the paths that are currently used for load sharing. As delaying the delivery of lost chunks can affect the association throughput, and can cause the whole association to stall, LS-SCTP includes a retransmission technique that ensures fast delivery of lost data chunks. As will be shown in the performance evaluation, in Section 3.4, these features allow LS-SCTP to utilize all of the available paths efficiently, which consequently benefits the upper layer applications.

During the association initialization, LS-SCTP allows the user to enable/disable the load sharing capability, as well as controlling the number of interfaces that can be used simultaneously for load sharing. This feature is important for battery-powered hosts, as it enables the LS-SCTP user to conserve the battery power by controlling the simultaneous use of the interfaces.

The rest of this chapter is organized as follows. In Section 3.2, we discuss the reasons that make SCTP, in its current form, not suitable for load sharing. In Section 3.3, we describe the details of LS-SCTP design, which allow SCTP to be load sharing capable. We present performance results from our LS-SCTP implementation in Section 3.4. We conclude this chapter with a summary in Section 3.5.

3.2 Load Sharing in SCTP

Transmission paths can have different characteristics, in terms of Round Trip Time (*RTT*) and congestion state. Consequently, if SCTP simply stripes the load on the different paths, data chunks can arrive out of order at the receiver. Out of order arrival of data chunks can

unnecessary initiate Selective Acknowledgements (SACKs) transmission to the sender, to report the gaps in the Transmission Sequence Number (TSN), which increase the network load. In addition, the arrival of four SACKs, as oppose to 3 SACKs in TCP, at the sender reporting the same TSNs will be interpreted as loss of the TSNs. This can lead to unnecessary retransmission of data chunks, as well as halving the congestion window (*cwnd*) of the path on which the TSNs were originally sent [35].

To illustrate this problem, we assume two multi-homed hosts A and B. Host A has interfaces A_1 and A_2, and host B has interfaces B_1 and B_2, and all the four addresses are bound to an SCTP association. For one of several possible reasons (e.g., path diversity, policy based routing), we assume that the data traffic from A to B_1 is routed through A_1, and from A to B_2 is routed through A_2. Figure 3.1 shows the timeline of events.

The vertical lines represent interfaces B_1, A_1, A_2 and B_2. Each arrow depicts the departure of a packet from one interface and its arrival at the destination. The labels on the arrows are either SCTP TSNs or labels $ST_C(T_{GS} - T_{GE})$ which represents a packet carrying SACK chunk with cumulative ACK T_C, and gap ACK for TSNs T_{GS} through T_{GE}. C_1 is the *cwnd* at A for destination B_1, and C_2 is the *cwnd* at A for destination B_2. C_1 and C_2 are denoted in terms of Path Maximum Transfer Units (PMTUs). We assume that the Round Trip Time (*RTT*) to $B_1 > 2 * RTT$ to B_2, and at a certain time the *cwnd* of both destination addresses is 4 PMTU. Also we assume that data chunks with TSNs 21-24 were sent to B_1, and data chunks with TSNs 25-28 were sent to B_2. Because of the difference in the *RTT*s, data chunks sent to B_2 arrive earlier to the receiver end point initiating 4 SACKs that report the loss of TSNs 21-24, this will falsely lead the sender to retransmit TSNs 21-24 to B_2, and cutting the *cwnd* of B_1 to one half of its value.

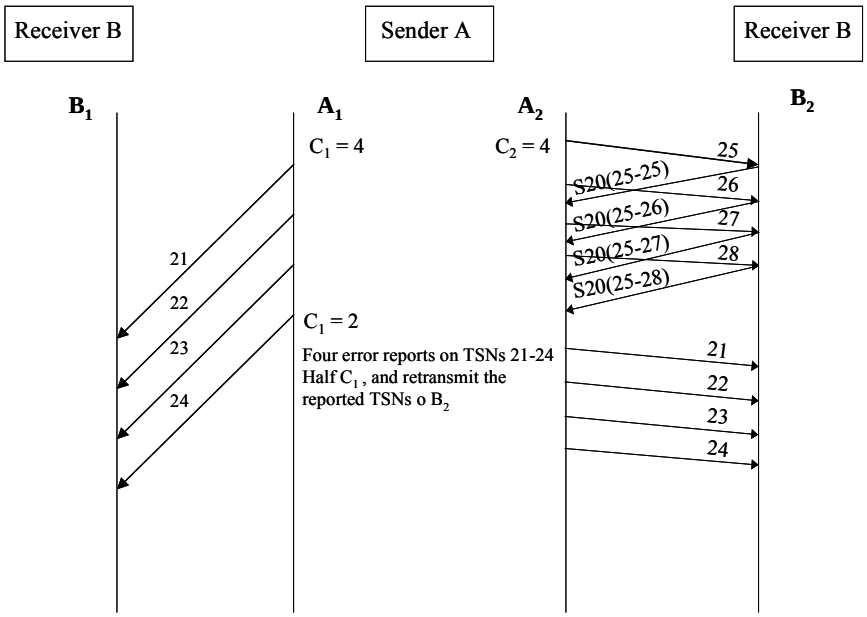


Figure 3.1 Effect of distributing SCTP packets on different paths.

The continuous generation of duplicate SACKs not only forces the slow start threshold to a very small value but also forces the congestion window to fluctuate around the small slow start threshold, and prevents the slow start threshold and the congestion window from growing again. Therefore, the end-to-end throughput becomes very low although no link congestion ever takes place. As standard SCTP is designed to deal with a single path transmission, a single monotonically increasing TSN is used on the different paths. Therefore there is no explicit notification telling if duplicate SACKs are caused by packet losses or by out of sequence induced by the asymmetric characteristics of the paths.

Techniques for adapting the fast transmission threshold with the degree of reordering in the network can partially solve the problem [85]. Although these techniques adapts the sender with the reordering in the network, the receiver is still not able to differentiate between the reordering that is due to the difference in the delay of the paths and that is due to packet losses. Thus the receiver continues to generate SACKs to report the out of

order arrival of data chunks, which consume the network resources. Delaying the generation of the SACK from the receiver to account the out of order arrival of data chunks, due to the difference in the paths delays, works fine in the case of lossless networks. For networks with losses, delaying the SACKs lead to a delay in reporting packet losses to the sender, which can eventually timeout.

3.3 LS-SCTP Design

LS-SCTP is designed to be robust to the variations in the paths characteristics. This is achieved by separating the association flow control from congestion control. In LS-SCTP, the flow control is on association basis, thus both the sender and receiver use their association buffer to hold the data chunks regardless their transmission paths. On the other hand, congestion control is performed on per path basis, thus the sender has a separate congestion control for each path, which can be used simultaneously for data chunks transmission. This provides the sender endpoint with a virtual congestion window (*cwnd*) size equal to the aggregate of the *cwnd*s of all the paths within the association that are used for load sharing. The congestion control on each path is following SCTP standard [3], to ensure fair integration with other traffic in the network. An architectural view of LS-SCTP is shown in Figure 3.2.

In order to separate the flow control from the congestion control, LS-SCTP uses two different sequence numbers. The first sequence number is the Association Sequence Number (ASN), that is a per association sequence number, and is used to reorder the received data chunks at the receiver association buffer, regardless the path from which they have been received.

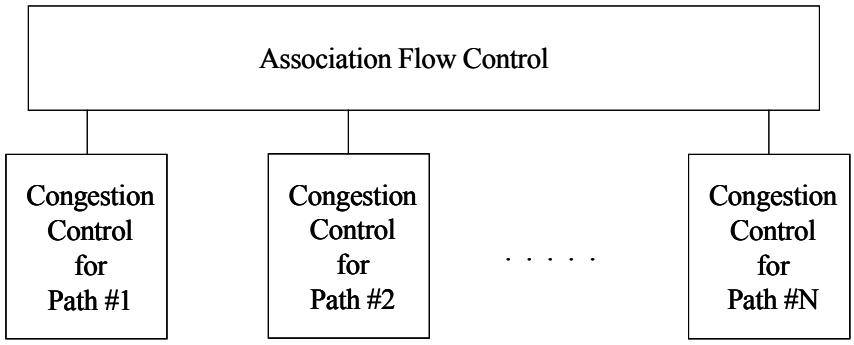


Figure 3.2 Architectural view of LS-SCTP.

The second sequence number is the Path Sequence Number (PSN), which is a per path sequence number, used for reliability and congestion control on each path. In addition, LS-SCTP continues to use the Stream Sequence Number (SSN) for ordering the data chunks within the association streams.

To support load sharing, we defined a new data chunk, *Load Sharing Data Chunk*, by adding two new parameters to the standard SCTP data chunk. The first parameter is a 4 bits Path Identifier (PID), which identifies the path used for the data chunk transmission. The second parameter is 12 bits Path Sequence Number (PSN), which is a monotonically increasing sequence number for the data chunks transmitted over the same path. Figure 3.3 shows the load sharing data chunk.

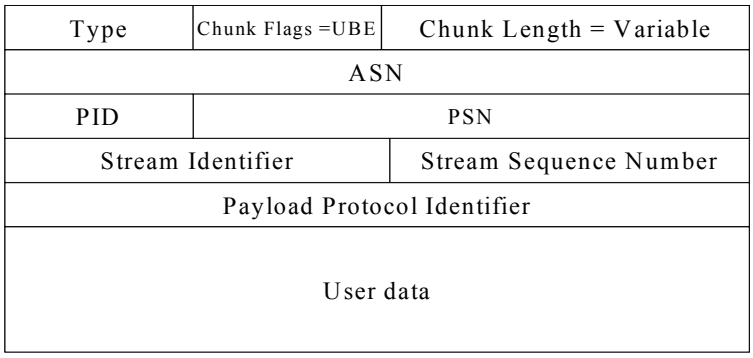


Figure 3.3 Load sharing data chunk.

For acknowledging the received data chunks, LS-SCTP uses *Load Sharing SACK (LS-SACK),* shown in Figure 3.4. LS-SACK, includes 2 additional parameters than standard SCTP SACK control chunk, a time stamp, and a per path Cumulative Acknowledgment. As the LS-SACKs are received on all the paths within the association, and due to the different path delays, LS-SACKs can be received out of order. Out of order arrival of LS-SACKs can cause the sender to develop an incorrect view of the receiver's buffer space, as the LS-SACK includes the receiver's free buffer space at the time it was sent.

For this reason, the LS-SACK includes a time stamp that is used by the receiver of the LS-SACKs to determine their order. The receiver of an old LS-SACK does not update its view of the peer's free buffer space based on the values in the LS-SACK, but on the other hand it uses the remaining information in the LS-SACK to update the congestion control variables for the path from which it was received.

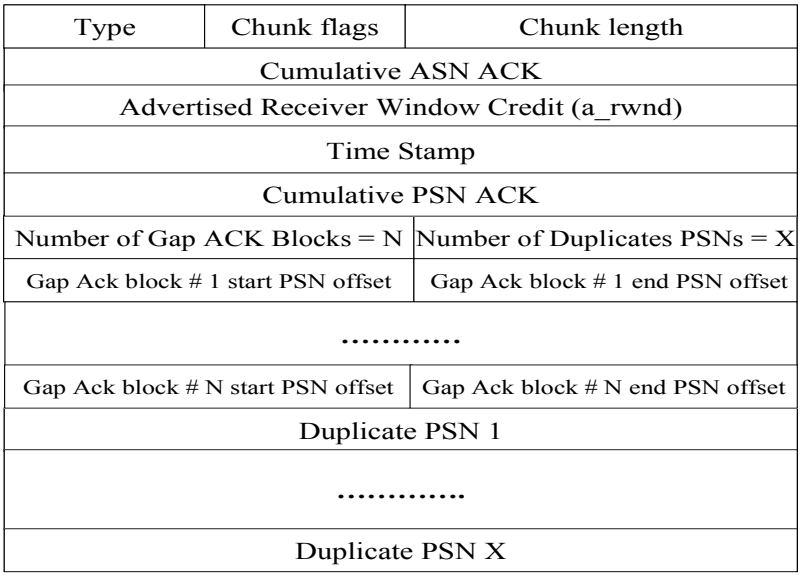


Figure 3.4 Load sharing SACK chunk.

The congestion control variables for each path, including *cwnd*, slow-start threshold (*ssthresh*), *RTT*, etc., are maintained in a data structure called the "Logical Buffer". In addition, the logical buffer keeps track of the PSNs sent or received on each path. The receiver window size (*rwnd*), which represents the available space in the peer's association buffer as well as the total outstanding data count are kept on an association basis. Figure 3.5 illustrates the functional components of LS-SCTP [86].

As the sender's application layer passes a data packet to the LS-SCTP, it is assigned an in-sequence ASN as well as a Stream ID and SSN, before it is placed in the association buffer. The path assignment module selects the transmission path for the data chunks and accordingly the PID and PSN are assigned to each data chunk.

At the receiver, the congestion control variables in the logical buffer corresponding to the path from which the data chunks were received are updated, and the data chunks are placed in the association buffer, until they are delivered to the application, in the order of the stream they belong to.

To ensure backward compatibility with standard SCTP, during the association initiation both end points have to exchange a Load-Sharing-Supported parameter in order to start using LS-SCTP. If any of the end points is not load sharing capable or not willing to use load sharing over the association, in order to conserve the battery power for example, it does not include this parameter.

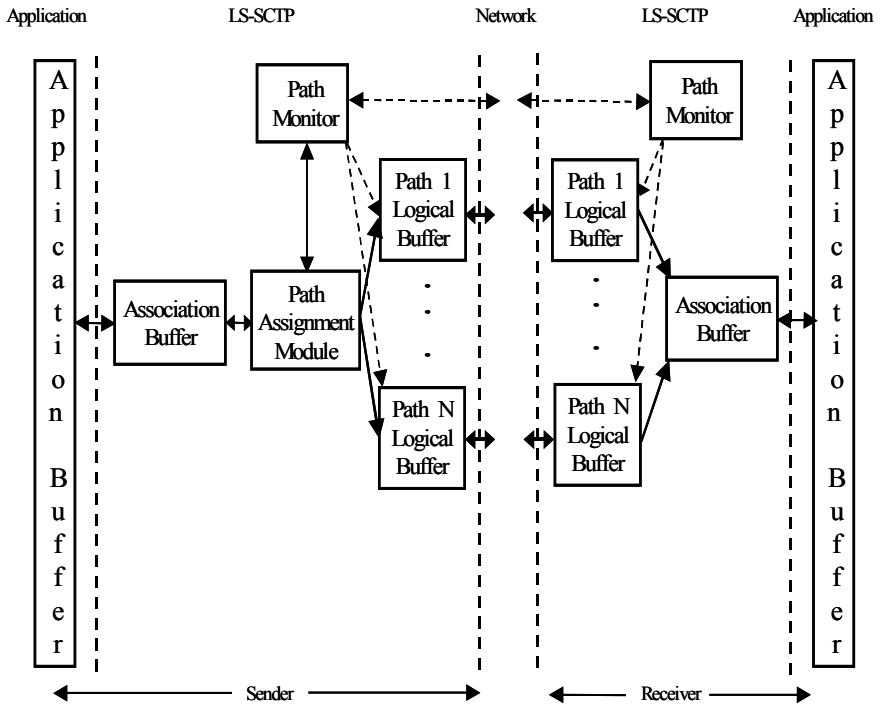


Figure 3.5 Functional components of LS-SCTP.

In this case, standard SCTP is used over the new association. In addition, the *Upper Layer Application* (*ULA*) can set the maximum number of interfaces that can be used simultaneously for load sharing. This feature is important for controlling the power consumption in battery-operated mobile devices.

3.3.1 Path Assignment Module

The Path Assignment module in LS-SCTP is responsible for assigning transmission paths for the data chunks. Round robin path assignment is not suitable when the transmission paths have different and variable characteristics. Such mechanisms can limit the throughput of the association to the throughput of the slowest path. Thus it is important that the path assignment be based on the ratio of the bandwidth of the paths within the association. LS-SCTP uses the current congestion window of each path, as an estimate of

its current bandwidth-delay product [87]. The path assignment module assigns data chunks to the paths according to the *cwnd*/*RTT* of each path. After assigning a data chunk to a path, the data chunk is assigned the *PID* that identifies the path, as well as an in-sequence *PSN*.

At the receiver, the data chunks do not compete on association buffer, as the sender controls the amount of data injected on all the paths, based on its view of the free space in the receiver's association buffer (*rwnd*), as well as the total outstanding data on all the *active* paths. In order to compensate the differences in the paths *RTT*, the minimum receiver association buffer should be based on the summation of the bandwidth of all the paths and the maximum *RTT* [46], as shown in equation 3.1.

$$\text{Minimum Receiver Association Buffer} = \left(\sum_{i}^{N} B_i\right) * RTT_{Max} \qquad (3.1)$$

where, B_i is the bandwidth of path i, N is the number of *active* path within the association and RTT_{Max} is the maximum *RTT*.

To improve the probability that retransmitted data chunk(s) reaches the peer endpoint, the path assignment module retransmits the data chunk(s) on a different path than the one used for the original transmission. Delaying the arrival of a data chunk can stall the association, because data chunks that were sent on other paths will be queued at the receiver waiting the missing chunk. At a certain point the receiver's association buffer space will limit the sender from sending more data chunks, until it receives the missing chunk. The *PSN* that was assigned to the data chunk(s) on the original path is returned to the *PSN* pool of this path, and can be used for new data chunk(s). The retransmitted data chunk(s) will be assigned *PID* and *PSN* on the retransmission path. In addition, the

outstanding data counts on the old and new paths are adjusted according to the size of the retransmitted data chunk(s).

Similarly, when time-out occurs on one of the *active* paths within the association, the path assignment module assigns the timed-out data chunks to different paths than the one they were originally timed-out on, which have enough capacity to transmit them. On the contrary of standard SCTP, that uses a new data chunk for probing the timed-out path, the path assignment module assigns a copy of one of the retransmitted data chunks for probing the path [47]. The reason behind this, that there is a probability that the timed-out path can continue to fail, which delays the re-transmission of the new data chunk until the path time-out again. Mean while the receiver buffer will start to fill up with data chunks, following the delayed one, that are received from other *active* paths within the association, and this can eventually lead the whole association to stall. The situation can be worse if the path fails for a long period. As the *RTO* of a path increases exponentially with consecutive time-outs, the association can repeatedly stall for exponentially increasing periods, as will be shown in Section 3.4. This can continue until either the path recovers or it is marked as *inactive* by the path monitor.

3.3.2 Path Monitor

As LS-SCTP utilizes multiple paths for transmission, a failure of a single path, or the increase in a path loss rate can affect the throughput of the whole association. For this reason, LS-SCTP includes a path monitoring mechanism referred to as the *Path Monitor* that extends the limited path monitoring capabilities of standard SCTP, described in Section 2.1.3. The path monitor is responsible for updating the *active* paths list, which

includes all the paths that can be used for load sharing. This can be based on network feedback, regarding the failure/recovery of paths within the association. For example, when mobile IP protocol [88] reports a new *Care Of Address* (*COA*), with a *PATH-ADD* message, the path monitor performs the following actions: 1) It adds the new path to the existing association, using the *Address Configuration Change* (*ASCONF*) chunk [88]. 2) It creates a logical buffer for the new path. 3) Finally, it adds the new path to the *active* paths list. Consequently, LS-SCTP starts to use the new path for data chunks transmission. On the other hand, when the path monitor receives a *PATH-LOSS* message from the mobile IP, it performs the following: 1) It removes the path from the *active* paths list 2) It deletes the logical buffer that corresponds to the path. 3) Finally, it removes the path from the association using the *ASCONF* chunk. If the network is not capable of providing a feedback regarding the paths status, LS-SCTP depends solely on its inherent path monitoring capability for updating the *active* paths list. The path monitor removes a path from the *active* paths list when the number of consecutive retransmission time-outs on the path exceeds *Path.Max.Retrans*, which is set to 5, or when the path quality deteriorate to a limit that can affect the performance of the whole association. Currently, we are using a reasonable default threshold for the average loss rate on each path, and basing the path membership in the *active* paths list on the threshold value. In future research, we are planning to investigate techniques for adaptive setting of the path membership threshold. *Inactive* paths remain as a part of the association, and the sender keeps monitoring them through Heartbeat control chunks. As soon as a path recovers, the path monitor adds it again to the *active* paths list. As will be shown in the experimental

results, this technique prevents stalling the association while waiting for a missing data chunk.

3.4 Performance Study

In order to examine the performance of LS-SCTP, we extended our publicly available *OPNET* network simulator [89] SCTP implementation [90]. In our simulation, we used the network topology shown in Figure 3.6. We assume that an SCTP association is already initiated between the two hosts, and the association is unidirectional, which means that data chunks will only be sent from the host A (sender) to host B (receiver). In addition, we use only one stream within the association. Nodes 1-N are used to configure the characteristics of the transmission paths in terms of bandwidth, loss rate, delay and delay jitter, as well as path failures. The application packet size is set to 1 KBytes. We set the application packet inter-arrival time to ensure that the application will always have packets for transmission. During all our simulations, the application starts generating packets after 0.1 second from starting the simulation. The receiver's association buffer size is set according to equation (3.1). Unless specified otherwise, the bandwidth of all the paths between host A and host B are set to 1.5 Mbps, with RTT = 100 ms and average packet loss rate = 10^{-4}. We set the probing interval *HB.Interval* for the path monitor to 1 second.

We used the association throughput as the performance metrics, which is defined as the number of bytes delivered to the receiver's application layer per second.

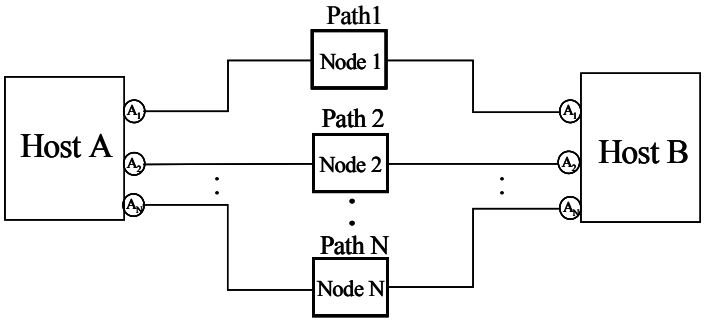


Figure 3.6 Network model for the simulation study.

In order to investigate the suitability of LS-SCTP, specific questions we looked at in our performance study:

1) Does LS-SCTP scale well with the number of transmission paths?

2) How sensitive is LS-SCTP throughput to the asymmetric characteristics of the paths in terms of bandwidth, loss and delay?

3) How does LS-SCTP react to path failures?

3.4.1 Scalability with the Number of Paths

We examined the scalability of LS-SCTP with increasing the number of transmission paths. Figure 3.7 shows the LS-SCTP association throughput as we increase the number of *active* paths within the association. It can be observed that the performance of LS-SCTP scales well with increasing the number of *active* paths, and it is able to aggregate their bandwidth efficiently.

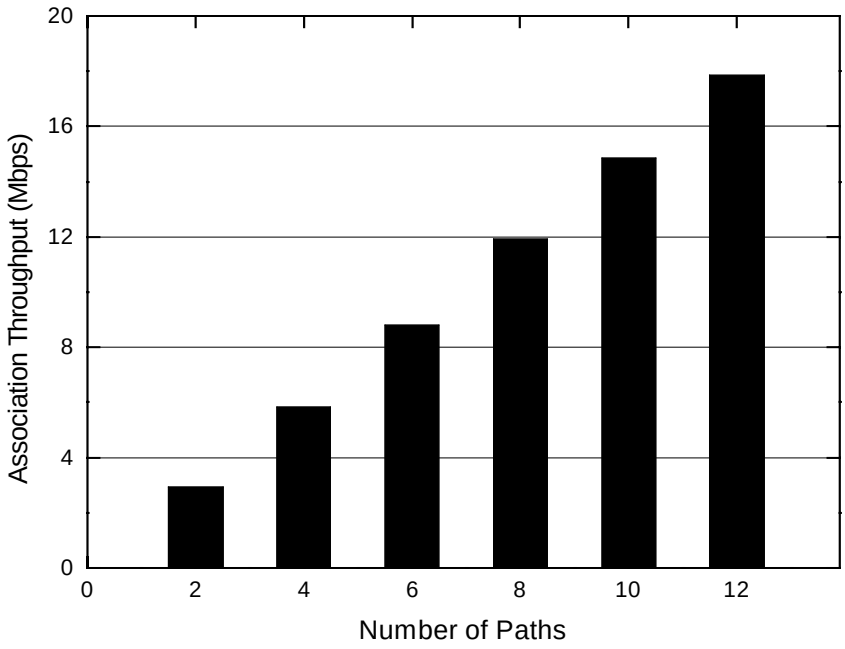


Figure 3.7 Association throughput versus the number of transmission paths.

3.4.2 Effect of Bandwidth Fluctuation

We studied the effect of bandwidth fluctuation on LS-SCTP performance. We used two paths between host A and host B, namely path-1 and path-2. We represented the bandwidth fluctuation on path 1, by using a square-wave background traffic with amplitude 1 Mbps, and period t = 40 seconds. While we assumed that there is no background traffic on path-2. We run the simulation for 120 seconds. Figure 3.8 shows the association throughput. It can be seen that LS-SCTP is able to adapt with the bandwidth fluctuation on the transmission paths, as the sender is splitting the load on the paths based on an estimate of their bandwidths. Also LS-SCTP is able to utilize the available bandwidth between the communicating endpoints. The difference between the association throughput and the available bandwidth is mostly due to the overhead from various protocol levels.

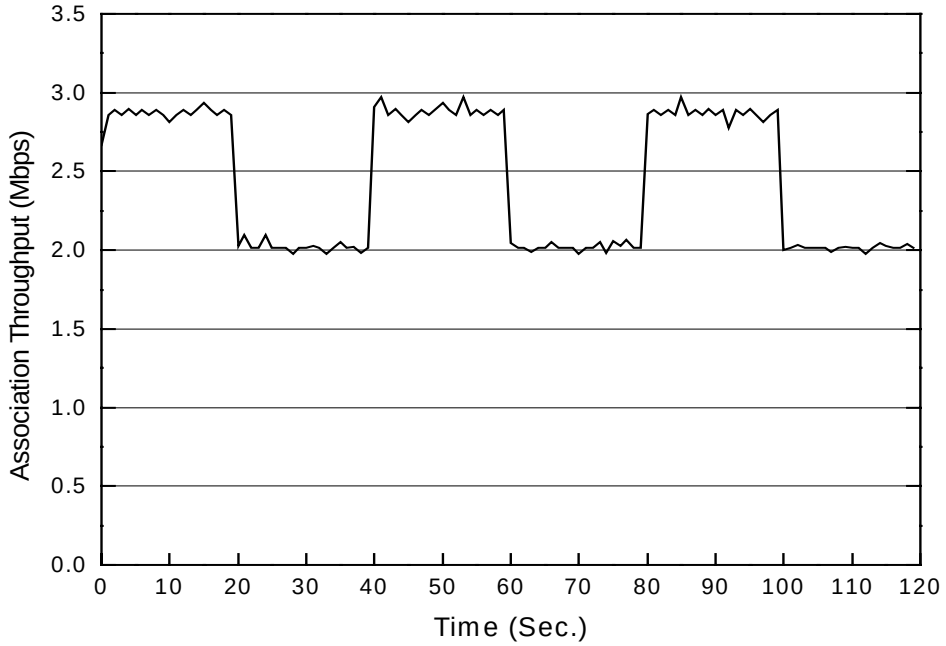


Figure 3.8 Effect of bandwidth fluctuation on the association throughput.

3.4.3 Effect of Packet Losses

We examined the performance of LS-SCTP under different packet loss rates. We assumed that the association includes two paths. We changed the average packet loss rate of node 1, on path-1, between 10^{-5} and 10^{-2}, while we assumed that the average packet loss rate on path-2 is set to 10^{-4}. Figure 3.9 shows the total throughput of LS-SCTP under different packet loss rates. For reference, we included the throughput achieved by SCTP under the same loss rates. Because of the randomization in this experiment, each point in the graph represents the average of ten samples with different seeds. It can be seen that due to the packet retransmission technique used in LS-SCTP, it still achieves higher throughput than SCTP despite the different packet loss characteristics of the paths.

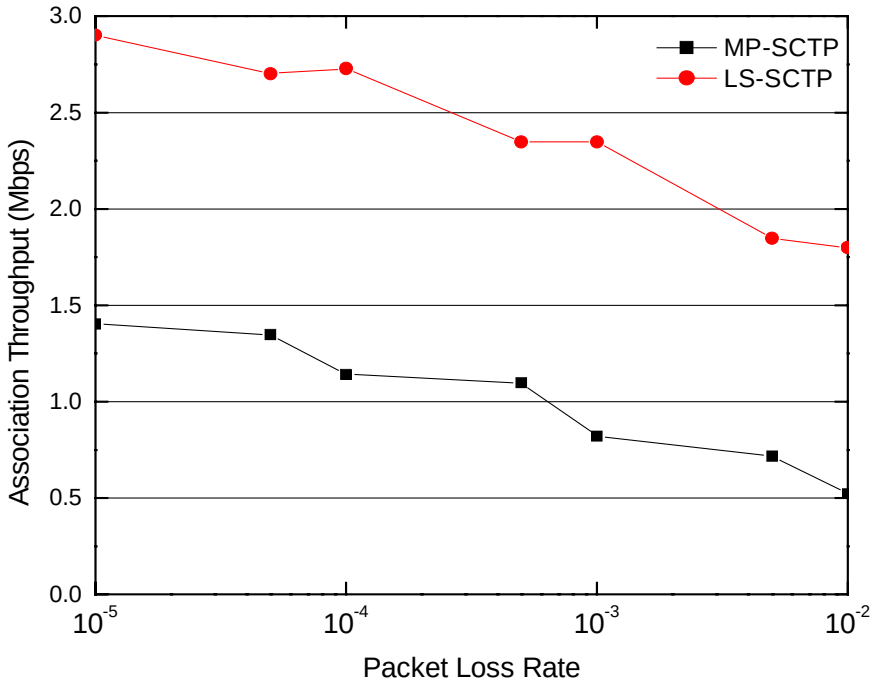


Figure 3.9 Total throughput versus packet loss rate.

3.4.4 Effect of Delay Asymmetry

We examined the performance of LS-SCTP under paths with asymmetric delays. We assumed that the association includes two paths. We configured the delay on path-1 to 10 ms. We varied the delay on path-2 from 10 to 70 ms. Figure 3.10 shows the association throughput for SCTP and LS-SCTP.

As can be seen that as the delay between the paths increases SCTP throughput will decrease due to continuous drop in the *cwnd* and *ssthresh*, as discussed in Section 3.2. As LS-SCTP uses a separate congestion control mechanism for each path, the receiver is able to handle the out of order arrival of packets from different paths, thus it is able maintain the association throughput despite the difference in the paths delay.

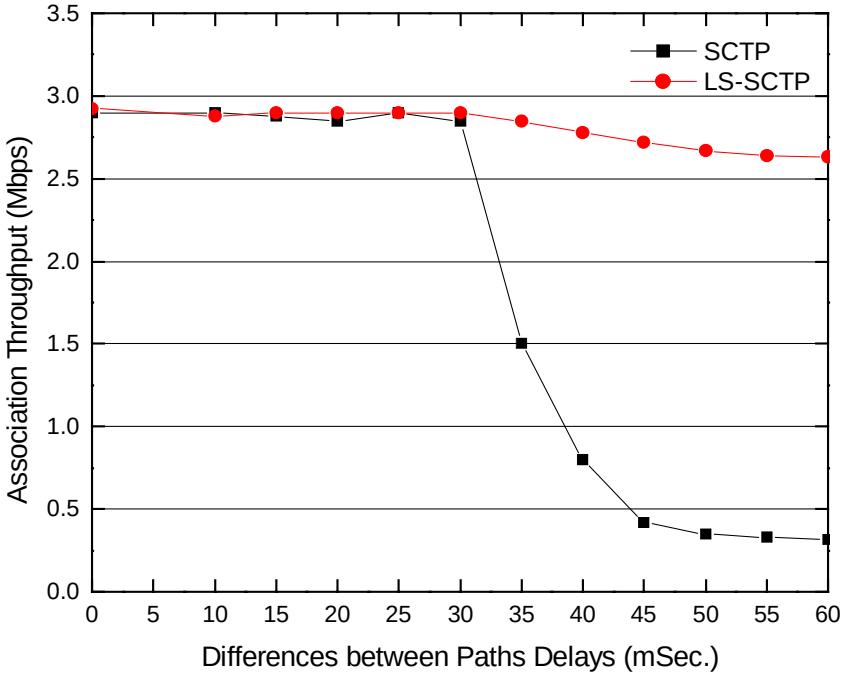


Figure 3.10 Total throughput vs. difference in paths delay.

3.4.5 Effect of Path Failure

To examine the effect of path failure on the LS-SCTP, we assumed two *active* paths within the association, and path-1 failed for ten seconds, after five seconds from initiating the association. We compared two techniques for handling timed-out paths. The first technique, which we refer to as load sharing unaware time-out handling, is similar to that used in SCTP, where the timed-out data chunks are retransmitted on an alternative path, and a new data chunk is used for probing the bandwidth of the timed-out path. The second technique, which we refer to as load sharing aware time-out handling, is the technique proposed in Section 3.3, where the timed-out data chunks are retransmitted on alternative path, other than the one they already timed-out on, and a copy of one of the retransmitted data chunks is used for probing the timed-out path. After the receiver sends a LS-SACK to acknowledge the data chunk copy, the sender starts to use the path again for transmission of data chunks.

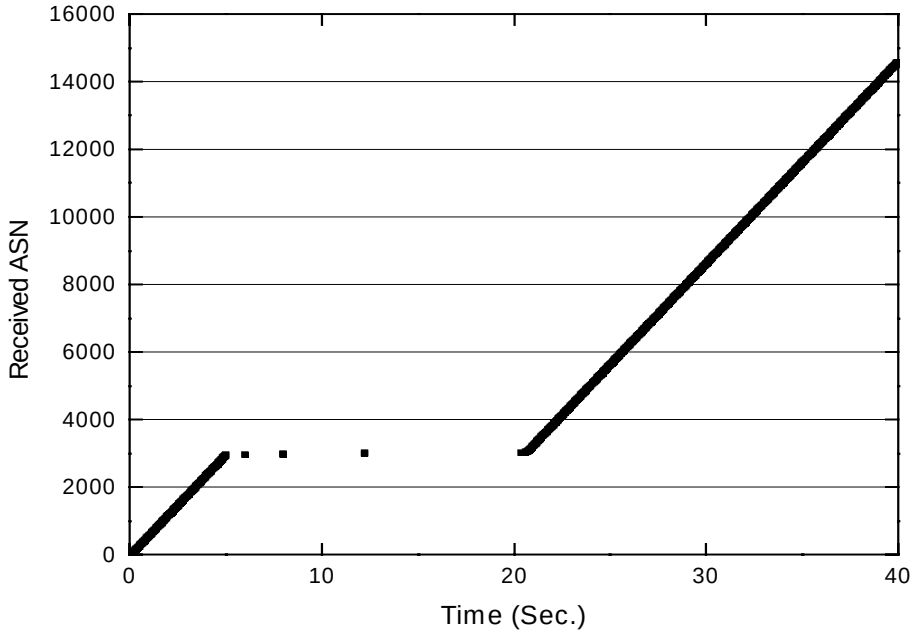


Figure 3.11 Received ASNs progression under load sharing unaware time-out handling.

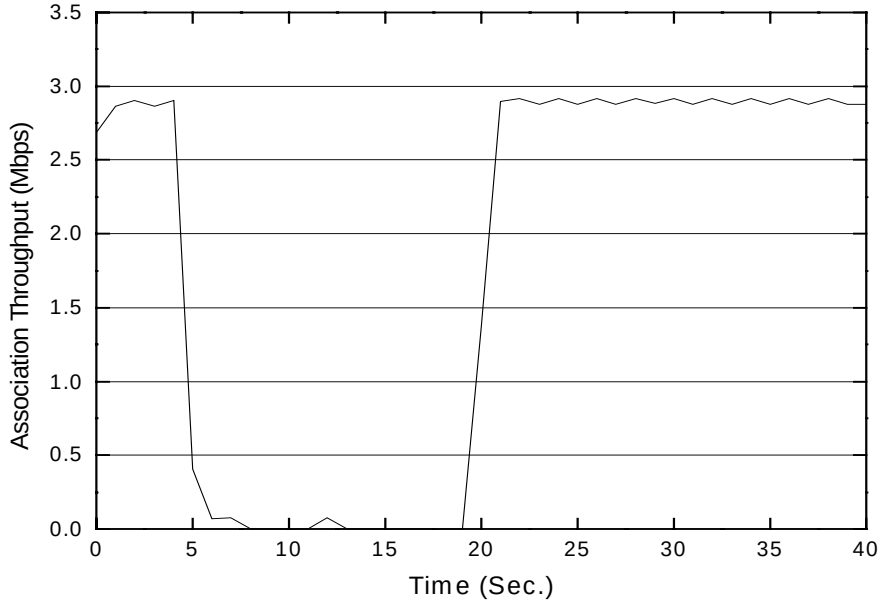


Figure 3.12 Association throughput under load sharing unaware time-out handling.

Figure 3.11 shows the received ASNs progression, under load sharing unaware time-out handling. As can be seen that the whole association stalled, for around 15 seconds, after path-1 timed-out. The reason behind this behavior, that after the first time-out of path-1, the sender retransmitted all the timed-out data chunks on path-2, then started to probe path-1 with a new data chunk, ASN = 2944. At the same time, the sender continued to send the data chunks following ASN = 2944 on path-2. This caused the receiver association buffer to fill up, as it is waiting for ASN = 2944, preventing the sender from sending more data chunks. This caused the whole association to stall, until ASN = 2944 is timed-out and retransmitted on path-2, allowing the receiver to deliver all the data chunks in the association buffer to the application. At this time, the sender resumed the transmission again. This cycle repeated for three times. Also, as we can notice from the figure, that even after path-1 is recovered at t = 15 seconds, the association continued to stall, until path-1 is timed-out again at t = 20 seconds The reason behind this that path-1 *RTO* increased exponentially after each time-out, which led the association to stall for exponentially increasing periods of time. Figure 3.12 shows the association throughput, with load sharing unaware time-out handling. As can be seen that the association throughput dropped to zero most of the interval between t = 5 to t = 20 seconds.

We repeated the simulation using load sharing aware time-out handling. Received ASN progression, in Figure 3.13, shows that using a copy of a retransmitted packet to probe path-1 *cwnd,* led the association to resume using path-2. This had the effect of reducing the association throughput, as shown in Figure 3.14, but with minimum interruption to the association progress. After the recovery of path-1, the sender started to utilize it for transmission. This consequently increased the association throughput.

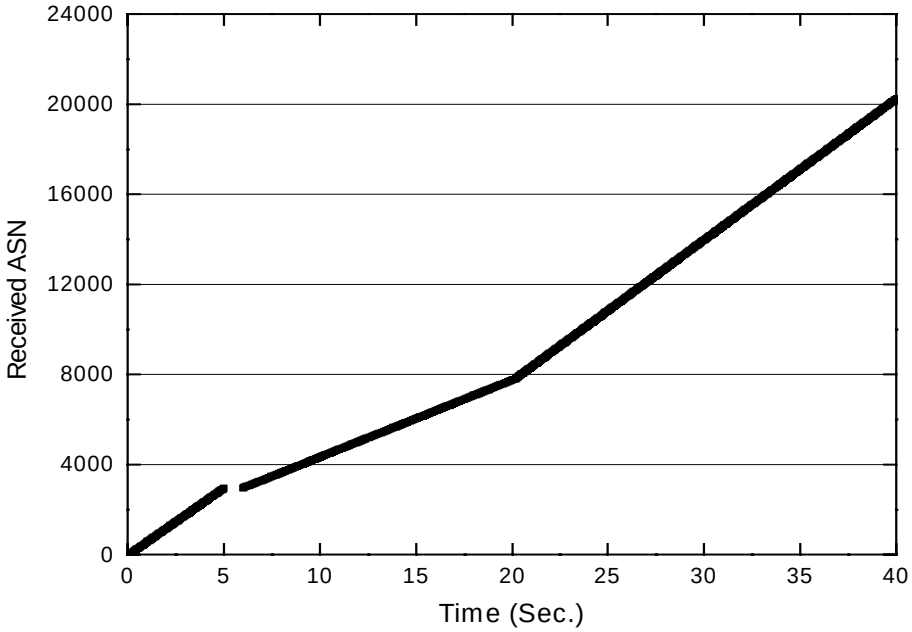


Figure 3.13 Received ASNs progression under load sharing aware time-out handling.

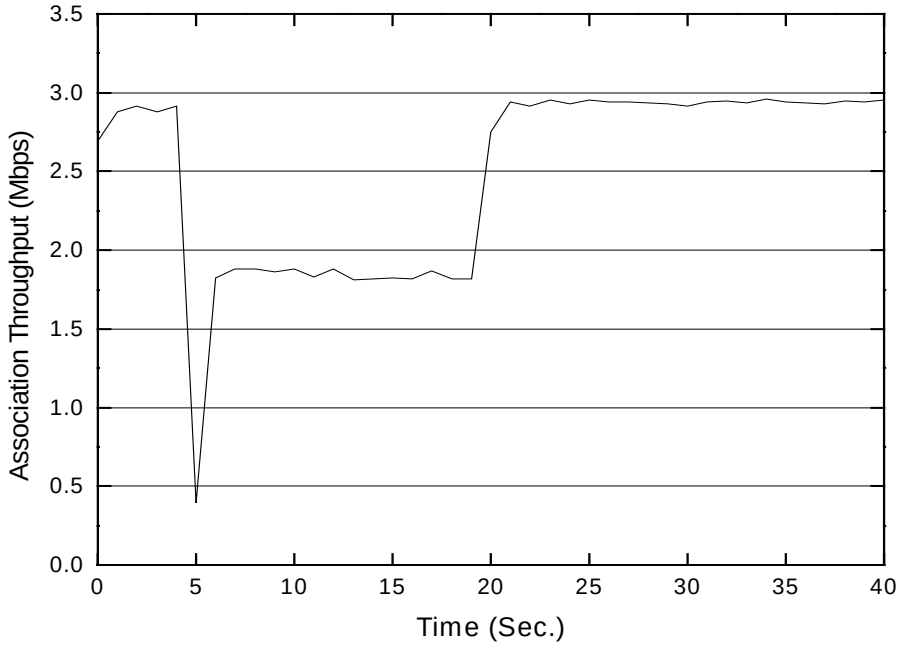


Figure 3.14 Association throughput under load sharing aware time-out handling.

The reduction of the association throughput after path-1 failure occurred for a duration approximately equal path-1 *RTO,* before the sender stopped considering path-1 for transmitting new data chunks. This throttling interval can be decreased by increasing the receiver association buffer, than the minimum value specified in equation (3.1), as this will allow the receiver buffer to absorb more sender's transmission before filling up.

3.5 Summary

In this chapter, we presented an extension for SCTP, LS-SCTP, which aggregates the available bandwidth on the paths between the sender and receiver. We first showed that SCTP in its current form is not suitable for load sharing. Then, we suggested remedies for SCTP to make it load sharing capable. We proposed separating the association flow control and congestion control. The congestion control is performed per path while the flow control is performed per association. Also we proposed modifying the sender to control the distribution of data on the available paths based on an estimate of the bandwidth of each path. In addition, we presented a technique that allows the sender to react to path failure and path quality deterioration without stalling the whole association. LS-SCTP performance study shows that it is scalable with the number of transmission path. Also it shows that LS-SCTP is robust to the asymmetric characteristics of the paths and able to deal efficiently with path failures. We believe that the proposed bandwidth aggregation mechanism is beneficial for providing high throughput and reliable communication for mobile applications.

Chapter 4

Multi-Level Error Protection for Mobile Wireless Video

In this chapter, we present a mechanism that extends retransmission-based error control to provide different levels of protection to video packets through redundant retransmission on diverse paths. The mechanism factors in the importance of the retransmitted packets to the reconstructed video quality as well as the end-to-end latency constraints to minimize the overhead and maximize the reconstructed video quality at the receiver. The proposed retransmission mechanism maintains the video quality under different loss rates and with less overhead compared to error control methods that depend on controlling the intra-update rate. In addition, the mechanism is shown to be more robust to wireless losses than schemes that combine layered coding with path diversity.

4.1 Introduction

In recent years there has been an increasing trend towards personal computers and workstations becoming portable. Desire to maintain connectivity of these portable computers to the existing installation of Local Area Networks (LANs), Metropolitan Area

Networks (MANs), and Wide Area Networks (WANs), in a manner analogous to present day computers, is fueling an already growing interest in wireless networks. In the same time, the increase in the bandwidth of wireless channels and the computing power of mobile devices increase the interest in video communications over wireless networks. However, in such networks there is no end-to-end guaranteed Quality of Service (QoS) and packets may be discarded due to bit errors. Wireless channels provide error rates that are typically around 10^{-2}, which range from single bit errors to burst errors or even intermittent loss of the connection. The high error rates are due to multi-path fading, which characterizes mobile radio channels, while the loss of the connection can be due to the mobility in such networks. In addition, designing the wireless communication system to mitigate these effects can be complicated by the rapidly changing quality of the radio channel.

The effect of the high error rates in wireless channels can be devastating for the transport of compressed video. Video standards, such as MPEG [5] and H.263 [6], use motion-compensated coding to reduce the temporal and statistical redundancy between the video frames. Although motion-compensated coding can achieve high compression efficiency, it is not designed for transmission over lossy channels [7]. In this coding scheme the video sequence consists of two types of video frames: *intra-frames* (I-frames) and *inter-frames* (P- or B-frames). I-frame is encoded by only removing spatial redundancy present in the frame. P-frame is encoded through motion estimation using preceding I- or P-frame as a reference frame. B-frame is encoded bi-directionally using the preceding and succeeding reference frames. For each image block in an inter-frame, motion estimation finds a closely matching block within its reference frame, and generates the displacement

between the two matching blocks as a motion vector. The pixel value differences between the original inter-frame and its motion-predicted frame are encoded along with the motion vectors. This poses a severe problem, namely error propagation (or error spread), where errors due to packet loss in a reference frame propagate to all of the dependent frames leading to visual artifacts that can be long lasting and annoying [8].

Different approaches have been proposed to tackle the error propagation problem. One approach is to reduce the time between intra-coded frames, in the extreme case to a single frame. Unfortunately, I-frames typically require several times more bits than P- or B-frames. While this is acceptable for high bit-rate applications, or even necessary for broadcasting, where many receivers need to resynchronize at random times, the use of the intra-coding mode should be restricted as much as possible in low bit rate point-to-point transmission, as typical for mobile wireless networks. The widely varying error conditions in wireless channels limit the effectiveness of classic forward error correction (FEC), since a worst-case design would lead to a prohibitive amount of redundancy. Closed-loop error control techniques, like retransmission, have been shown to be more effective than FEC and successfully applied to wireless video transmission. But for interactive video applications, the playout delay at the receiver is limited, which limits the number of admissible retransmissions [9].

In this chapter, we propose a mechanism to provide error resilience to interactive video applications in mobile wireless networks. The mechanism extends retransmission-based error control with redundant retransmissions on diverse paths between the sender and receiver. As different paths can have independent loss characteristics, sending multiple copies of the retransmitted packet on different paths can increase the probability that the

packet get received in less number of retransmissions. With a network loss rate l, the error rate can be reduced to

$$Error\ Rate = l^{1+\sum_{i=1}^{N} M_i} \tag{4.1}$$

where N is the maximum number of retransmission trials, which is typically determined by the initial playout delay in the receiver as well as the round-trip delay. M_i is the number of retransmission copies during the i^{th} retransmission, which depends on the importance of the retransmitted data to the reconstructed video quality. The maximum number of copies MAX(M_i) is equal to the number of available paths between the sender and receiver. The priority for each data unit in the stream is determined by the application. Thus in the context of motion compensated coding, the application can assign higher priority for I-frames data than P- or B- frames data. Also P-frames might be assigned varying priority levels, since P-frames that are closer to the preceding I-frame are more valuable for preserving picture quality than later P-frames in the *Group of Pictures* (*GOP*). This prioritization scheme can also be applied on the macro-block basis for encoding schemes with the flexibility to select the coding mode (i.e., intra or inter coding) on the macro-block level [6]. To ensure in-time delivery of retransmitted packets, and to prevent re-transmitting expired packets, the retransmission is controlled by the packet lifetime, as well as estimate(s) of the path(s) delay. As will be shown in our performance study in Section 4.4, this can significantly limit the effect of error propagation and improves the quality of received video, under a limited playout delay.

As the interactivity of the video session can be hurt by the continuous failure of the transmission paths, which can be due to mobility, our mechanism monitors the paths *Round Trip Time* (*RTT*) and accordingly it switches between them seamlessly.

We implemented the proposed mechanism as a sub-layer above *Real-time Transport Protocol* (*RTP*) [10]. We refer to this sub-layer as *Multi-Path-RTP* (MP-RTP). MP-RTP is responsible for: 1) Maintaining the priority level and the lifetime for each packet, as specified by the application, as well as implementing a delay constrained retransmission. 2) Monitoring the status of the available paths, and switching between them.

This chapter is organized as follows. Section 4.2 presents in detail the proposed mechanism. We present experimental results and performance evaluation in Section 4.3. Finally, we conclude the chapter with a summary in Section 4.4.

4.2 Proposed Solution

As our proposed solution is based on path diversity, we first describe different approaches to set up multiple diverse paths between the sender and receiver in wireless networks, then we describe in detail the proposed architecture that uses redundant retransmissions and path diversity for providing error resilience to interactive video in mobile wireless networks.

4.2.1 Path Diversity in Wireless Networks

There are several ways to set up multiple diverse paths in a wireless network. In a single hop wireless network, such as the cellular phone network or wireless local area network, a mobile node would need to establish channels to multiple base stations instead of one. This is already done in “soft” hand-off systems, during the hand-off phase. Alternatively,

the mobile and base station can each be equipped with multiple transmitter and receiver antennas, and one can consider each corresponding pair of transmitter and receiver antennas as constituting a separate path.

The particular communication environment of multi-hop wireless ad hoc networks makes multi-path transport (MPT) very appealing. In ad hoc networks: i) individual links may not have adequate capacity to support a high bandwidth service[1]; ii) a high loss rate is typical; and iii) links are unreliable. MPT can provide larger aggregate bandwidth and load balancing for ad hoc video applications. In addition, the path diversity inherent in MPT can provide better error resilience performance [91]. Furthermore, many of the ad hoc network routing protocols, e.g., dynamic source routing (DSR) [92], ad hoc on-demand distance vector (AODV) [93], and zone routing protocol (ZRP) [94], are able to return multiple paths in response to a route query. Multi-path routing can be implemented by extending these protocols with limited additional complexity. In the CDMA system, a node can communicate with multiple neighbors simultaneously by having multiple transceivers in each mobile, and using either receiver-oriented or link-oriented codes, or a code for each transmitter-receiver pair. Analogously, in a FDMA or a TDMA based system, a mobile could talk to its neighbors using multiple frequency channels or time-slots. However, there are many challenges in supporting MPT in ad hoc networks. First, from multiple paths returned by a route query, the routing process should select a set of maximally disjoint paths. Second, finding and maintaining multiple paths requires higher

[1]Although in some cases the nominal bandwidth of a wireless link is comparable to that of a wireline link, the available bandwidth may vary with signal strength as in IEEE 802.11b. In addition, capacity lost due to protocol overhead in ad hoc networks is much higher than that in wireline networks (e.g., RTS, CTS, ACK packets, and the 30-byte frame header in IEEE 802.11b).

complexity and may cause additional overhead on traffic load (e.g., more route replies received). Third, a problem inherent in MPT is the additional delay and complexity in packet resequencing.

4.2.2 System Architecture

The ability to successfully decode a compressed bit stream with inter-frame dependencies depends heavily on the recipient of reference frames, and to a lesser degree on dependent frames. Thus, we propose a mechanism to provide adaptive end-to-end unequal error protection for packets belonging to different frames, without sacrificing the timely requirement for interactive video. We achieve the unequal protection through redundant retransmissions over diverse paths between the sender and receiver. Due to the statistical independence of the packet loss events over different paths, by re-transmitting the packets over separate paths, we are maximizing the probability that at least one packet is received error-free, in least number of retransmissions. This behavior is required especially for interactive video, with limited playout delay at the receiver. This scheme is adaptive in the sense that the retransmission overhead will only be added when there is loss in the stream, and the degree of the overhead is proportional to the importance of the lost packets. We implemented the mechanism as a sub-layer above RTP. We refer to this sub-layer as MP-RTP. Figure 4.1 shows the system architecture [95].

For each video frame, the sending application assigns a priority level, which is based on the frame's importance to the reconstructed video quality. For example, I-frames can be assigned higher reliability level than P- or B- frames.

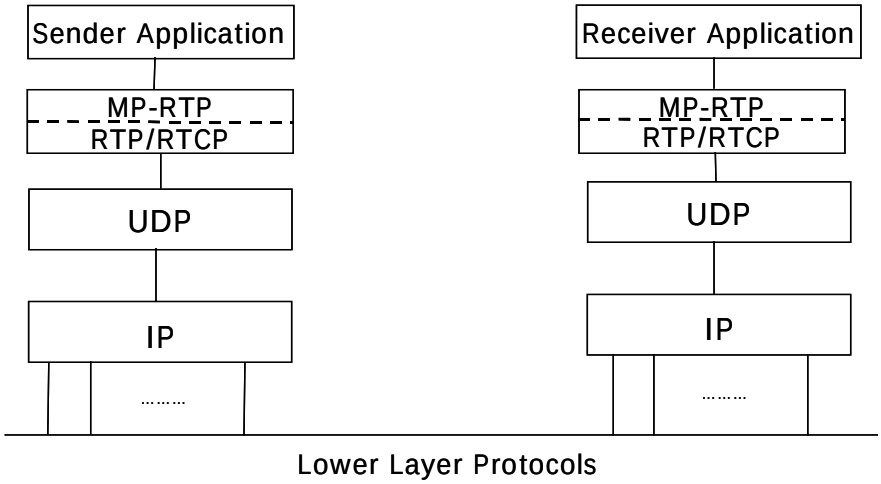


Figure 4.1 System architecture.

Also P-frames can be assigned varying reliability levels, since P-frames that are closer to the preceding I-frame are more valuable for preserving picture quality than later P-frames in the GOP. In addition, the sending application calculates the lifetime for each video frame N, $T_L(N)$, as follows:

$$T_L(N) = T_R(N) + D_S \tag{4.2}$$

where $T_R(N)$ is an estimate for the rendering time of frame N at the receiver, and D_S is a slack term to compensate the inaccuracies in estimating the *One Way Delay* (*OWD*) from the sender to the receiver, as will be discussed later, as well as the receiver's processing delay. Assuming that there is no compression and/or expansion of total display time at the receiver, the rendering time for frame N, $T_R(N)$, is calculated as follows:

$$T_R(N) = T_0 + T_D + N / R \tag{4.3}$$

where T_0 is the video session initiation time, T_D is the receiver playout delay, which determines the rendering time for the first frame in the sequence. Playout delay can be obtained from the receiver during the session initiation. R is the frame rate.

As the MP-RTP sub-layer receives a frame it fragments the frame, if required, into multiple packets, then RTP headers are added and the packets are sent to the receiver. In addition, a copy of each packet is kept in a retransmission buffer, along with its lifetime and priority. Typically, all the packets within one frame will have the same lifetime and priority. MP-RTP clears packets from the retransmission buffer, as it receives the real-time control protocol-receiver reports (RTCP-RR), which are sent regularly from the receiver, indicating the highest sequence number received, as well as other information regarding the quality of the received stream [10]. Initially, packets are sent on a primary path with the receiver, selected by the sender during the session initiation.

The MP-RTP at the receiver is responsible for sending retransmission requests to the sender as soon as it detects a missing packet. The format of the retransmission request, shown in Figure 4.2, is similar to RTCP-RR except that it is extended to include the 32 bits sequence number of the missing packet. As the retransmission request is susceptible to losses, the MP-RTP retransmits these reports on different paths to the sender.

MP-RTP uses heartbeat packets, shown in Fig. 4.3, to maintain an estimate for the *RTT* of the available paths. Each heartbeat packet includes a time stamp indicating the transmission time. The MP-RTP at the receiver responds to the heartbeat packet by sending a heartbeat-acknowledgment packet, shown in Figure 4.4, on the same path from which the heartbeat was received.

MP-RTP uses heartbeat packets, shown in Fig. 4.3, to maintain an estimate for the *RTT* of the available paths. Each heartbeat packet includes a time stamp indicating the transmission time.

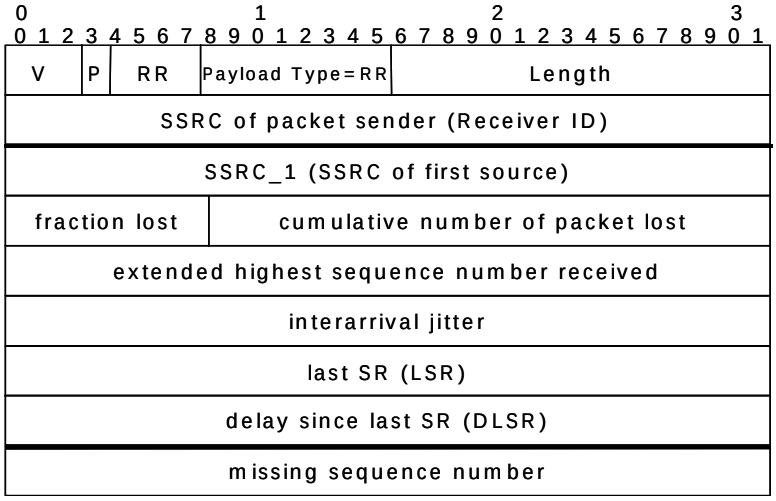


Figure 4.2 Extended RTCP-RR to include the missing sequence number.

The MP-RTP at the receiver responds to the heartbeat packet by sending a heartbeat-acknowledgment packet, shown in Figure 4.4, on the same path from which the heartbeat was received. The heartbeat-acknowledgement includes a copy of the timestamp in the corresponding heartbeat packet. The *RTT* estimates are calculated using a smoothed average of the current and previous measurements. These estimates are used to obtain an approximation for the paths *OWD* (i.e., $OWD \approx RTT / 2$). The application can compensate the inaccuracies in the *OWD* approximation as it assigns the frames lifetime, as shown in (2). In addition, MP-RTP uses the *RTT* estimates to switch the primary path, which can break which due to the mobility in the wireless network. To minimize the interruption for the interactive video session, as the primary path *RTT* increases beyond a certain threshold, MP-RTP sets the alternative path with the shortest *RTT* to be the primary path. The switching threshold can be based on the maximum delay allowed for the interactive video application.

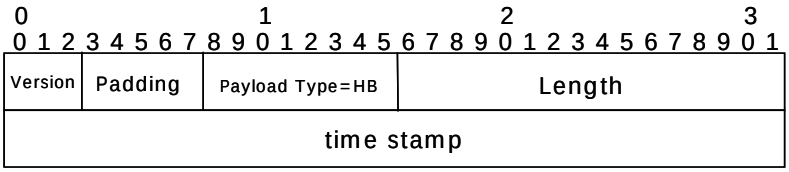


Figure 4.3 Heartbeat packet.

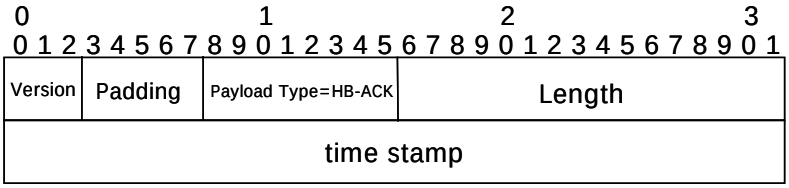


Figure 4.4 Heartbeat acknowledgement.

Currently, we are using a fixed value for the switching threshold. In future work, we are planning to investigate techniques to dynamically adapt the value of the switching threshold.

As soon as the sender receives a retransmission request, it performs the retransmission algorithm shown in Figure 4.5. By controlling the retransmission through the frames lifetime, as well as estimate(s) of the path(s) delay, MP-RTP prevents retransmission of expired packets while trying to meet the frames lifetime constraint.

If retransmitted the packet will not be received before the rendering time for the frame to which it belongs, the packet is discarded and the upper layer application is notified about the dropped packet to allow the encoder to utilize schemes, such as error tracking, that stops the prediction loop to limit the error propagation. Based on the information regarding the location of the lost packet, provided by the MP-RTP, the error tracking algorithm at the encoder reconstructs the resulting error distribution, in the next frame to be encoded, then it encodes the blocks contained in this region in intra-mode [9].

```
Let T_c be the current time at the sender,
     T_L(j) is the lifetime for frame j
     N is the number of paths between sender and receiver

if (lost packet belongs to high priority frame j)  {
    for all paths i                  where  path  i ε {1, N}
     if ( T_C +  OWD_i  < T_L(j) )
        Retransmit on path i

    if  (packet cannot be retransmitted)  {
         Discard packet.
         Notify the encoder with the location of the lost
         packet for error tracking.
   }
}
else if (lost packet belongs to low priority frame j) {
   if ( T_C +  min(OWD_i)  < T_L(j) )
        Retransmit on path i.
}
```

Figure 4.5 MP-RTP redundant retransmission algorithm.

By controlling the retransmission through the frames lifetime, as well as estimate(s) of the path(s) delay, MP-RTP prevents retransmission of expired packets while trying to meet the frames lifetime constraint. If retransmitted the packet will not be received before the rendering time for the frame to which it belongs, the packet is discarded and the upper layer application is notified about the dropped packet to allow the encoder to utilize schemes, such as error tracking, that stops the prediction loop to limit the error propagation. Based on the information regarding the location of the lost packet, provided by the MP-RTP, the error tracking algorithm at the encoder reconstructs the resulting error distribution, in the next frame to be encoded, then it encodes the blocks contained in this region in intra-mode [9].

4.3 Performance Analysis

We studied the performance of the MP-RTP mechanism via a *top-down* approach. First, we used a popular Markov link model [96], where lower layer detail is embodied in the bursty errors generated. This simple model enables us to examine the system performance over a wide range of pack loss rates and loss patterns. Next, lower layer details, including user mobility, multi-path routing, multi hop routes, and the MAC layer are taken into account in the OPNET simulations [89], which provide a more realistic view of the impact of these factors on the system performance.

A three-state Markov model was used for each link with the states representing a "good," "bad" or "down" status for the link [91]. The "down" state means the link is totally unavailable. The "good" state has a lower packet loss rate than the "bad" state. The packet loss rates we used are, $p_0 = 1.0$, $p_1 \in$ [0.1% 25%], and p2 = 0, and, for the "down," the "bad," and the "good" states, respectively. The transition parameters are chosen to generate loss traces with desired loss rates and mean burst lengths.

In our simulation, multiple paths were set up for each connection, and each path was continuously updated as follows: After every 2 s, two links were chosen randomly from a link pool to construct a new path.

Delay for channel i is modeled by an exponential distribution with the mean delay D_i = 30 ms. In addition, the radio channels are operating at 2.0 Mbps. We set the path maximum transfer unit (MTU) of 400 bytes for all the paths. The heartbeat interval is set to 150 ms.

To generate the video sequence used in our simulation, we used open source *XviD*

MPEG-4 compliant video codec [97]. We extended the source coder to implement the error-tracking algorithm described in [9]. The algorithm carries out error tracking with macro-block resolution rather than the pixel resolution. This reduced the computational burden as well as the memory requirements compared to the actual encoding of the video. Sixty seconds of a high motion video sequence (football match) are encoded at 15 frames per second (fps), which results in a sequence of 900 frames. The frame resolution is *Quarter Common Intermediate Format* (*QCIF*, 176 x 144 pixels/frame), which is the most common format at low bit rates, and the coding rate is 200 Kbps. We repeated our experiments with limited motion video sequence (TV news) and we get similar results to that shown here. We limited the playout delay at the receiver to 100 ms, to represent an interactive video application. We set the switching threshold, discussed in Section 4.3, to 300 ms. We selected this value because given the channel delays and the playout delay at the receiver, having the *RTT* of the primary path higher than this threshold will result in all frames arriving later than their rendering time at the receiver and will be discarded.

The average Peak Signal to Noise Ratio (PSNR) is used as a distortion measure of objective quality. PSNR is an indicator of picture quality that is derived from the root mean squared error (RMSE). The PSNR for a degraded $N_1 \times N_2$ image f' with respect to the original image f is computed as follows:

$$PSNR = 20 \log_{10} \frac{255}{\left(\frac{1}{N_1 N_2} \sum_{x=0}^{N_1-1} \sum_{y=0}^{N_2-1} \left[f(x,y) - f'(x,y) \right]^2 \right)^{1/2}} \tag{4.4}$$

Without transmission losses, the average PSNR of the decoded frames for the video sequence used in our performance study is 27 dB.

After obtaining a transmission trace of a video sequence, we run the decoder on the trace to measure the image distortion due to packet losses, using the PSNR. In order to generate statistically meaningful quality measures, for each simulation scenario we repeated the experiment ten times with different seeds. The presented PSNR values are the average of the ten experiments.

In our experiments, we set the application to choose I-frames and half of the P-frames starting from the I-frame in a GOP to be high priority frames, while other frames are set to low priority frames. As will be shown later in our analysis that this setting is a compromise between video quality and overhead.

Specific questions we looked at in our performance study: 1) How effective is our mechanism in maintaining the quality of the received video under different packet loss rates? 2) How the reconstructed video quality is affected by increasing the number of paths between the sender and receiver? 3) How much overhead is added by our mechanism compared to error control mechanisms that depend on controlling the intra-update rate? 4) How the reconstructed video quality is affected by changing the percent of P-frames within a GOP protected by redundant retransmissions? 5) How robust is the single layered video transported with MP-RTP compared to multilayer coded video transported over diverse paths? 6) How the MP-RTP mechanism will behave when considering the impact of lower layer components such as user mobility, multi-path routing, and the MAC layer.

4.3.1 Effect of Packet Loss Rate on Video Quality

We tested MP-RTP using two diverse paths between the sender and the receiver. The

average packet loss rates (PLR) p_l for path-0 and path-1 were set to 10 % and 5 % respectively. Path-0 was selected as the primary path during the video session initiation. We set the encoder so that the intra-update rate of 1/45 frames (i.e., one I-frame every 45 frames). Figure 4.6 shows the PSNR for each frame in the video sequence. For comparison we repeated the experiment using the retransmission scheme without redundant retransmissions, where the missing packets are retransmitted on the secondary path, which we refer to as Single Path RTP (SP-RTP). As can be shown from the figure that MP-RTP is able to maintain the video quality, at high packet loss rates. On the other hand, with the single path retransmission scheme, the video quality can be dropped for long durations due to loss of packets in reference frames. Although the sender can keep re-transmitting the packet, the receiver will discard these retransmissions, as they arrive after the frame rendering time.

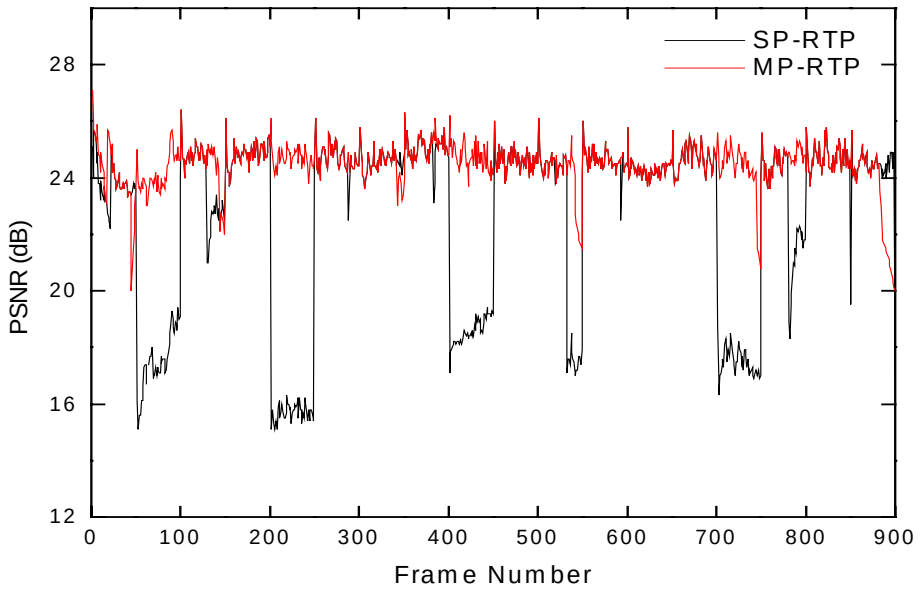


Figure 4.6 PSNRs of received frames.

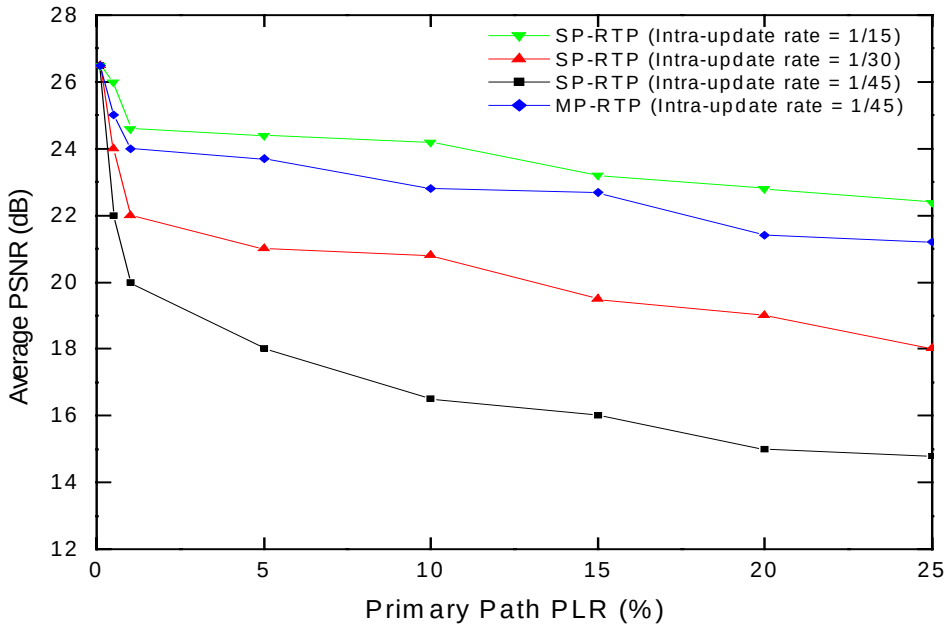


Figure 4.7 Average PSNR versus average packet loss rate.

We repeated the same experiment for different Intra-update rates. For MP-RTP, we used an intra-update rate of 1/45 frames. Figure 4.7 shows that the SP-RTP scheme achieves a similar performance to MP-RTP only when the I-frame frequency is increased more than three times to 1/15 frames. As the I-frames have larger sizes than P- and B-frames, reducing the I-frame update period will consume unnecessary bandwidth even when the loss rate is low.

If the intra-update rate is set to 1/45 frames for the single path case, it can be seen that the quality deteriorates rapidly. Again this is mostly due to losses in reference frames, as a result of the high packet loss rate and the bounded delay for interactive video. The errors are propagated from reference frames to the following frames up to the next I-frame. On the other hand, redundant retransmissions over diverse paths ensures that in the single retransmission allowed at least one copy of the packet will be received, preventing the error propagation.

4.3.2 Effect of Changing the Number of Paths

We tested MP-RTP with different number of paths between the sender and receiver. We varied the average packet loss rate on the primary path from 0.1 % to 20 %. We chose the packet loss rates for the other three secondary paths equal 1%, 10% and 20% respectively. As can be seen from Figure 4.8 that with a single path the quality deteriorate at high packet loss rates, due to error propagation. With increasing the number of paths between the sender and the receiver, and due to the independent loss characteristics of the paths, the probability that the retransmitted packets will be received before their deadline increases. Also it can be seen that there is a slight enhancement in the video quality as we increase the number of paths more than 3. This means that the independent loss characteristics of 3 paths is enough to ensure that the at least one copy of the retransmitted packet will be received. This suggests that the sender can limit the number of copies to 3, which limits the overhead of the mechanism.

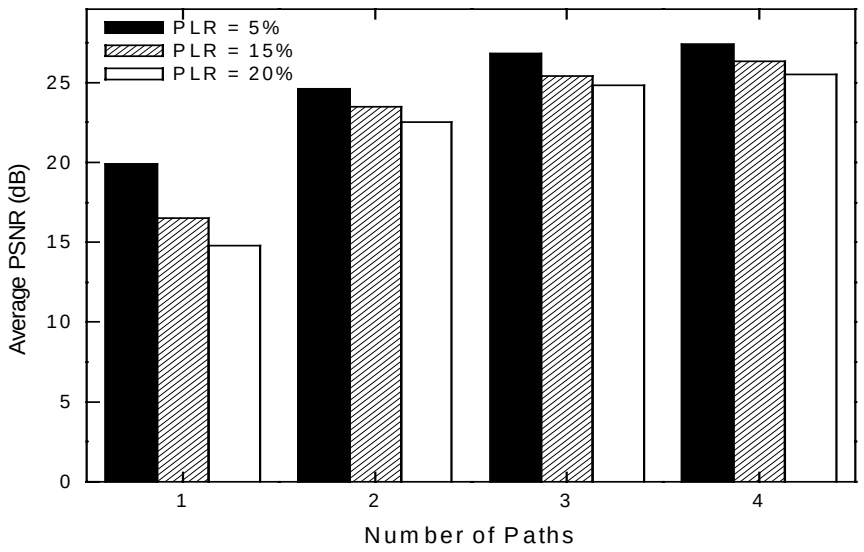


Figure 4.8 Average PSNR versus number of paths.

4.3.3 Redundant Retransmission Overhead

We compared the overhead of MP-RTP, due to the redundant retransmissions, retransmission requests and heartbeat packets, to the overhead of an error control mechanism that depends on controlling the intra-update rate to limit the error propagation. We define the overhead ratio to be the total number of bytes sent in Intra-update rate scheme to the total number of bytes sent in MP-RTP, to attain a given video quality represented by the average PSNR. In order to calculate the maximum overhead for MP-RTP, we used 3 paths. We varied the average packet loss rate for the primary path, while the packet loss rates for the other paths were set to 5 % and 10 % respectively.

Figure 4.9 shows the overhead ratio for average PSNR equal to 23 dB. As was shown before, the single path retransmission case required an intra-update rate of 1/45 frames, to attain a video quality of around 23 dB, while MP-RTP required an intra-update rate of 1/15 frames, to achieve the same quality. It can be seen from the figure that the overhead of MP-RTP is less than that for the intra-update scheme.

The reason behind this that the redundant retransmission mechanism implemented in MP-RTP is adaptive, in the sense that it only adds the retransmission overhead when there is loss in the video stream. In addition, the degree of the overhead is proportional to the importance of the lost packets. Although heartbeat packets are periodically sent, they have less contribution to the overhead, as they are small in size compared to the size of video frames.

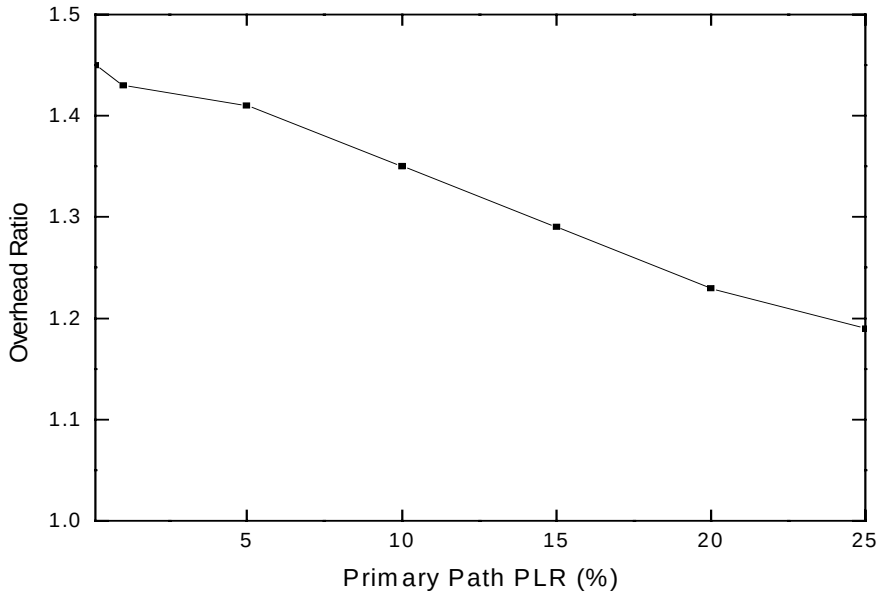


Figure 4.9 Overhead ratio versus primary path packet loss rate.

4.3.4 Effect of Changing the Reliability Level for P-frames

As P-frames that are closer to the I-frame in a GOP have more effect on the received video quality than farther frames, we examined the effect of varying the priority level of the P-frames, starting from the I-frame in the GOP up to the next I-frame, on the received video quality under different average packet loss rates.

We used an intra-update rate of 1/45 frames, and two paths. We varied the average packet loss rate for the primary path, while we set the path average packet loss rate for the other path to 10 %. As can be seen from Figure 4.10 that the average quality will keep increasing as we protect more P-frames within the GOP, but the increase rate become slower after around 50 %. This is because any loss within the following P-frames will only propagate up to the next I-frame. These results also confirm the analytical model presented in [98]. This suggests protecting up to 50% of the P-frames within the GOP through redundant retransmission, as protecting more P-frames come with a price of increasing the transmission overhead.

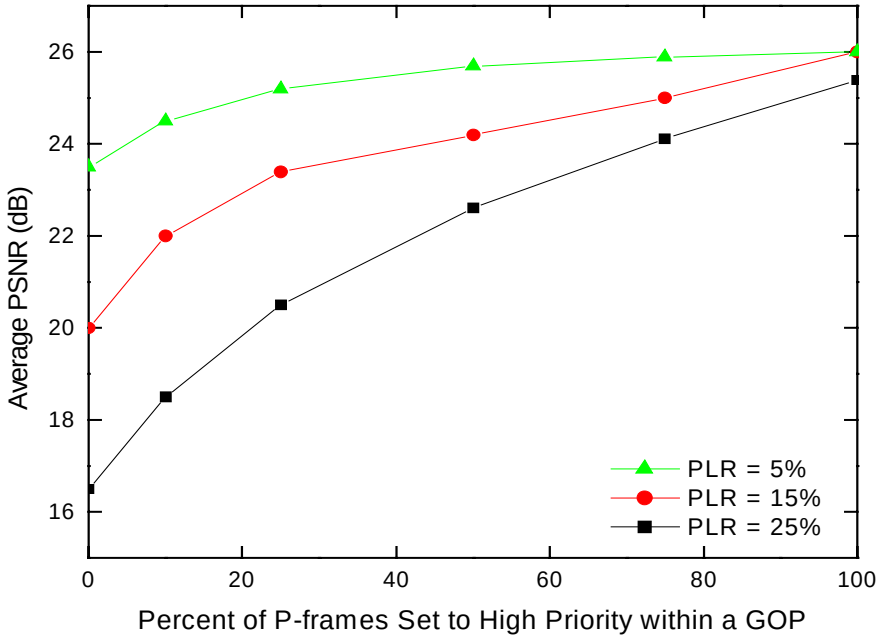


Figure 4.10 Average PSNR versus the percent of P-frames set to high priority frames.

4.3.5 Comparison between Single Layer Coded Video with MP-RTP and Layered Coded Video with Multi-Path Transport

In this experiment we compare MP-RTP with the error control mechanism, proposed in [71], that combines Layered Coding (LC) with Multi-Path Transport (MPT). The mechanism protects the base layer (BL) packets through single path retransmission. We used a layered coder that generates a BL and a single enhancement layer (EL). The BL is transmitted on primary path, while the EL is transmitted on secondary. Packets lost from the BL are retransmitted on secondary path, while packets lost from the enhancement layer are not retransmitted. The channel packet loss rate for secondary path is set to 10 %, while we varied the channel packet loss rate for the primary path. To generate the layered video we used an H.263+ public domain codec [99]. We followed the SNR profile in H.263 when generating the layers.

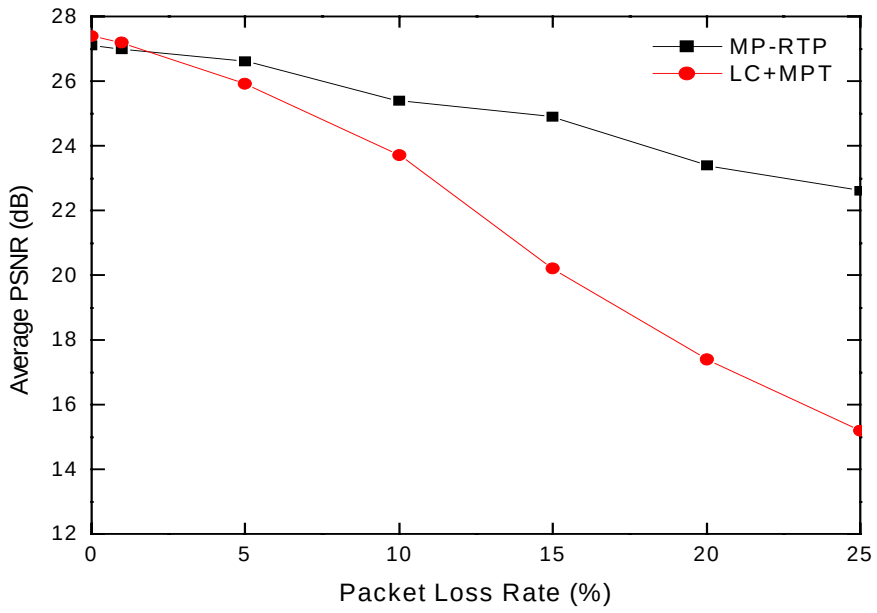


Figure 4.11 Comparison between single layer coded video with MP-RTP and layered coded video with multi-path transport.

Figure 4.11 shows the average PSNR versus the primary path packet loss rate. We repeated the experiment with the same video sequence coded with a single layer coder and transported with MP-RTP. We used the same network setup used for the layered video. The figure shows that under high loss rate the quality of the layered video dropped more quickly than the single layer video. The reason that the quality of the layered coded video depends on the conditions of the path that carries the base-layer packets. In addition, retransmitted packets may still be lost, due to the variable characteristics of the retransmission path. As far as the BL packets are lost, EL packets are discarded, even if they are received without errors. This causes the video quality to fluctuate with the error characteristics of the BL path. Thus, for interactive video, depending on single path retransmission to protect the BL packets is not enough to maintain the video quality under high loss rates. On the other hand, the redundant retransmission mechanism in MP-

RTP, which benefits from the independent characteristics of the diverse paths between the sender and receiver, succeeds in maintaining the video quality regardless the packet loss rate.

4.3.6 Performance of MP-RTP in Multi-Hop Ad Hoc Networks

The simplicity of the Markov model enables us to examine the performance of the proposed schemes over a wide range of packet loss patterns. In this section, we use the OPNET models to examine the impact of lower layer factors, such as MAC operation and user mobility affect video transport, which are not revealed by the Markov model [100].

DSR is a source routing protocol proposed for mobile ad hoc networks [92], where intermediate nodes do not need to maintain up-to-date routing information for forwarding a transit packet since the packet carries the end-to-end path in its header. It is an on-demand protocol where route discovery is performed for a node only when there is data to be sent to that node. There are a number of extensions of DSR to multi-path routing [91][101][102]. We chose to extend DSR to multi-path DSR (MDSR) as described in [91]. Each node maintains two routes to a destination. We allow both the destination node and intermediate nodes to reply to a route query. When the destination node replies, it also copies the existing routes from its own route cache into the route reply, in addition to the route that the route query traversed. This algorithm is a greedy algorithm in the sense that it always finds the best paths returned *so far*.

We only experimented with the two-path version, since the results in [102] indicate that the largest improvement is achieved by going from one to two or three paths. The MDSR models is built based on the OPNET DSR model [103].

Using the OPNET model, we simulate an ad hoc network with 16 nodes in a 600 m x 600 m region. Given the dimensions of the region, 16 nodes result in a density that maintains a connected network for most of the time [104]. Each node is randomly placed in the region initially. As in [91], we used a version of the popular *random waypoint* mobility model, where each node first chooses a random destination in the region, then moves toward it at a *constant* speed. When it reaches the destination, it pauses for a constant time interval, chooses another destination randomly, and then moves toward the new destination [105]. Note that this is a simplified version of the Random Waypoint model. Since there is no randomness in the nodal speed, the convergence problem reported in [106] does not present itself here. We used a pause time of 1.0 s for all the experiments reported in this paper. The speed of the nodes varies from 0 m/s to 10 m/s, which models movement of pedestrians or vehicles in city streets.

We use the IEEE 802.11 protocol in the MAC layer working in the DCF mode. Its physical layer features, e.g., frequency hopping (FH), are not modeled. The channel has a bandwidth of 1 Mb/s.

The transmission range is 250 m. If the sender of a packet is within this range of the receiver, and the sender has successfully accessed the channel during the transmission period, the packet is regarded as correctly received. The maximum number of link layer retransmissions is seven, after which the packet is dropped.

Among the 16 nodes, one is randomly chosen as the video source and another node is chosen as the video sink, where a 5 s playback buffer is used to absorb the jitter in received packets. The video source starts a session using two routes, sending encoded

video at 200 kb/s to the sink. All other nodes generate background traffic to send to a randomly chosen destination. The interarrival time of the background packets is exponentially distributed with a mean of 0.2 s. The background packet has a constant length of 512 bits.

We examined the impact of node mobility on the video transport using single path transport and MP-RTP. Figure 4.12 shows the PSNR of the received video when the nodes are moving with speed 10m/s, and using single path transport with 1/45 intra-frame update rate. The PSNR curve shows large drops. We conjecture that during these periods the source node and destination node were either far away from each other, or were in a hot-spot. Compared with Figure 4.6, it is clear that mobility has a negative effect on video.

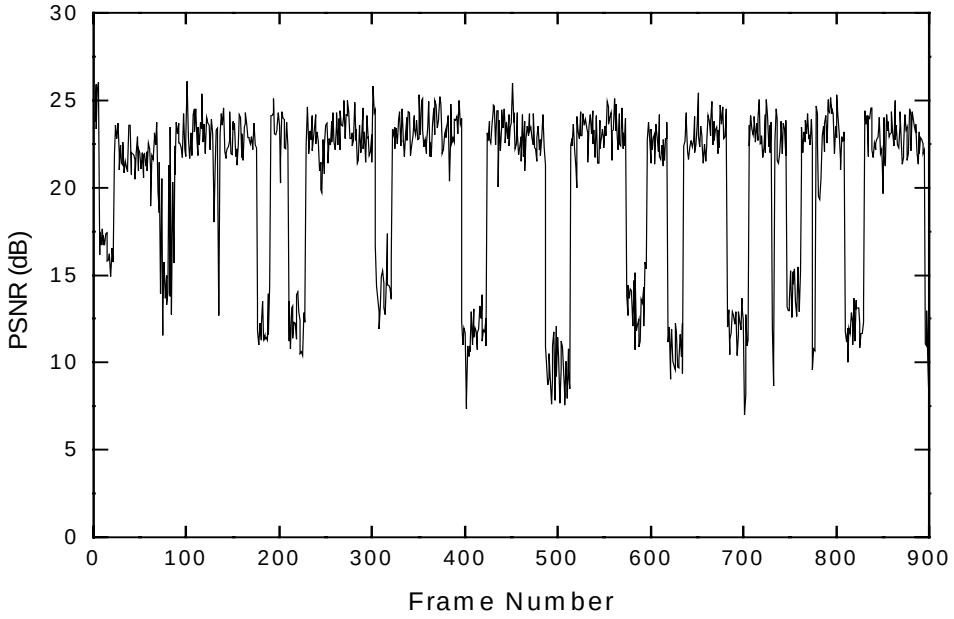


Figure 4.12 PSNRs of the received frames with single path transport. 16 nodes in 600 m x 600 m region at a speed of 10 m/s.

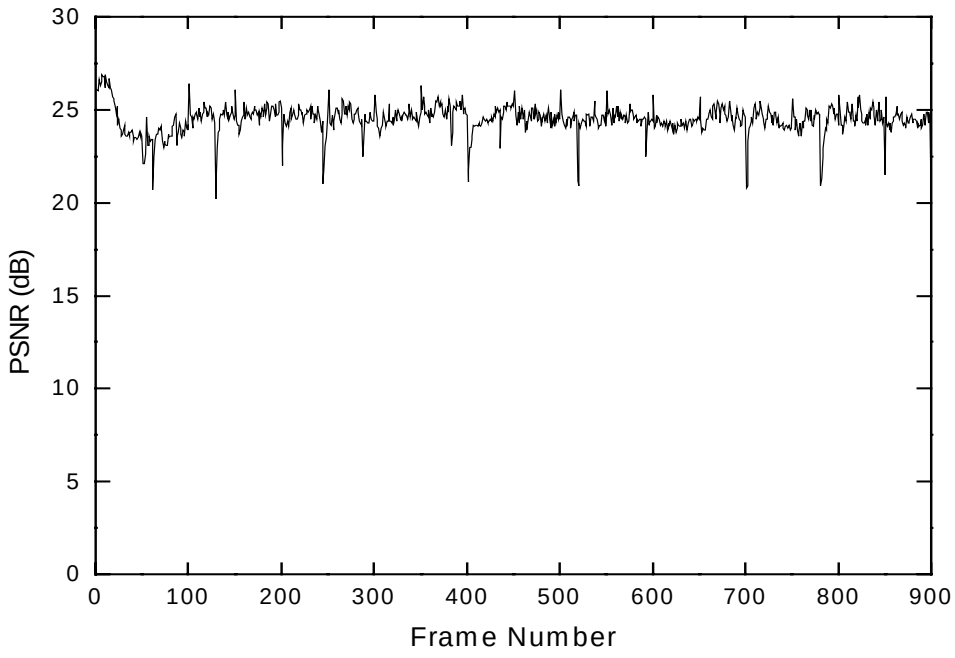


Figure 4.13 PSNRs of the received frames with MP-RTP. 16 nodes in a 600m x 600m region at a speed of 10 m/s.

Figure 4.13 shows the PSNR for the received frames transported vide MP-RTP. Again the nodes are moving with speed 10 m/s. The PSNR curve is very stable, with only a few narrow drops.

Figure 4.14 is the resulting average PSNR for different speeds using the MP-RTP. The figure shows that during the initial increase in mobility, routes break down more easily, which leads to an increase in the mean packet loss rate and a drop in the PSNR. As speed further increases, the average PSNR becomes stable, as new topologies are more quickly formed and new routes are more quickly established. A hot spot in the region, where nodes cluster and compete for the channel, is more quickly dispersed. As speed increases, the period of time a node remains disconnected is smaller. The turning point (4 m/s in Figure 4.14) is determined by the node density in the region and the transmission range.

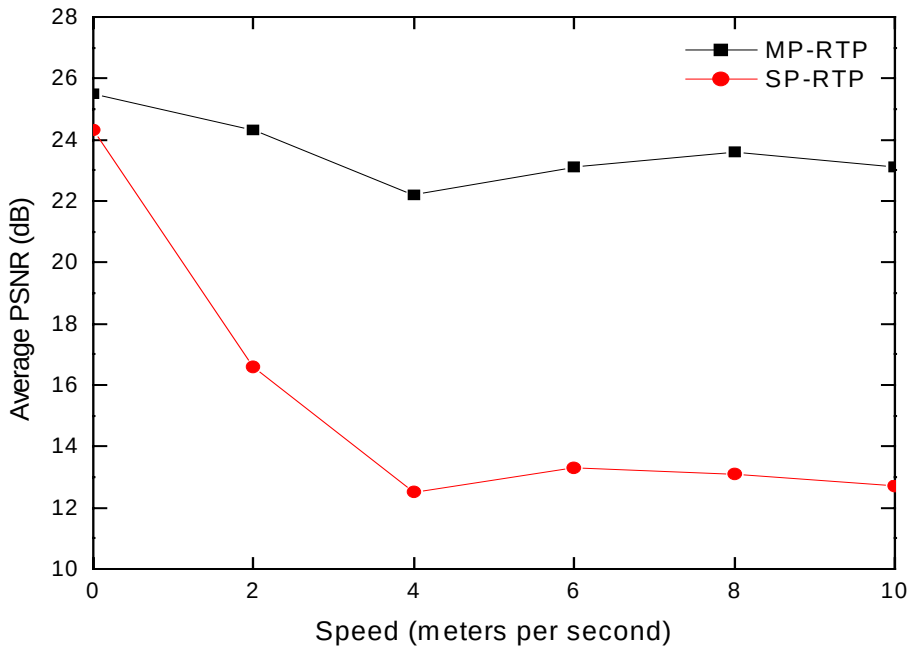


Figure 4.14 Average PSNR versus node speed.

These observations match the observation, described in [107] and [91], that mobility is both harmful and helpful.

We conjecture that similar phenomenon exists for other scenarios with a different number of nodes or a different transmission range, given that the node density is high enough to maintain a connected network for most of the time.

When the nodal speed increases even further, the routing process would be unable to track the quickly changing topology. Therefore, drops in the average PSNR is expected. The figure also shows that MP-RTP keeps the PSNR for the received video higher than for the single path transport and layered coding with MPT, as the likelihood that that both paths experience packet losses simultaneously is quite high.

Figure 4.15 shows the transmission overhead MP-RTP and SP-RTP as we change the nodes speed. We measure the transmission overhead as the number of packets including

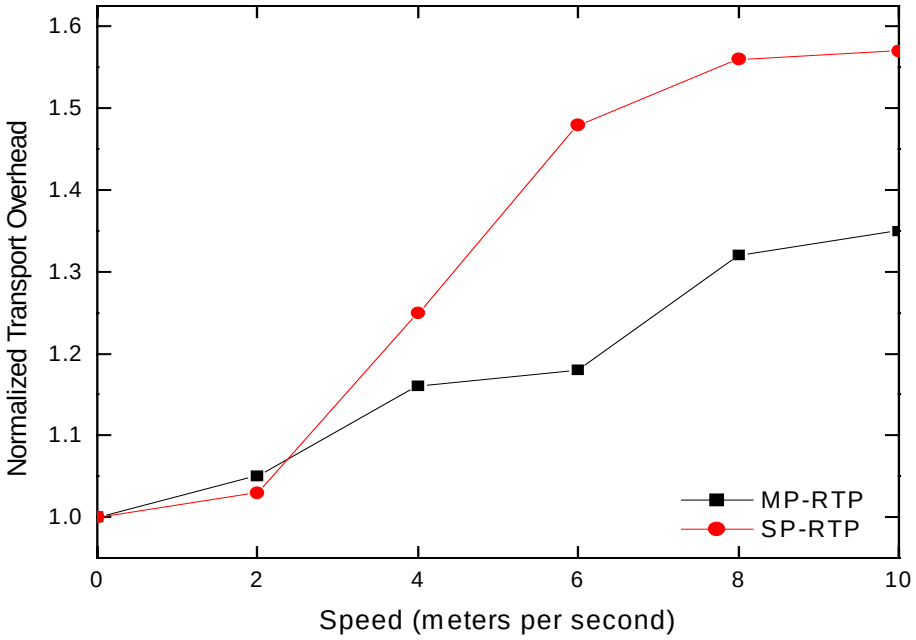


Figure 4.15 Normalized transport overhead for MP-RTP and SPT at different node speeds.

the routing protocol packets, MP-RTP heartbeats and redundant transmission overhead.

At each speed we repeated the SP-RTP experiments with different intra-frame update rates to achieve an average PSNR close to that achieved by MP-RTP at the same node speed and with 1/45 intra-frame update. At low speeds, SP-RTP provides lower transmission overhead due to the overhead of MP-RTP in setting the multiple paths at the network layer.

However, as we increased the node speeds, the overhead required by the SP-RTP to maintain the average PSNR through decreasing the distance between the intra-frames exceed the overhead of the routing protocol to maintain the routes that continuously brake due to mobility. Again, the reason behind this that MP-RTP only adds the retransmission overhead when there is loss in the video stream. On the other hand, intra-

frames are large in size, thus decreasing the distance between intra-frames will require a high bit rate.

4.4 Summary

In this chapter, we presented an error control mechanism for interactive video in mobile wireless networks. The nature of video encoded using motion compensation requires higher protection for reference frames that dependent frames, otherwise errors due to packet losses in reference frames propagate to dependent frames. Interactive video complicates the problem by bounding the time available for the error control. To tackle these problems, our mechanism provides unequal error protection to data within the video stream according to their importance to the reconstructed video quality. We realized the unequal error protection through extending the classic retransmission based error control, with redundant retransmissions, where the sender retransmits multiple copies of the packet. The number of copies depends on the reliability level required for the data within the packet. In order to increase the probability that at least one of the retransmitted packets arrive at the receiver, in less number of retransmissions, we propose to send the redundant retransmissions on diverse paths. A delayed constrained retransmission, based on the packet lifetime and estimate of the delay from the sender to receiver, is used to prevent re-transmitting expired packets. We implemented the proposed mechanism as an extension to RTP, refereed to as Multi Path - RTP (MP-RTP). Performance results show that the mechanism is able to provide a better quality for interactive video under different packet loss rates and mobility speeds, than single path transport mechanism. In addition, MP-RTP is shown to be more robust than error control schemes that combine layered coding with the path diversity in ad-hoc networks.

Chapter 5

Adaptive Real-time Video Streaming over AIMD Transport Protocols

In this chapter, we present adaptation strategies for real-time video streams over *Additive-Increase Multiplicative-Decrease* (*AIMD*) transport protocols, such as TCP or SCTP, where we switch among several versions of the coded video to match the available network bandwidth accurately, and meet client delay constraints. By monitoring the application buffer at the server, we estimate the current and future server buffer drain delay, and derive the transmission rate to minimize client buffer starvation. We also show that the adaptation accuracy can be significantly improved by a simple scaling to transport protocol send-buffer size. The proposed mechanisms were evaluated through simulation and testbed implementation with real Internet traces. Performance results show that the adaptation mechanism is responsive to bandwidth fluctuations, while ensuring that the client buffer does not underflow, and that the quality adaptation is smooth so that the impact on the perceptual quality at the client is minimal.

5.1 Introduction

Transmitting real-time video across a heterogeneous best effort network such as the Internet, while achieving acceptable perceptual quality at the client, is still a major challenge. There has been significant interest in developing adaptive streaming media mechanisms for best-effort networks for many reasons, (i) The cost of a non-QoS connections is low; (ii) Although some networks provide the option for a QoS connection, most networks today have no such provision; and (iii) Due to the long connection duration of a streaming session, performing admission control only at the beginning of a session, may lead to high call-rejection rates or unnecessarily poor media quality.

While the majority of traffic on the Internet today is comprised of TCP flows, conventional wisdom holds that TCP is unsuitable for "real-time" traffic due to its lack of throughput guarantees and insistence on reliability [11]. For these reasons, there have been many proposals for new transport protocols for the purpose of solving the video transport problem over the Internet. These protocols need to be TCP-friendly to ensure that they will not cause network collapse. However, proving a new transport protocol to be TCP-friendly can be difficult, because the dynamics of TCP congestion control is extremely complex [12]. Thus an advantage of using SCTP in transporting video is that the transport is ensured to be friendly to other flows sharing the same network, as it has a similar congestion to TCP congestion control [27], also to take advantage of its many useful features for multimedia transport, illustrated in Section 2.1.2. Also in many situations streaming over TCP/SCTP is unavoidable, such as when the clients are located behind network firewalls permitting only inbound HTTP traffic.

The SCTP flow of an application experiences rate variations for two distinct reasons -- the first being the flow's own congestion control behavior, i.e., the window-based congestion control algorithm of SCTP introduces saw-tooth fluctuation in the streaming rate, and the second being competing traffic in the network. Client-side buffers can be used for smoothing out the saw-tooth fluctuation of a TCP flow [11]. However, despite any amount of buffering, competing traffic can have persistent effects on the streaming rate, and consequently on the viewing quality. The problem is more challenging in the case of real-time video since client buffering is limited by end-to-end latency limit, and also data cannot be perfected into the client buffer when extra bandwidth is available. Thus streaming video applications must deal with persistent rate changes, before the client-side buffers are overwhelmed. The usual way is to employ quality-adaptation control, adjusting the basic quality-rate trade off of the video.

The primary design goal of quality-adaptation control mechanisms is to adapt the outgoing video stream so that, in times of network congestion, less video data is sent into the network and consequently fewer packets are lost and fewer frames are discarded. This rests on the underlying assumption that the smooth and timely play out of consecutive frames is central to a human observer's perception of video quality. Although a decrease in the video bit rate noticeably produces images of coarser resolution, it is not nearly as detrimental to the perceived video quality as inconsistent, start-stop play out. By switching between different quality levels during the stream, the mechanism makes a fundamental trade-off by increasing the video compression in an effort to preserve a consistent frame rate at the client.

Quality adaptation mechanisms for stored video prefetch future portions of the video stream into client storage when the available bandwidth exceeds the consumption rate. In real-time video perfetching is not possible, and the server application transmits the video stream at the consumption rate, unless the available bandwidth is less than the consumption, at which time the stream is transmitted at the available bandwidth rate. The amount of prefetched data in the client application buffer never increases after time t = *D*, where *D* is the initial playout delay. If the prefetch buffer becomes empty, the client can partially receive the stream by reading directly from the network while incurring some loss of data. High fractions of lost data occur if the scheme fails to follow the variations in available bandwidth.

In this chapter we focus on sender-driven quality adaptation, for real-time video streams, to minimize any overheads at the client. In particular, we focus on quality adaptation using stream switching, as it has been shown to provide better viewing quality than adding/dropping layers, due to the layering overhead [13]. First, we introduce an adaptive stream switching mechanism for real-time video that does not require either modifications to the network transport protocol at the sender or at the receiver, or support from the network infrastructure. By monitoring the application buffer occupancy, the mechanism detects the network bandwidth variations and estimates the current and future server buffer drain delay, and accordingly it adapts the video transmission rate to minimize the client buffer starvation while ensuring that the adaptation affects the perceptual quality at the client minimally. Then, we show that by scaling the TCP *send-buffer* according to the available bandwidth-delay product, the adaptation accuracy can be

improved significantly. Our results show that the *send-buffer* scaling has minimum effect on the SCTP throughput, while reducing the latency of packets in the SCTP buffer.

The rest of this chapter is organized as follows. In Section 5.2, we describe our system architecture. Section 5.3 describes the adaptation mechanisms, including the switching down and switching up strategies and the SCTP *send-buffer* scaling to improve the adaptation accuracy. In Section 5.4, we evaluate the performance for our proposed mechanisms in terms of the number of quality changes as well as buffer underflow events at the client. Section 5.5 presents the implementation and performance evaluation of the bandwidth adaptation mechanisms in a network testbed. We conclude this chapter with a summary in Section 5.6.

5.2 System Architecture

In our architecture, shown in Figure 5.1, real-time media is encoded into multiple quality streams[1], which are fed to the adaptive media server [108]. The server selects one of these streams $R^{in}(t)$ and injects it into the server buffer. The server buffer is drained as fast as the network connection permits, i.e., $R^{out}(t)$. The network output is fed into the client buffer. To smooth out short time scale bandwidth variations and to remove jitter, D units of time of the stream are allowed to build up in the client prefetch buffer before playback begins. The size of this buffer is limited by the maximum end-to-end latency constraints of the system. In addition, without loss of generality, we assume that the streams are CBR encoded.

[1] These streams may be independent encodings of the media at different bit-rates, resolutions, frame rates etc. Furthermore, additional streams may be derived from each independent encoding by discarding parts of the stream. For instance, from a stream with a GOP structure, IBPBP… we can derive two additional sub-streams corresponding to lower frame-rates by: a) dropping all B and P frames b) dropping all B frames.

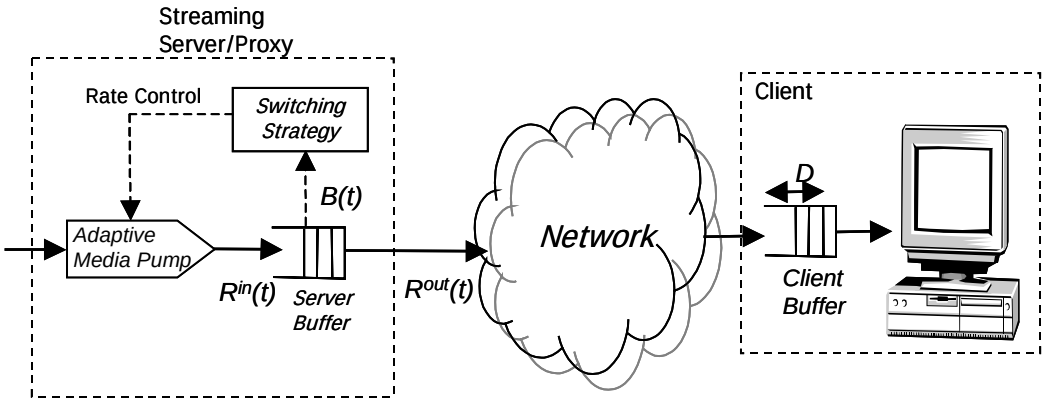


Figure 5.1 Streaming system architecture.

In the following sections, we present adaptation algorithms that enable the server to stream the media across varying network bandwidth conditions while maximizing the video quality at the client. We measure quality in terms of the achieved average bit-rate, variations in the quality at the client, and data loss due to prefetch buffer starvation which can result in frame drops or pauses in the video to allow for rebuffering. Hence, we want to track the available bandwidth faithfully, while minimizing prefetch buffer underflow and maintaining a relatively constant quality by minimizing the number of stream switches.

5.3 Stream Switching Strategies

Estimation of network bandwidth in the case of SCTP-based streaming is not straight forward since SCTP hides the network congestion status from the application. A somewhat delayed effect can, however, be seen in the application buffer. If the server is streaming at a certain rate and the network capacity goes below this rate, this will be reflected as increase in the server buffer occupancy. Similarly, if the server is streaming with a rate lower than the current network capacity, the server buffer will start to empty.

The proposed mechanism monitors the application buffer at the server in order to estimate the current available bandwidth in the network, and accordingly it adapts the streaming rate. It reacts to the network congestion state so as to prevent client buffer underflow (the client prefetch buffer and the server buffer are mirror images of each other), in addition it tries to keep the number of quality changes at the client to a minimum so as to have minimal effect on the user perceived quality. We introduce below the criteria used for stepping up or down the streaming rate. The algorithm is described in the context of a real-time stream which the server receives from a live-source and forwards to the client over a SCTP connection.

Consider that we have N available video streams (different encodings or derived sub-streams) with corresponding bit-rates V_j [2] ($j = 1,2,\ldots,N$) and we make the decision to switch at discrete time instances t_k. We present different strategies to make this decision:

5.3.1 Switching Down Strategies

We first present strategies to switch down the streaming rate based on observed reduction in the available network bandwidth.

5.3.1.1 Instantaneous Decision Strategy

The minimum delay that any incoming packet (at time t_k^+) experiences with the server buffer at measured fullness $B(t_k)$ is the time required to drain the buffer. This delay $\Delta_{k^+}^{\min}$ may be estimated as $\Delta_{k^+}^{\min} = \dfrac{B(t_k)}{R^{out}(t_k)}$, where $R^{out}(t_k)$ is the output rate estimated

[2] In this discussion we have assumed that the stream bit-rates do not change with time. This is true when the encoder has a good rate control, and all the data from the encoder reaches the server. When these assumptions are violated, we need to estimate these stream bit-rates.

at time instant t_k. $R^{out}(t_k)$ can be obtained using a Weighted Exponential Moving Average (WEMA) of the past and current bandwidth observations at the server. In order to satisfy the client delay constraints and prevent the client buffer from underflowing (server buffer from overflowing), we should have $\Delta_{k^+}^{\min} < D$. Hence, whenever we observe $\Delta_{k^+}^{\min} > \alpha D$, we reduce the input rate to the largest available rate smaller than $R^{out}(t_k)$, so that we drain the server buffer build-up and preempt any overflow. The conservative factor α ($0 < \alpha < 1$), is introduced in order to account for possible variations in the input and output rates during sampling interval $[t_k, t_{k+1})$. Hence, for this interval, we select the input rate $\overline{R}_k^{in}$ as:

$$\overline{R}_k^{in} = \max_{\substack{j=1,\ldots,N \\ V_j < R^{out}(t_k)}} \{V_j\} \tag{5.1}$$

where V_j are the N available video bit-rates. The factor α should be selected based on the expected variations in the rates.

This decision strategy aggressively reduces the input rate whenever it estimates buffer drain time as being greater than the computed threshold. Such a strategy follows any reductions in the available output bandwidth rapidly, however, as it does not take into consideration the rate of change of the buffer, it can lead to unnecessary reductions in the input rate. For instance, even if the buffer fullness was being steadily reduced (based on a previous switching decision), this strategy could further reduce the input rate. This can lead to under-utilization of available bandwidth, and undesirable quality for the user.

5.3.1.2 Look-Ahead Decision Strategy

In this strategy, in addition to the measured current server buffer fullness, we estimate the buffer fullness one sampling interval in the future before we make our decision. The server buffer at sampling instant t_{k+1} may be estimated as:

$$\hat{B}(t_{k+1}) = \max\left\{\left[B(t_k) + \int_{t_k}^{t_{k+1}} \left(R^{in}(u) - R^{out}(u)\right)du \right], 0\right\} \tag{5.2}$$

where $R^{in}(u)$ and $R^{out}(u)$ are the instantaneous input and output rates. If the sampling intervals are chosen small enough, we can assume that the input and output rates are constant over the entire interval[3]. Hence we may rewrite equation (5.1) as

$$\hat{B}(t_{k+1}) = \max\left\{\left[B(t_k) + \left(\overline{R}_k^{in} - \overline{R}_k^{out}\right)(t_{k+1} - t_k)\right], 0\right\} \tag{5.3}$$

where $\overline{R}_k^{in}$ and $\overline{R}_k^{out}\left(= R^{out}(t_k)\right)$ are the constant input and output rates interval $[t_k, t_{k+1})$. We then estimate the delay $\Delta^{\min}_{(k+1)^-}$ that would be experienced at the end of this interval (before time instant t_{k+1}) as $\Delta^{\min}_{(k+1)^-} = \dfrac{\hat{B}(t_{k+1})}{\overline{R}_k^{out}}$. Since during this sampling interval we want to avoid server buffer overflow, we would like to constrain $\Delta^{\min}_{(k+1)^-} \le \beta D$, where β $(0 < \beta < 1)$ is another conservative factor introduced to account for possible variations in the input and output rates. Combining with equation (5.3) we can easily determine that:

$$\overline{R}_k^{in} \le \frac{\beta D \overline{R}_k^{out} - B(t_k)}{(t_{k+1} - t_k)} + \overline{R}_k^{out} \tag{5.4}$$

[3] While this is not true at all instants, on average this is a reasonable assumption.

We may thus use equation (5.4) to determine what input rate to switch to such that we avoid server buffer overflow, at the end of the current interval, as a result of the decision. While the instantaneous decision is like a zero-th order control system, this is more like a first order control system. This strategy can avoid unnecessarily aggressive reductions and stream switches (thereby improving the visual quality) in the input rate by sometimes borrowing from, and sometimes provisioning for the future. However, it also makes assumptions that the output rate does not change significantly over the interval $[t_k, t_{k+1})$. Hence, when the timescale of network variations is smaller than the sampling interval (i.e. the network conditions change rapidly) the instantaneous decision is likely to outperform the look-ahead decision, and conversely if the sampling interval is smaller than the timescale of network variations, the look-ahead decision is likely to outperform the instantaneous decision.

5.3.1.3 Combined Decision Strategy

We may combine the benefits of these two decision strategies to minimize the number of stream switches (for better visual quality) while following the available bandwidth accurately, and satisfying the user delay constraints. The combined algorithm for the decision strategy to switch down is shown in Figure 5.2.

5.3.2 Switching Up Decision Strategy

While we attempt to switch down the server streaming rate as soon as we observe a low network bandwidth, we cannot similarly switch up the streaming rate.

1. Compute buffer fullness $B(t_k)$ and estimate $R^{out}(t_k)$

   Set $\overline{R}_k^{in} = \overline{R}_{k-1}^{in}$.

2. Estimate $\Delta_{k^+}^{\min}$ and $\Delta_{(k+1)^-}^{\min}$

3. If $\Delta_{k^+}^{\min} > \alpha D$ and $\Delta_{(k+1)^+}^{\min} > \beta D$

   Determine upper bound on input rate

   $$R_{\max}^{in} = \max\left\{\left(\frac{\beta D \overline{R}_k^{out} - B(t_k)}{(t_{k+1} - t_k)} + \overline{R}_k^{out}\right), \overline{R}_k^{out}\right\}$$

   Select $\overline{R}_k^{in} = \max_{\substack{j=1,\ldots,N \\ V_j < R_{\max}^{in}}} \{V_j\}$

   Else

   Test whether we can switch up

4. Wait until next sampling interval. Goto Step 1.

Figure 5.2 Combined switching down decision strategy.

This is because switching up too rapidly can actually create congestion in the network and thereby lead to oscillations between switching up and switching down, adversely affecting the video quality. Hence, it might be suitable for the server to attempt to stream at a higher rate, only after a certain duration during which the server does not observe a congestion event.

The problem is that there is no explicit signal indicating when the server should switch up.

For this reason, our mechanism carries out active experiments by probing the network to ensure that there is enough capacity for the next higher streaming rate. We call these experiments, *switch-experiments*. The *switch experiment* is triggered whenever the server does not experience a congestion event for an interval T_E^i referred to as the *Inter-Experiment timer.*

The experiment is performed by switching to the next higher available streaming rate, such that:

$$\overline{R}_k^{in} = \min_{\substack{j=1,\ldots,N \\ V_j > \overline{R}_{k-1}^{in}}} \{V_j\} \tag{5.5}$$

Each experiment lasts for a maximum duration of T_S . During the switching experiments, the congestion monitor in the server continues to monitor the network and if no congestion is caused due to the experiment, this is considered an indication that the network can support the next higher bit rate stream. In this case, the server stays at the higher stream. However, if congestion is detected, as indicated in Step 3 of the combined switch down algorithm, the sender reverts to the lower rate. The sender also learns from the failed *switch experiment*, by exponentially backing off the *Inter-Experiment timer* T_E^i for this rate, before retrying the experiment. Backing off the timer is likely to reduce the number of rate switches at a time when the available bandwidth in the network cannot support higher streaming rate. The exponential back-off is performed as follows:

$$T_E^{i+1} = \min\left(\gamma T_E^{i+1}, T_E^{\max}\right) \tag{5.6}$$

where $T_E^{\max}$ is the maximum *Inter-Experiment* timer, and γ is the back-off factor. We clamp the back-off at a maximum to guarantee the sender will periodically probe for spare bandwidth. The *Inter-Experiment* timer of the new stream is rested to initial value T_E^{init} , when the switch experiment to this stream succeeds.

The switching experiment duration T_S starts with an initial value T_S^{init} and is updated using an exponential moving average of the time difference between starting a switching

experiment to the failure detection time. If no congestion is detected for a duration of T_S seconds, then the server decides to stay at this higher bandwidth. Otherwise the switching experiment is terminated by switching down.

In order to summarize the description of our overall adaptation strategy, we represented it as a flow diagram as in Figure 5.3.

In the flow diagram we omit details for simplicity. For instance, when we say Switch Down, we determine the rate to switch down to as in Step 3 of the combined switch down algorithm.

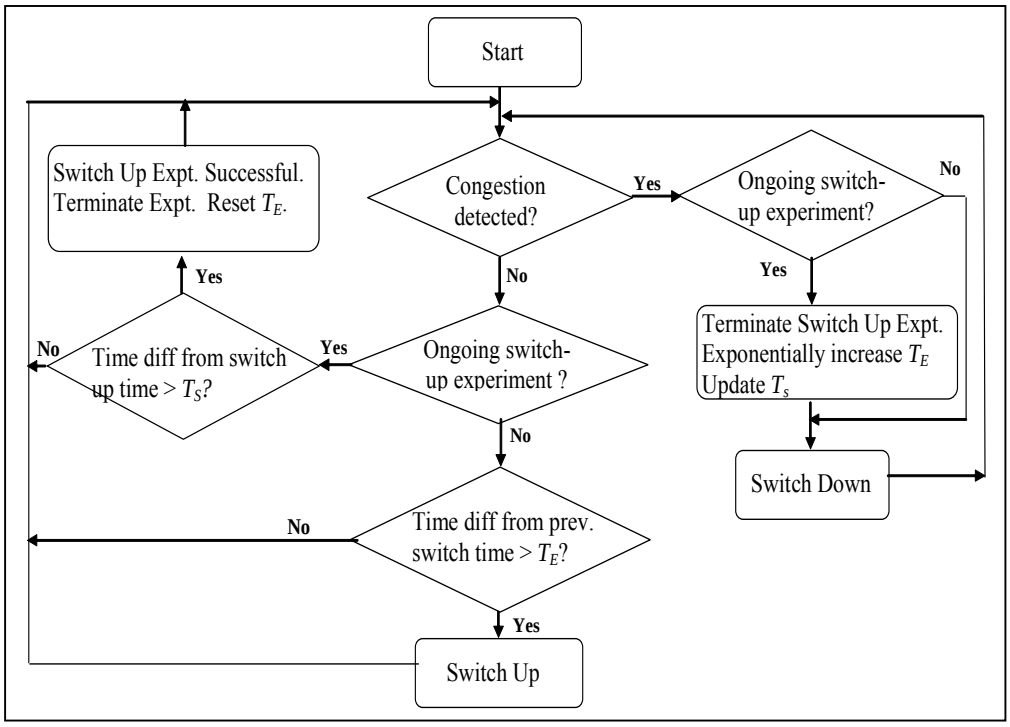


Figure 5.3 Flow diagram for adaptation.

5.3.3 Determining the Adaptation Parameters

The performance of our adaptation strategies is controlled by a set of different parameters that include the sampling interval, the buffer drain time parameters α and β, the switch up times T_S^{init} and T_E^{init}, and the exponential back-off parameter γ.

We should keep our sampling interval small so that we can effectively track the network bandwidth variations. However, a small sampling interval leads to larger overheads in system complexity and transmitted bandwidth[4]. In order to tradeoff these conflicting goals, we select the sampling interval based on the available network bandwidth; sample at small intervals when the network bandwidth is high (and the variations are likely to be more rapid) , and sample at larger intervals when the network bandwidth is low (and the variations are likely to be less frequent). We can do this by sampling every time we transmit a fixed number of bytes. Empirically, we have determined that sampling every time we transmit ~16000 bytes (since we transmit complete packets) provides a good tradeoff for the adaptation. The other adaptation parameters have been tuned empirically to provide a good visual quality, however we can derive analytical bounds on their values based on the statistical properties of the network and video bit-rates. This is a direction of future research.

5.3.4 SCTP Send-Buffer Scaling

Based on the network congestion feedback, the SCTP sender uses the variable *CWND* to estimate the appropriate congestion window size, which determines the maximum number of unacknowledged packets in flight in the network at any time. In addition,

[4] Packets within a sampling interval can be aggregated and transmit them together to reduce network overheads. Hence a small interval leads to smaller packet sizes and larger overheads.

SCTP uses a fixed size *send-buffer* to store application data before the data is transmitted. This buffer has two functions. First, it handles rate mismatches between the application sending rate and SCTP's transmission rate. Second, it is used to keep copies of the packets in flight so they can be retransmitted when needed. Since the *CWND* determines the number of in-flight packets from the *send-buffer*, setting the *send-buffer* to be smaller than the *CWND* would reduce the throughput of the flow. On the other hand, setting the *send-buffer* much larger than the *CWND,* will cause more packets to be delayed in the *send-buffer* until they get chance to be sent in the network when acknowledgments have been received for the previous packets in the buffer. This will lead the application to lose control over quality adaptation, as it has to wait longer before making its quality adaptation decisions. In addition, fixed buffer size can introduce significant latency into the SCTP stream, as the packets have to sit in the SCTP send buffer until they get chance to be sent in the network [109].

We propose to dynamically adapt the *send-buffer* size to be at least *CWND* packets. However, if the *send-buffer* size is limited to *CWND*, then 1) SCTP must inform the application when it has available space for more packet(s), as well as when it increases the *CWND*, and the application must write the next packet(s) before SCTP can send it. Thus, system timing and scheduling behavior can affect TCP throughput. 2) Back-to-back acknowledgment arrivals exacerbate this problem. These adverse effects throughput, and can be reduced by adjusting the buffer size so that it is larger than *CWND*. We selected to set the *send-buffer* size to be 2 * *CWND,* which ensures that the SCTP has a window worth of unsent data to keep the self-clock of acknowledgments flowing [87]. Also using a buffer size of 2 * *CWND* will not affect the throughput of SCTP.

The proposed modification can be implemented at the application level by monitoring two SCTP variables, *CWND* and sending buffer occupancy, and ensures that the sending buffer occupancy does not exceed 2 * *CWND*. However, this implementation is expensive as the application has to continuously poll those SCTP variables, also it can delay the adaptation, as the application is not informed immediately when there is an available space in the SCTP *send-buffer*. For this reason, we chose to implement the *send-buffer* scaling at the SCTP level, so that the SCTP sender continuously update the *send-buffer* following equation:

$$SCTP\ send\text{-}buffer(t) = 2 * CWND(t) \tag{5.7}$$

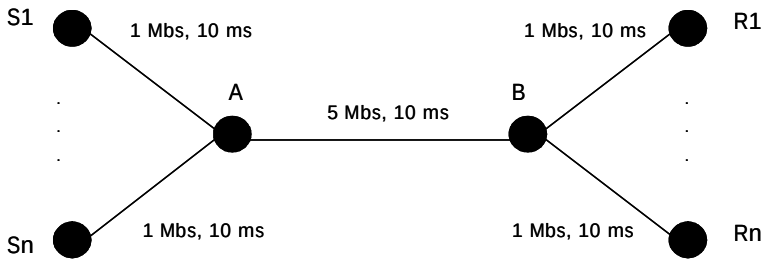
The sending application uses non-blocking write calls to ensure that the application is not blocked while there is no space in the SCTP buffer to accept more data, and the data stays in the application buffer. Thus the application buffer size will accurately reflect the current network conditions.

Although the idea of scaling the *send-buffer* size have been proposed by other authors [109][87], their work focuses on improving SCTP throughput in high delay * bandwidth paths and protocol latencies, while our work focus on improving media adaptation accuracy. Our performance evaluation results, in Section 5.4, show that this simple modification does not affect the SCTP throughput, compared to SCTP with a fixed buffer size, while it improves the accuracy of the quality adaptation.

5.4 Performance Analysis

In order to examine our proposed adaptation strategies, we implemented them in OPNET network simulation tool [89]. We used the topology shown in Figure 5.4, where we assume that source S1 is representing the video server, while receiver R1 represents the video client, and S1 is using SCTP for streaming the video to the R1. Sources S2 – Sn are generating traffic that share the bottleneck A-B with the video stream. Unless specified otherwise, we assume that the bottleneck bandwidth is 5 Mbps and the Round Trip time (*RTT*) is 10 ms.

For the source S1, we use a XviD video codec [97] to encode a 320×240 - 15 frames per second (fps) video sequence at different bit-rates, v_j (512, 420, 335, 255, 210, 170 Kbps). The adaptation mechanism parameters are: N = 6, D = 3 sec , $\gamma = 2$, $T_E^{init} = 10$ sec, $T_E^{\max} = 60$ sec, $T_S^{init} = 10$ sec. Additionally, we select the parameters α = 0.4, and β = 0.5. We measure the video quality in terms of the achieved average bit-rate, variations in the quality at the client, and data loss due to prefetch buffer starvation which can result in frame drops or pauses in the video to allow for rebuffering. Hence, we want to track the available bandwidth faithfully, while minimizing prefetch buffer underflow and maintaining a relatively constant quality by minimizing the number of stream switches.



5.4 Performance study topology.

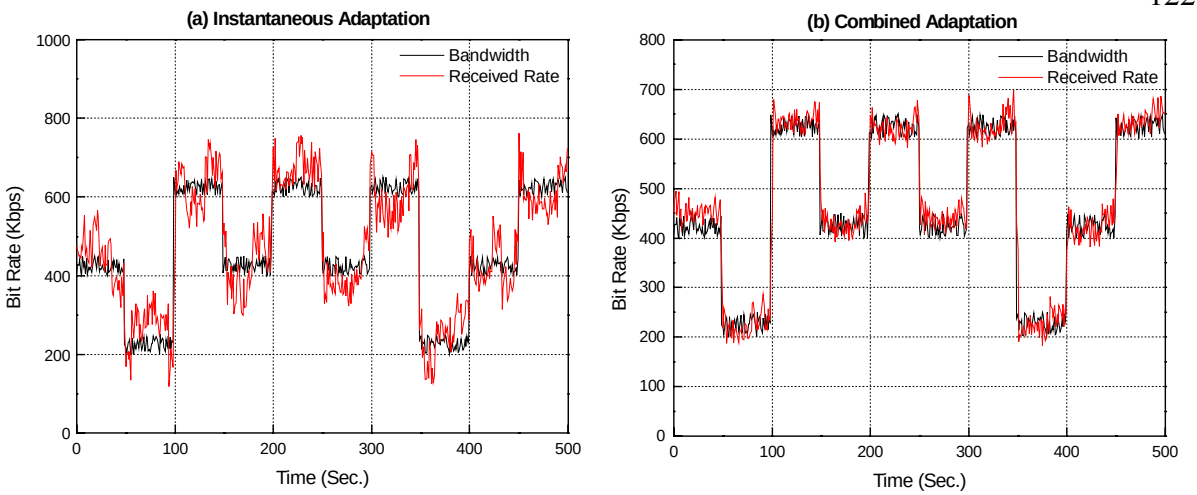


Figure 5.5 Adaptation performance. (a) Instantaneous adaptation. (b) Combined adaptation.

5.4.1 Bandwidth Adaptation

We varied the number of active source-receiver pairs, every 50 seconds, so as to vary the bottleneck bandwidth A-B between S1 and R1 in the range of 200, 400, 600 kbps. We present sample results (~500 seconds) to highlight the performance of our adaptation and compare the instantaneous and combined decision strategies. In Figure 5.5, we plot the network trace against the measured bandwidth at the client that uses a 2 second averaging moving window with overlaps of 1 second. As expected the instantaneous decision leads to over-aggressive switching down, thereby not following the network trace accurately, unlike the combined decision strategy. However, as we have mentioned before, bandwidth fidelity is not the only performance metric we evaluate. We also measure the number of stream switches and the number of dropped packets (due to server buffer overflow). The combined adaptation outperforms the instantaneous adaptation both in terms of the achieved bandwidth as well as in terms of a smaller number of stream switches and lost packets.

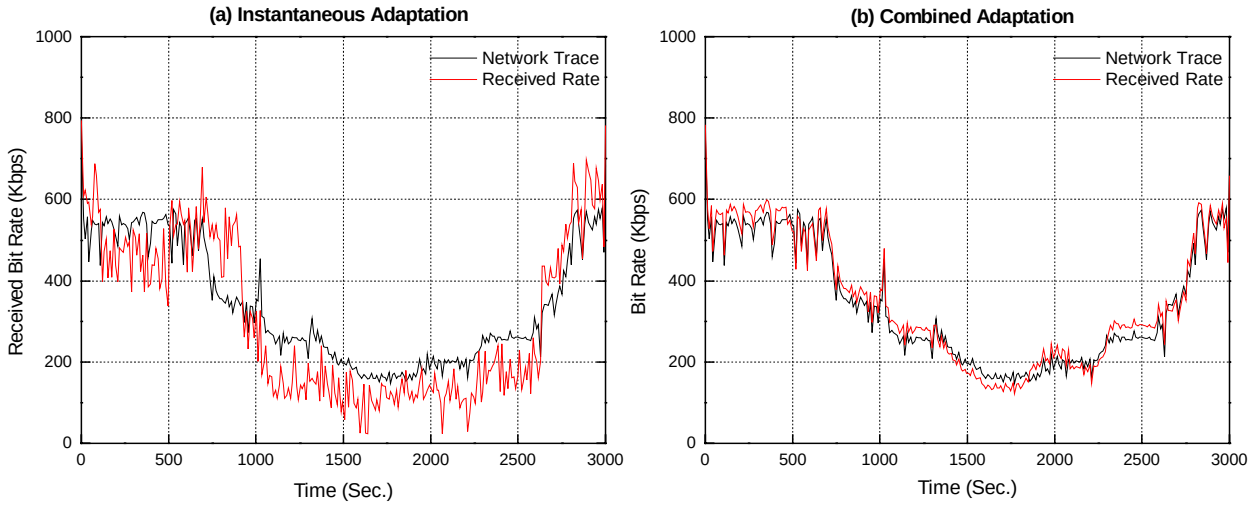


Figure 5.6 Adaptation performance for real trace. (a) Instantaneous adaptation. (b) Combined adaptation.

We quantify these results for a real network trace, obtained from the PlanetLab [110], between two nodes one in the US east coast and the other in the west coast. The traces were collected over 70 minutes with 50 competing TCP connections and the bandwidth varied between 700 Kbps and 120 Kbps. We show the adaptation performance over 3000 seconds in Figure 5.6. Clearly, the instantaneous decision strategy is overly conservative leading to many switched down and under-utilization of the available bandwidth.

Table 5.1 presents the number of stream switches and the number of dropped packets. The combined strategy has better achieved bandwidth as well as fewer stream switches. Also the number of the lost frames is less than the instantaneous strategy. We also use subjective quality assessments for the reconstructed video to validate these observations. A direction of further research is combining these numbers into one visual quality metric.

		Inst. Adapt.	**Comb. Adapt.**
Achieved Average Bit-Rate (kbps)		325	360
Number of Stream Switches		344	109
Data Loss	Number of Lost Packets	544 (0.92%)	109 (0.13%)
	Number of lost video frames	78	6

Table 5.1 Adaptation performance results for real trace.

To have a better understanding of both adaptation strategies, Figure 5.7 shows the clients average buffer occupancy for instantaneous and the combined adaptation strategies. The client buffer has been averaged every 10 seconds. It is clear that aggressive reaction of the instantaneous adaptation scheme cause continuous underflow of the client's buffer, leading to many packets to lose their deadlines. On the other hand, the combined adaptation scheme keeps the buffer occupancy stable.

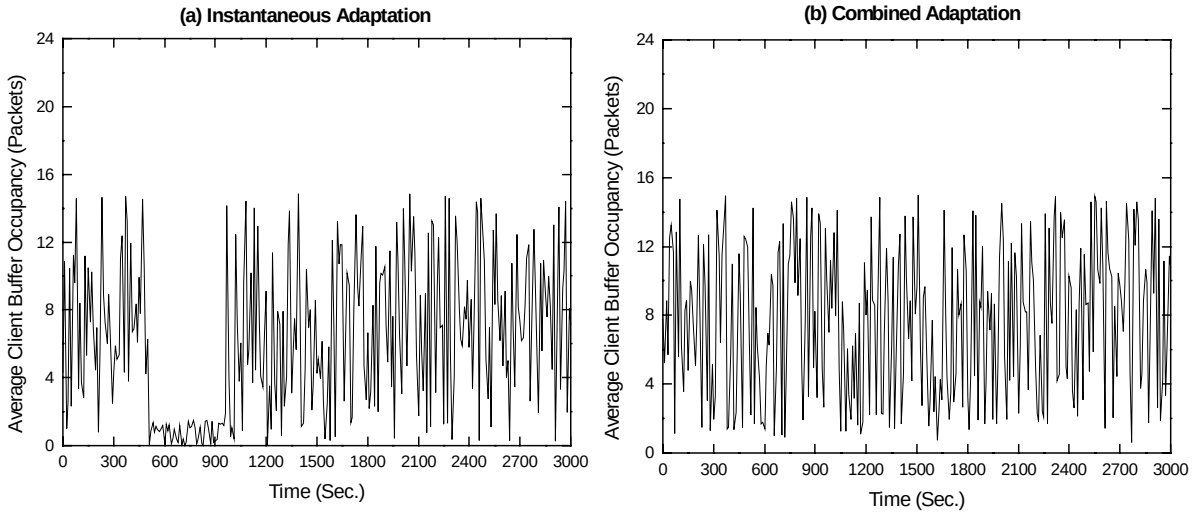


Figure 5.7 Client's average buffer occupancy. (a) Instantaneous adaptation. (b) Combined adaptation.

5.4.2 Effect of SCTP Send-Buffer Scaling

To examine the combined adaptation mechanism with the SCTP *send-buffer* scaling, we compared the adaptation mechanism, with and without the *send-buffer* scaling. To vary the bandwidth between sender S1 and receiver R1, we varied the number of active SCTP connections, as shown in Table 5.2.

Simulation Time (Sec.)	Number of SCTP Connections
0	25
15	20
30	40
70	20

Table 5.2 Number of active SCTP connections.

Figure 5.8 shows the bandwidth available to the SCTP connection between S1-R1, as well as the video stream rate received at the client R1. The graph shows that although the adaptation mechanism without the buffer scaling is able to track the available bandwidth, using the buffer scaling will allow the application to track the available bandwidth more accurately, as the server buffer will be more reflective to the SCTP available bandwidth than the using a fixed size for the SCTP *send-buffer*.

To ensure that the SCTP *send-buffer* scaling will not affect the SCTP throughput We run different number of SCTP streams through the bottleneck A-B, and we set all the connections to last for 200 seconds. We assumed that the SCTP connections will always have data to send. For each set of SCTP streams we run the experiment with and without the SCTP *send-buffer* scaling. We calculated the normalized SCTP throughput as the

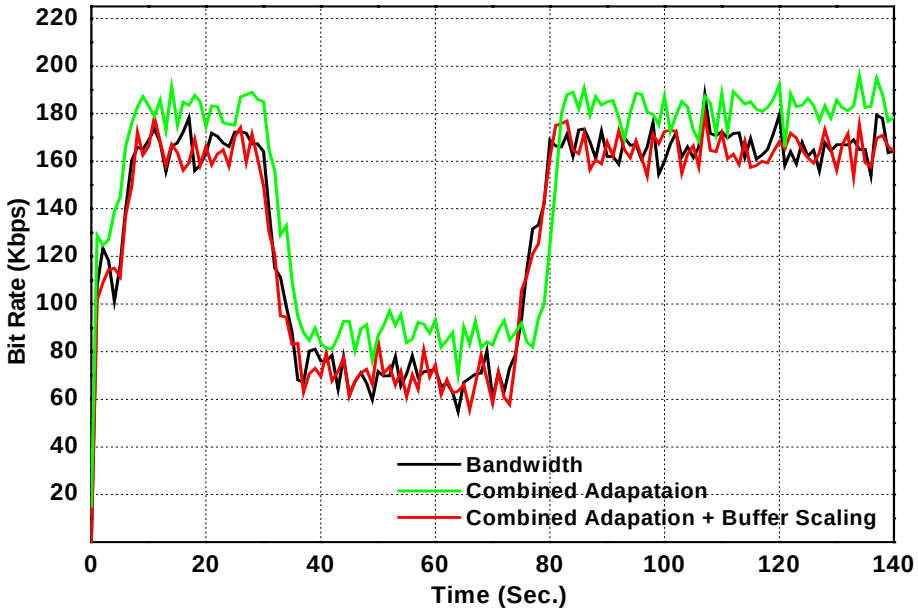


Figure 5.8 Video Stream rate with a variable available SCTP bandwidth.

average throughput of a SCTP connection with the buffer scaling option to that without the buffer scaling.

Figure 5.9 shows the normalized SCTP throughout, with 90% confidence interval, versus the number of active SCTP connections. The figure shows that the SCTP buffer scaling

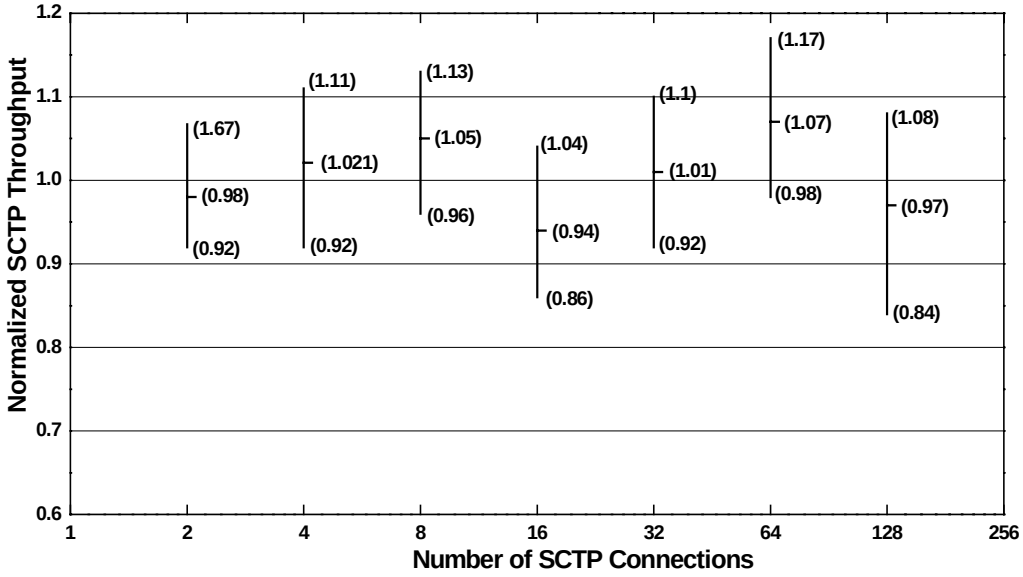


Figure 5.9 Normalized SCTP throughput versus number of connections.

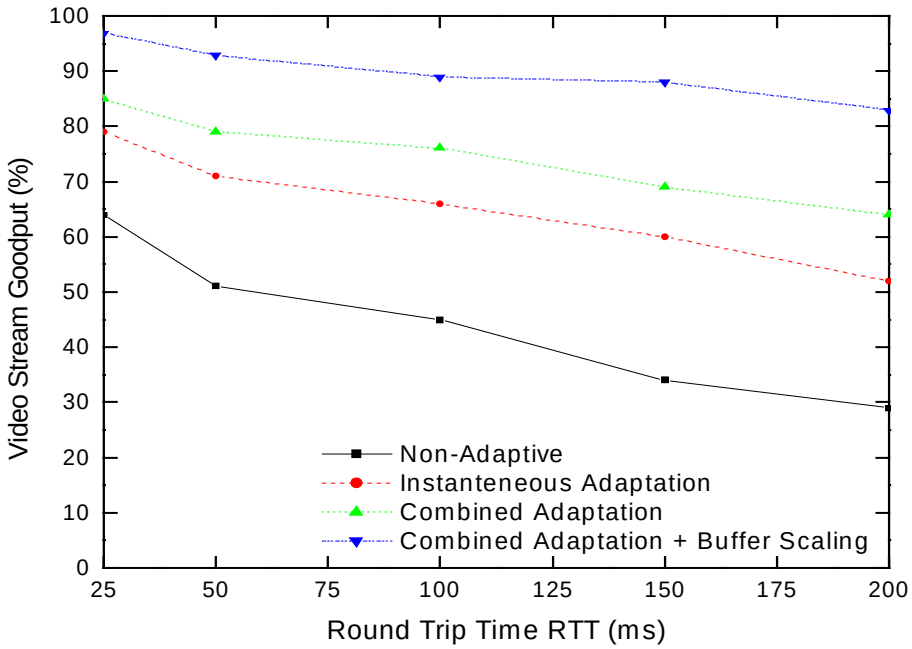


Figure 5.10 Video stream goodput versus round trip time.

will not have an adverse effect on the SCTP throughput.

In Figure 5.10, we examined the video stream goodput as a function of the *RTT*. We define the goodput as the percent of video packets that arrive before the display time of their video frame at the client to the total number of video packets sent from the server. In all the experiments we set the client pre-buffering to 3 seconds.

Results show that without quality adaptation 35% to 71% of the non-adaptive stream packets will miss their deadlines at the client. The reason behind this that the video sender is not adapting its rate to the available bandwidth, which leads many packets to be delayed at the sender and to miss their deadlines. This is reflected at the client as buffer underflow events, that leads the displayed video to continually freezes.

The proposed adaptive streaming enhances the goodput, given the limited pre-buffering delay. The *send-buffer* scaling provides the best goodput, as the video packets will only

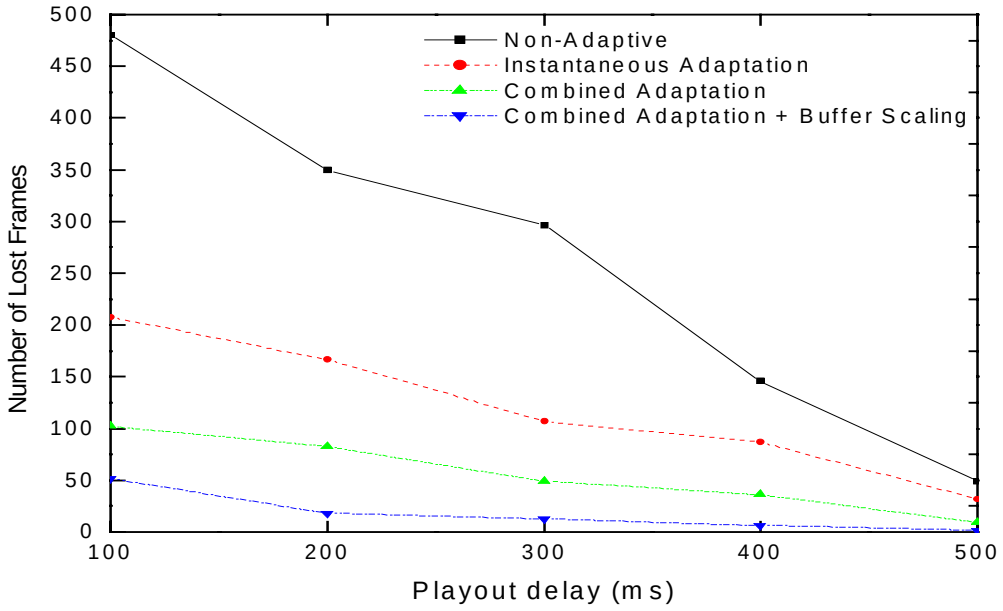


Figure 5.11 Number of lost frames versus client's playback delay.

be accepted by the SCTP if they have a chance to be sent soon, thus they will not be delayed in the TCP *send-buffer* waiting to be sent, thus reducing the packets end-to-end delay and allowing more packets to arrive in time. The figure also shows that in all cases the stream goodput becomes worse as the *RTT* increases. The sensitivity of the goodput to the *RTT* in is due to the fact that increasing the *RTT* increases the time for the packets to reach the client, thus they can miss their deadlines, with the limited client pre-buffering.

Figure 5.11 shows the number of lost frames as we change the playout delay at the client to represent different degrees of application's strict delay requirements. The lost frames are either frames lost in the network or frames that lost their display deadlines. The combined adaptation with buffer scaling provides the minimum number of lost frames, even with the strict playout delay, as it adapts the video rate to faithfully follow the available bandwidth.

5.5 Implementation Evaluation

In order to further examine the proposed adaptation mechanisms in a realistic environment [111], we implemented the adaptation mechanism within the IBM's Adaptive Rich Media Streaming system [112]. Our testbed consists of a live source attached to an ARMS broadcaster that is connected to the video server via 100 Mbps Ethernet. The client connects to the server via a NIST Net box [113], which is used to control the bandwidth between the server and the client. The testbed is shown in Figure 5.12.

At the broadcaster, we use two independent encodings of 320×240 video, one at 512 Kbps and 15 frames per second (fps), and the other at 260 Kbps at 10 fps. From these we derive multiple sub-streams, by dropping frames, with bit-rates V_j (512, 417, 334, 255, 251, 213, 85 Kbps). The audio bit rate is fixed and equal to 32 Kbps. The adaptation mechanism parameters are: $N = 3$, $D = 3$ sec , $\gamma = 2$, $T_E^{init} = 10$ sec, $T_E^{\max} = 60$ sec, $T_S^{init} = 10$ sec. Additionally, we select the parameters $\alpha = 0.4$, and $\beta = 0.5$.

Again we used a real network trace, obtained from the PlanetLab [110], between two nodes one in the US east coast and the other in the west coast. The traces were collected

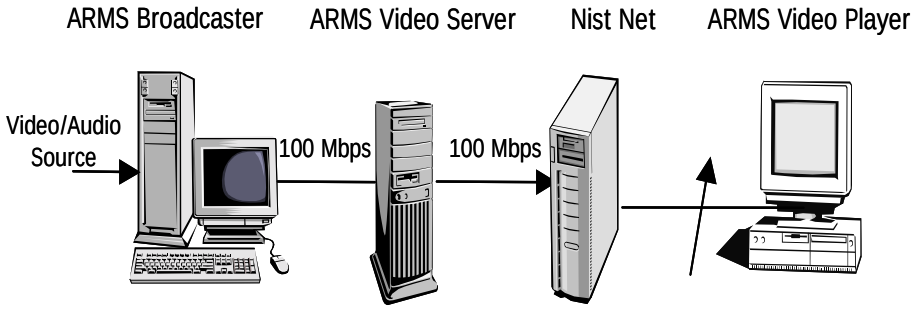


Figure 5.12 Performance study testbed.

over 70 minutes with 50 active TCP connections and the bandwidth varied between 700 Kbps and 120 Kbps. At the broadcaster, we again create two independent encodings of 320×240 video, one at 512 Kbps and 15 fps, and the other at 260 Kbps at 10 fps. The multiple sub-streams with bit-rates V_j are shown in Table 5.3.

Encoding	Sub-Stream	GOP Structure	Bit-Rate (Kbps)
1	1	I B B B P B B B P	255
	2	I B X B P B X B P …	213
	3	I X B X P X B X P …	171
	4	I X X X P X X X P …	129
	5	I X X X X X X X P …	85
2	1	I B B P B B P …	512
	2	I B X P B X P …	417
	3	I X X P X X P …	334

Table 5.3 Generated streams for the testbed experiment.

The audio stream is again maintained at 32 Kbps. We show the adaptation performance over 3000 seconds in Figure 5.13. Clearly, the instantaneous decision strategy is overly conservative leading to many switches down and under-utilization of the available bandwidth. We present the number of stream switches and the number of dropped packets in Table 5.4.

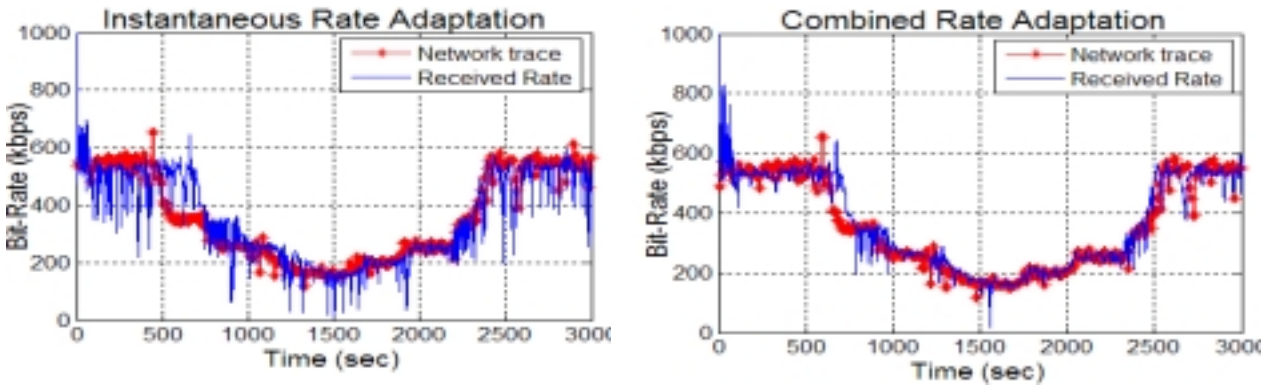


Figure 5.13 Adaptation performance from testbed and real trace.

The combined strategy has better achieved bandwidth as well as fewer stream switches, but has more lost frames. However, a majority of frames lost are B frames, and hence do not contribute to any error propagation.

		Inst. Adapt.	**Comb. Adapt.**
Achieved Average Bit-Rate (kbps)		355	380
Number of Stream Switches		324	118
Data Loss	Number of Lost Packets (Audio and Video)	65 (0.14%)	459 (0.8%)
	Number of lost video frames	7	92

Table 5.4 Adaptation performance results for testbed with real trace.

The number of lost P frames (37) is smaller than the number of times that the instantaneous decision switches down to an all I channel, or worse to only audio, which means that the visual interruptions for the combined decision are fewer than for the instantaneous strategy. We also use subjective quality assessments to validate these observations. A direction of further research is combining these numbers into one visual quality metric.

5.6 Summary

In this chapter we presented server adaptation mechanisms, for real-time multimedia streaming over AIMD transport protocols. The server estimates information about the available network bandwidth by monitoring the application buffer and performs stream switching to meet bandwidth and delay constraints.

We investigate instantaneous and look-ahead strategies for switching down the transmission rate, and switch up the rate in a controlled manner after observing periods of no congestion. We estimate the current and future server buffer drain delay, and derive the transmission rate to minimize client buffer starvation. In addition, we presented a simple scaling mechanism to the SCTP send buffer that allows the adaptation mechanism to follow accurately the network bandwidth and reduce the adaptation decision time. We evaluated these algorithms and compare their performance in terms of their effect on the decoded video quality by measuring the average streaming bandwidth achieved by the algorithm, the number of stream switches, and the data loss (caused due to server buffer overflow). The strategy with combined look-ahead and instantaneous decisions can follow the network bandwidth accurately, while minimizing stream switches, and providing high visual quality. In addition, the proposed SCTP *send-buffer* scaling does not affect the SCTP throughput, compared to SCTP with a fixed buffer size, while it improves the accuracy of the quality adaptation.

Chapter 6
Conclusions and Future Work

We conclude this dissertation with a summary of our contributions and directions for future work.

6.1 Summary

This thesis identifies three fundamental challenges that degrade the performance of multimedia in heterogeneous networks:

1. Low wireless channel bandwidth.

Multimedia performance in many wireless networks suffers because low channel bandwidth, which limits the quality of the media to be streamed to the wireless users.

2. High wireless bit-error rate.

Wireless channels suffers from high bit-error rate that are typically around 10^{-2}, which range from single bit errors to burst errors or even intermittent loss of the connection The high error rates are due to multi-path fading, which characterizes mobile radio channels,

while the loss of the connection can be due to the mobility in such networks. This can have a devastating for the transport of compressed video in wireless networks.

3. Unsuitability of Additive-Increase Multiplicative-Decrease (AIMD) Transport Protocols for Video Transport.

While the majority of traffic on the Internet today is comprised of AIMD transport protocols, such as TCP/SCTP flows, conventional wisdom holds that such protocols are unsuitable for "real-time" traffic due to its lack of throughput guarantees and insistence on reliability. Many proposals for new transport protocols for the purpose of solving the video transport problem over the Internet. These protocols need to be TCP-friendly to ensure that they will not cause network collapse. However, proving a new transport protocol to be TCP-friendly can be difficult, because the dynamics of TCP congestion control is extremely complex. Thus an advantage of using TCP/SCTP in transporting video is that the transport is ensured to be friendly to other flows sharing the same network. Also in many situations streaming over TCP/SCTP is unavoidable, such as when the client machines are located behind network firewalls permitting only inbound HTTP traffic.

This dissertation analyzed the problems posed by the above challenges, and solved them using a combination of application-layer techniques and modifications and enhancements to transport-layer at the sender and receiver.

• **Transport Layer Bandwidth Aggregation Protocol:** We designed and evaluated an extension to SCTP to be able to aggregate the bandwidth of all the active transmission paths between the communicating endpoints. We refer to the protocol extension as Load

Sharing-SCTP (LS-SCTP). LS-SCTP dynamically adds/drops communication paths to the aggregated paths bundle. To ensure that the aggregate connection will not stall due to the high loss rate or continuous failure of individual paths, LS-SCTP includes path monitoring and packet assignment mechanisms that stripe the packets on the paths according to their current conditions. The new protocol extension is effective in providing throughput and reliability in wireless networks with low bandwidth and unreliable channels. The details of the protocol extensions are discussed in Chapter 3.

• **Unequal Error Protection for Wireless Video:** We designed an error control mechanism for interactive video in mobile wireless networks. The mechanism provides unequal error protection to data within the video stream according to their importance to the reconstructed video quality. We realized the unequal error protection through extending the classic retransmission based error control, with redundant retransmissions, where the sender retransmits multiple copies of the packet. The number of copies depends on the reliability level required for the data within the packet. In order to increase the probability that at least one of the retransmitted packets arrive at the receiver, in less number of retransmissions, we send the redundant retransmissions on diverse paths. A delayed constrained retransmission, based on the packet lifetime and estimate of the delay from the sender to receiver, is used to prevent re-transmitting expired packets. We implemented the proposed mechanism as an extension to RTP, refereed to as Multi Path - RTP (MP-RTP). The mechanism is able to provide a good quality for interactive video under different packet loss rates and under wide range of node mobility speeds. In addition, comparing the overhead of the mechanism to the overhead of reference frame updates error control mechanism, it was shown that for a given video reconstruction

quality, MP-RTP has less overhead. The details of MP-RTP are discussed in Chapter 4.

• **Enabling real-time video over AIMD transport protocols:** We proposed server adaptation mechanisms for real-time video over AIMD transport protocols. The server estimates information about the available network bandwidth by monitoring the application buffer and performs stream switching to meet bandwidth and delay constraints. We investigate a strategy that combined instantaneous and look-ahead strategies for switching down the transmission rate, and switch up the rate in a controlled manner after observing periods of no congestion. We estimate the current and future server buffer drain delay, and derive the transmission rate to minimize client buffer starvation. In addition, we presented a simple scaling mechanism to the transport protocol *send-buffer* that allows the adaptation mechanism to follow accurately the network bandwidth and reduce the adaptation decision time. The strategy with combined look-ahead and instantaneous decisions shows to follow the network bandwidth accurately, while minimizing stream switches, and providing high visual quality. In addition, the proposed *send-buffer* scaling does not affect the transport protocol throughput, compared to fixed buffer size, while it improves the accuracy of the quality adaptation. These details are discussed in Chapter 5.

6.2 Availability

Standard SCTP as well as implementations of the protocols presented in this thesis are also available in the *OPNET* network simulator.

They can be downloaded from

http://www-ee.ccny.cuny.edu/wwwa/web/aabdelal/#Downloads.

The Nist Net network emulation package is available at

Nist Net Network Emulator, http://snad.ncsl.nist.gov/itg/nistnet/

6.3 Future Directions

There are several interesting directions for future work based on the work described in this dissertation.

Some of these are extensions of our work, while some others are motivated by the more general problem of multimedia over limited bandwidth and error-prone networks.

Adaptive Multimedia Scheduling: The network layer should adaptively schedule packets not only based on the their delay requirements, but also based on estimates of the link delays and continuously varying channel conditions. Link delays can be estimated either through monitoring the size changes of the link layer buffer, while channel conditions can be obtained by real-time measurement of the channel's signal-to-interference-plus-noise power ratio (SINR) on the physical layer, which can be used to determine the channel bit rate and bit error rate (BER). Then the adaptive scheduler can decide the service order of the packets so as to meet the delay requirements set by the application. The scheme can also be extended to provide different levels of services, i.e., best effort and time-bounded services.

Cross-layer Approach for Adaptive Hybrid Error Protection for Video in Ad hoc Networks: Inspired by our research in adaptive unequal protection for wireless video, we are planning to investigate a hybrid error protection scheme that switches between the FEC and retransmission based on the channel condition and delay requirements of the underlying application, as well as knowledge of the channel condition retrieved from the

lower layers. Under short burst losses, FEC with a little redundancy can be effective in quickly recovering lost packets while under long burst losses, retransmission can recover those packets irrecoverable by FEC. Since retransmission occurs only when some indication of packet losses exists, this hybrid scheme contributes to reducing the overall bit overhead. In addition, since FEC recovers most of random and short burst losses which happens more frequently, it expedites packet recovery and effectively confines error propagation.

Cross-layer Optimized Multi-path Routing: Recent research shows that multi-path routing is significantly better than single path routing in ad hoc network [102]. However, current multi-path forwarding mechanisms do not consider the channel condition when selecting the path to forward the data on. We believe that through sharing the information collected by the MAC and link layers regarding the signal-strength, channel quality, link stability and energy level, the network layer can implement multi-path reliable forwarding mechanism to deliver data over existing paths, to achieve traffic load balancing. The decision should also take into account the channel, node and network condition as well as the application requirements, including link quality, path stability and packet delay.

Adaptive Real-time Video: The performance evaluation of our proposed adaptive real-time video over AIMD transport protocols, described in Chapter 5, encourage us to continue our future research to develop analytical methods to determine the parameters of our adaptation algorithms, that should take in account the network conditions and the required video quality. We are also planning to tune the parameters of the adaptation to be able to receive information from transport protocol with less of a delay. In addition,

we will investigate methods of integrating different performance metrics, including frame loss rate and number of stream switches, into one visual quality metric that is consistent with subjective quality evaluations.

Adaptive Media Playout for Low-Delay Video Streaming: When media is streamed over best-effort networks, media data is buffered at the client to protect against playout interruptions due to packet losses and random delays. While the likelihood of an interruption decreases as more data is buffered, the latency that is introduced increases. We plan to investigate adaptive media playout mechanisms, which vary of the playout speed of media frames depending on channel conditions. This allows the client to buffer less data when there is available bandwidth for streaming and when the network loss rate is low, thus introducing less delay, for a given buffer underflow probability.

Playout buffer requirements for video streaming over AIMD transport Protocols: Client-side buffer is currently used to smooth out accommodate the variability of the bandwidth introduced by the saw-tooth fluctuation of a TCP/SCTP flow. However, currently there are no guidelines for the provisioning of the client's buffer, and smooth playout is insured through over-provisioning. We are interested in investigating the right size for the playout buffer in order to insure a prescribed video quality. The size of the playout buffer is required to be adaptable based on the available network bandwidth and the desired video quality.

Appendix A

Protocol Feature	SCTP	TCP	UDP
State required at each endpoint	Yes	Yes	No[a]
Reliable data transfer	Yes	Yes	No
Congestion control and avoidance	Yes	Yes	No
Message boundary conservation	Yes	No[b]	Yes
Path MTU discovery and message fragmentation	Yes	Yes[b]	No
Message bundling	Yes	Yes[b]	No
Multi-homed hosts support	Yes	No	No
Multi-stream support	Yes	No	No
Unordered data delivery	Yes	No	Yes
Security cookie against SYN flood attack	Yes	No	No
Built-in heartbeat (reachability check)	Yes	No[c]	No

[a]With udp a node can communicate with another node without going through a setup procedure or changing any state information. This is sometimes called connection-less, but in reality each udp packet has the needed state within it to form a connection so that no ongoing state needs to be maintained at each endpoint.

[b]Because TCP treats all the data passed from its upper layer as a formatless stream of data bytes, it does not preserve any message boundaries. However, due to byte-stream-based nature, TCP can automatically resize all the data into new TCP segments suitable for the Path MTU before transmitting them.

[c]Most TCP implementations do implement a "keep-alive" mechanism. This mechanism is very similar to the SCTP heartbeat, with the main difference being the time interval used. In TCP the "keep-alive" interval is, by default, set to two hours. The goal of this "keep-alive" is long-term state cleanup, which is in sharp contrast to SCTP's much more rapid heartbeat, which is used to aid in fast fail-over.

Bibliography

[1] P. Palumbo and I. Brodsky. Wireless Streaming Media: Hard Numbers on Supply & Demand. Market research report by Datacomm Research Company & Accustream iMedia Research, 2002. Available from http://www.wow-com.com.

[2] J. B. Postel. Transmission Control Protocol. Information Sciences Institute, September 1981. RFC-793.

[3] R. Stewart et al. Stream Control Transmission Protocol, October 2000. RFC-2960.

[4] O. Kyas and G. Grawford. ATM Networks. Pearson Education, 2002.

[5] International Organization for standardization, Overview of the MPEG-4 Standard. December 1999.

[6] ITU-T Recommendation, H.263, Video Coding for Low Bitrate Communication, September 1997.

[7] Y. Wang, S. Wenger, J. Wen, and A. Katsaggelos. Error Resilient Video Coding Techniques. *IEEE Signal Processing Magazine*, 17(4):61-82, July 2000.

[8] B. Wah, X. Su, and D. Lin. A Survey of Error-Concealment Schemes for Real-time Audio and Video Transmissions over the Internet. In *Proc. of Int. Symposium on Multimedia Software Engineering*, December 2000.

[9] B. Girod and N. Farber. Feedback-Based Error Control for Mobile Video Transmission. In *Proc.* of *IEEE, Special Issue on Video for Mobile Multimedia*, 97(10):1707-1723, October 1999.

[10] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-time Applications. July 2003. RFC-3550.

[11] C. Krasic, K Li, and J. Wapole. The Case for Streaming Multimedia with TCP. In *Proc. of 8th International Workshop on Interactive Distributed Multimedia Systems (iDMS)*, September 2001.

[12] P. Hsiao, H. Kung, and K. Tan. Streaming Video over TCP Receiver-based Delay Control. *IEICE Transactions on Communications*, vol. E86-B, no. 2, February 2003.

[13] P. Cuetos, D. Saparilla, and K. Ross. Adaptive Streaming of Stored Video in a TCP –Friendly Context: Multiple Versions or Multiple Layers?. In *Proc. of International Packet Video Workshop*, April 2001.

[14] D. Saparilla and K. Ross. Streaming Stored Continuous Media over Fair-Share Bandwidth. In *Proc. of NOSSDAV 2000*, June 2000.

[15] G. Wu, E. Chong, and R. Givan. Streaming Stored Video over AIMD Transport Protocols. In *Proc. of IEEE 4th International Symposium on Multimedia Software Engineering (ISMSE)*, December 2002.

[16] L. Ong et al. Framework Architecture for Signaling Transport. October 1999. RFC-2719.

[17] L. Coene et al. Telephony Signaling Transport over SCTP Applicability Statement. April 2002, RFC-3257.

[18] R. Stewart et al. SCTP Partial Reliability Extensions. May 2004. RFC-3758.

[19] R. Stewart and Q. Xie. Stream Control Transmission Protocol (SCTP): A Reference Guide. Addison-Wesley, 2001,

[20] M. Handley, J. Padhye, and S. Floyd. TCP Congestion Window Validation. June 2000, RFC-2861.

[21] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgement Options. October 1996. RFC-2018.

[22] K. Fall and S. Floyd. Simulation-based comparisons of Tahoe, Reno and SACK TCP. *Computer Communication Review*, 26(3):5-21, July 1996.

[23] K. Ramakrishnan and S. Floyd. A proposal to add Explicit Congestion Notification (ECN) to IP. January 1999. RFC-2481.

[24] D. Chiu and R. Jain. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Computer Networks and ISDN Systems,* 17(1):1-14, June 1989.

[25] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. April 1999. RFC-2581.

[26] M. Allman. TCP Byte Counting Refinements. *ACM Computer Communication Review*, 29(3),July 1999.

[27] R. Brennan and T. Curran. SCTP Congestion Control: Initial Simulation Studies. In *Proc. of 17th International Teletraffic Congress*, December 2001.

[28] S. Fu and M. Atiquzzaman. SCTP: State of the art in Research, Products, and Technical Challenges. *IEEE Communications Magazine*, 42(4):64-76, April 2004.

[29] A. Caro et al. Improving Multimedia Performance over Lossy Networks via SCTP. ATIRP 2001, March 2001

[30] M. Atiquzzaman and W. Ivancic. Evaluation of SCTP Multi-streaming over Satellite Links. In *Proc. of International Conference on Computer Communications and Networks,* October 2003.

[31] A. Jungmaier, E. Rathgeb, M. Schopp, and M. Tüxen. SCTP- A Multi-link End-to-End Protocol for IP-based Networks. *International Journal of Electronics and Communications*, 55 (1):46-54, 2001.

[32] J. Noonan, P. Perry, and J. Murphy. Study of SCTP Services in a Mobile-IP Network. In *Proc. of IT&T Annual Conference*, October 2002.

[33] T. Ravier, R. Brennam, and T. Curran. Experimental Studies of SCTP Multi-homing. In *Proc. of First Joint IEI/IEE Symposium on Telecommunications Systems Research*, November 2001.

[34] A. Caro, J. Iyengar, P. Amer, G. Heinz, and R. Stewart. A Two-level Threshold Recovery Mechanism for SCTP. In *Proc. of Multi-Conference on Systemics, Cybernetics and Informatics (SCI),* July 2002.

[35] J. Iyengar, A. Caro, P. Amer, G. Heinz, and R. Stewart. SCTP Congestion Window Overgrowth During Changeover. In *Proc. of Multi-Conference on Systemics, Cybernetics and Informatics (SCI),* July 2002.

[36] G. Ye, T. Saadawi, and M. Lee. SCTP Congestion Control Performance in Wireless Multi-hop Networks. In *Proc. of MILCOM*, October 2002.

[37] H. Elsayed, A. Abd El Al, T. Saadawi, and M. Lee. Synchronization Algorithm for SCTP Network. In *Proc. of International Workshop on Multimedia Network Systems and Applications* (MNSA), May 2003.

[38] R. Katz, G. Gibson, and D. Pattersan. Disk System Architecture for High Performance Computing. In *Proc. of IEEE*, 77(12):1842-1858, December 1989.

[39] C. Brenden, S. Traw, and M. Smith. Striping within the Network Subsystem. *IEEE Network*, 9(4)22-32, July 1995.

[40] Computer Staff. Gigabit Network Testbeds. *IEEE Computer*, 23(9):77-80, September 1990.

[41] J. Duncanson. Inverse Multiplexing. *IEEE Communications Magazine*, 32(4):34-41, April 1994.

[42] A. Snoeren. Adaptive Inverse Multiplexing for Wide-Area Wireless Networks. In *Proc. of IEEE GlobeCom,* December 1995.

[43] L. Magalhaes and R. Kravets. Transport Level Mechanisms for Bandwidth Aggregation on Mobile Hosts. In *Proc. of IEEE ICNP*, November 2001.

[44] D. Phatak and T. Goff. A Novel Mechanism for Data Streaming Across Multiple IP Links for Improving Throughput and Reliability in Mobile Environments. In *Proc. of IEEE INFOCOM*, New York, June 2002.

[45] M. Papadopouli and H. Schulzrinne. Connection Sharing in Ad Hoc Wireless Network among Collaborating Hosts. In *Proc. of NOSSDAV*, June 1999.

[46] L. Magalhaes and R. Kravets. MMTP: Multimedia Multiplexing Transport Protocol. In *Proc. of Workshop on Data Communications in Latin America and the Caribbean (SIGCOMM-LA),* April 2001.

[47] H. Hsieh and R. Sivakumar. pTCP: An End-to-End Transport Layer Protocol for Striped Connections. In *Proc. of IEEE ICNP*, November 2002.

[48] M. Allman, H. Kruse, and S. Ostermann. An Application-Level Solution to TCP's Satellite Inefficiencies. In *Proc. of Workshop on Satellite-Based Information Services (WOSBIS)*, November 1996.

[49] T. Hacker and B. Athey. The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network. In *Proc. of International Parallel and Distributed Processing Symposium (IPDPS)*, April 2002.

[50] Y. Wang and Q. Zhu. Error Control and Concealment for Video Communications: A Review. In *Proc. of IEEE*, 86(5):974-997, May 1998.

[51] T. Turletti and C. Huitema. Video Conferencing on the Internet. *IEEE/ACM Transactions on Networkin*g, 4(3):340-351, June 1996.

[52] Z. Chen. Coding and Transmission of Digital Video on the Internet. Ph.D. thesis, University of Illinois at Urbana-Champaign, 1997.

[53] W. Zhu, Y. Wang, and Q. Zhu. Second-order Derivative-based Smoothness Measure for Error Concealment in DCT-based Codecs. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(6):713-718, October 1998.

[54] W. Kwok and H. Sun. Multi-directional Interpolation for Spatial Error Concealment. *IEEE Transactions on Consumer Electronic*s, 39(3):455-460, August 1993.

[55] M. Ghanbari and V. Seferidis. Cell-loss Concealment in ATM Video Codecs. *IEEE Transactions on Circuits and Systems for Video Technology*, 3(3):238-247, June 1993.

[56] S. Aign and K. Fazel. Temporal & Spatial Error Concealment Techniques for Hierarchical MPEG-2 Video Codec. In *Proc. GlobeCom,* November 1995.

[57] Q. Zhu, Y. Wang, and L. Shaw. Coding and Cell-loss Recovery in DCT-based Packet Video. *IEEE Transactions on Circuits and Systems for Video Technology*, 3(3):248-258, June 1993.

[58] S. H. Lee, P. J. Lee, and R. Ansari. Cell Loss Detection and Recovery in Variable Rate Video. In *Proc. of Third International Workshop on Packet Vide*o, March 1990.

[59] H. Sun and W. Kwok. Concealment of Damaged Block Transform Coded Images using Projections onto Convex Sets. *IEEE Transactions on Image Processin*g, 4(4):470-477, April 1995.

[60] I. Rhee. Error Control Techniques for Interactive Low-Bit Video Transmission over the Internet. In *Proc. of ACM Sigcomm*, September 1998.

[61] M. Ghanbari. Postprocessing of Late Cells for Packet Video. *IEEE Transactions on Circuits and Systems for Video Technolog*y, 6(6):669-678, December 1996.

[62] E. Posnak, S. Gallindo, A. Stephens, and H. Vin. Techniques for Resilient Transmission of JPEG Video Streams. In *Proc. of Multimedia Computing and Networkin*g, pages 243-252, February 1995.

[63] B. Wah and X. Su. Streaming Video with Transformation-Based Error Concealment and Reconstruction. In *Proc. of International Conference on Multimedia Computing and System*s, June 1999.

[64] T. Ferguson and J. Ranowitz. Self-synchronizing Huffman Codes. *IEEE Transactions on Information Theor*y, 30(4):687-693, July 1984.

[65] T. Kawahara and S. Adachi. Video Transmission Technology with Effective Error Protection and Tough Synchronization for Wireless Channels. In *Proc. IEEE International Conference on Image Processin*g, November 1996.

[67] T. Sikora. The MPEG-4 Video Standard Verification Model. *IEEE Transactions on Circuits and Systems for Video Technolog*y, 7(1):19-31, February 1997.

[68] G. Cote, B. Erol, M. Gallant, and F. Kossentini. H.263+: Video Coding at Low Bit Rates. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(7):849-866, November 1998.

[69] R. Aravind, M. Civanlar, and A. Reibman. Packet Loss Resilience of MPEG-2 Scalable Video Coding Algorithms. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(5):426-435, October 1996.

[70] M. Kansari and M. Vetterli. Low Bit Rate Video Transmission over Fading Channels for Wireless Microcellular Systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:1-11, Feb. 1996.

[71] S. Shunan, L. Shivendra, S. Panwar, and Y. Wang. Reliable Transmission of Video over Ad-hoc Networks Using Automatic Repeat Request and Multi-path Transport. *ACM Sigmobile Computing and Communications Review*, 7(1):59-61, January 2003.

[72] J. Apostolopoulos. Reliable Video Communication over Lossy Packet Networks using Multiple State Encoding and Path Diversity. In *Proc. of International Society for Optical Engineering,* January 2001.

[73] N. Gogate, D. Chung, S. Panwar, and Y. Wang. Supporting Image and Video Applications in a Multihop Radio Environment Using Path Diversity and Multiple Description Coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(9):777-792, September 2002.

[74] Y. Wang, S. Panwar, S. Lin, and S. Mao. Wireless Video Transport using Path Diversity: Multiple Description vs. Layered Coding. In *Proc. of IEEE International Conference on Image Processing*, September 2002.

[75] Y. Lee, J. Kim, Y. Altunbasak, and R. Mersereau. Layered Coded vs. Multiple Description Coded Video over Error Prone Networks. *Signal Processing: Image Communication*, 18:337-356, May 2003.

[76] A. Albanese, J. Bloemer, and J. Edmonds. Priority Encoding Transmission. In *Proc. of IEEE Symposium on Foundations of Computer Science*, November 1994.

[77] J. M. Danskin, G. M. Davis, and X. Song. Fast Lossy Internet Image Transmission. In *ACM Multimedia Conf*erence, November 1995.

[78] X. Wang and H. Schulzrinne. Comparison of Adaptive Internet Multimedia Applications. *IEICE Transactions on Communications*, vol. E82-B, no. 6, June 1999.

[79] P. Mehra and A. Zakhor. TCP-Based Video Streaming Using Receiver-Driven Bandwidth Sharing. *In Proc. of 13th International Packet Video Workshop*, April 2003.

[80] H. Kanakia, P. Mishra, and A. Reibman. An Adaptive Congestion Control Scheme for Real-time Packet Video Transport. *In Proc. of SIGCOMM Symposium on Communications Architectures and Protocols*, September 1993.

[81] E. Amir, S. McCanne, and H. Zhang. An Application Level Video Gateway. *In Proc. of ACM Multimedia*, November 1995.

[82] M. Stemm and R. Katz. Vertical Handoffs in Wireless Overlay Networks. *ACM Mobile Networks and Applications, Special Issue on Mobile Networking in the Internet*, 3(4):335-350, January 1999.

[83] A. Abd El Al, T. Saadawi, and M. lee, Load Sharing in Stream Control transmission Protocol, IETF Internet Draft (draft-ahmed- lssctp-01), Work in Progress, May 2005.

[84] A. Abd El Al, T. Saadawi, and M. Lee. LS-SCTP: A Bandwidth Aggregation Technique for Stream Control Transmission Protocol. *Computer Communications Journal, special issue on Protocol Engineering for Wired and Wireless Networks*, 27(10):1012-1024, May 2004.

[85] E. Blanton and M. Allman. On Making TCP More Robust to Packet Reordering. *ACM Computer Communication Review* 32(1):20-30, January 2002.

[86] A. Abd El Al, T. Saadawi, and M. Lee. Improving Throughput and Reliability in Mobile Wireless Networks Via Transport Layer Bandwidth Aggregation. *Computer Networks, special issue on Future Advances in Military Communications Systems & Technologies*, 46(5):635-649, December 2004.

[87] J. Semke, J. Mahdavi, and M Mathis. Automatic TCP Buffer Tuning. *ACM Computer Communication Review*, 28(4):315-323, October 1998.

[88] D. Johnson and C. Perkins, *IP Mobility Support for IPv4*, 2002, August 2002, RFC-3344.

[89] OPNET Modular. http://www.opnet.com.

[90] OPNET SCTP implementation. http://www-ee.ccny.cuny.edu/wwwa/web/aabdelal/#Downloads.

[91] S. Mao, S. Lin, S. Panwar, Y. Wang and E. Celebi. Video Transport Over Ad Hoc Networks: Multistream Coding With Multipath Transport. *IEEE Journal on Selected Areas in Communications*, 21(10):1721-1737, December 2003.

[92] D. Johnson, D. Maltz, Y. Hu and J. Jetcheva. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks, IETF Internet Draft (draft-ietf-manet-dsr-10.txt), Work in Progress, July 2004.

[93] C. Perkins, E. Royer and S. Dos. Ad Hoc on-Demand Distance Vector (AODV) Routing, July 2003, RFC-3561.

[94] Z. Haas and M. Pearlman. The Zone Routing Protocol (ZRP) for Ad Hoc Networks, work in progress, IETF Internet Draft (draft-ietf-manet-zone-zrp04.txt), Work in Progress, July 2002.

[95] A. Abd El Al, T. Saadawi and M. Lee. A Multi-Path Error Control Mechanism for Interactive Video in Mobile Wireless Networks. *In Proc. of Workshop on Applications and Services on Wireless Networks (ASWN),* August 2004.

[96] E. Gilbert. Capacity of a Burst-Noise Channel. *Bell System Tech Journal*, 39(9):1253-1265, September 1960.

[97] XviD MPEG-4 video codec, http://www.xvid.org.

[98] N. Feamster and H. Balakrishnan. Packet loss recovery for streaming video. *In Proc. of International Packet Video Workshop*, April 2002.

[99] H.263+ Codec. University of British Columbia. [Online]. Available: ftp://dspftp.ece.ubc.ca

[100] A. Abd El Al, T. Saadawi, and M. Lee. Improving Interactive Video in Ad-hoc Networks Using Path Diversity. *In Proc. of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS),* October 2004.

[101] S. Lee and M. Gerla. Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks. In *Proceedings of IEEE ICC*, June 2001.

[102] A. Nasipuri and S. R. Das. On-demand Multipath Routing for Mobile Ad Hoc Networks. In *Proc. of ICCCN*, October 1999.

[103] OPNET DSR Model. The Wireless Communications Technologies Group of the National Institute of Standards and Technology. [Online]. Available: http://w3.antd.nist.gov/wctg/prd_dsrfiles.html

[104] T. Philips, S. Panwar, and A. Tantawi. Connectivity Properties of a Packet Radio Network Model. *IEEE Transactions on Information Theory*, 35(5):1044–1047, September 1989.

[105] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proc. of ACM/IEEE MobiCom,* October 1998.

[106] J. Yoon, M. Liu, and B. Noble. Random Waypoint Considered Harmful. In *Proc. of INFOCOM*, April 2003.

[107] G. Holland and N. Vaidya. Analysis of TCP Performance over Mobile Ad Hoc Networks. *Kluwer Wireless Networks*, vol. 8, no. 2-3, pp. 275–288, March 2002.

[108] A. Abd El Al, T. Saadawi, and M. Lee. Improving Real-Time Video over AIMD Transport Protocols. Submitted to *IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, December 2005.

[109] A. Goel, C. Krasic, K. Li, and J. Walpole. Supporting Low Latency TCP-Based Media Streams. *In Proc. of Tenth International Workshop on Quality of Service (IWQoS)*, May 2002.

[110] PlanetLab Wide Area Network Testbed, http://www.planet-lab.org/

[111] D. S. Turaga, A. Abd El Al, Chitra Venkatramani, Olivier Verscheure, and T. Saadawi. Adaptive Live Streaming over Enterprise Networks. In *Proc. of IEEE International Conference on Multimedia & Expo* (*ICME),* July 2005.

[112] C. Venkatramani, P. Westerink, O. Verscheure, and P. Frossard. Securing Media for Adaptive Streaming. *In Proc. of ACM Multimedia*, November 2003.

[113] Nist Net Network Emulator, http://snad.ncsl.nist.gov/itg/nistnet/