Path Diversity Media Streaming over Best Effort Packet Switched Networks

by

Thinh P.Q. Nguyen

M.S. (University of California at Berkeley) 2000
B.S. (University of Washington) 1995

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Electrical Engineering
and Computer Sciences

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Avideh Zakhor, Chair
Professor Kannan Ramchandran
Professor Stanley Klein

Fall 2003

The dissertation of Thinh P.Q. Nguyen is approved:

Chair Date

Date

Date

University of California at Berkeley

Fall 2003

Path Diversity Media Streaming over Best Effort Packet Switched Networks

by

Thinh P.Q. Nguyen

Abstract

Path Diversity Media Streaming over Best Effort Packet Switched Networks

by

Thinh P.Q. Nguyen

Doctor of Philosophy in Engineering-Electrical Engineering and Computer Sciences
University of California at Berkeley

Professor Avideh Zakhor, Chair

Recent years has witnessed phenomenal growth of the Internet. According to Internet Software Consortium, number of hosts has reached 171 millions in January 2003 versus 71 millions in January 2000 and 5.8 millions in January 1995. Part of this explosive expansion is the proliferation of multimedia data such as image, audio, and video on the Internet. The current "best-effort" Internet, however, does not guarantee Quality of Service (QoS) such as minimum bandwidth, packet loss rate, and delay which are critical to many multimedia applications. As such, many significant challenges remain to design and deploy delay sensitive multimedia applications over the Internet effectively. In this dissertation, we develop a *path diversity* framework for concurrent media streaming to a receiver using multiple routes. Without requiring QoS, our framework improves the quality of the streamed media via multiple routes created using either multiple senders or relay nodes, in order to increase available bandwidth, reduce packet loss and delay. Our *path diversity* framework combines

many approaches from the network architecture and protocols, to source and channel coding to in order to improve the overall quality of the streamed media.

From an architecture point of view, our proposed path diversity framework combats packet loss, delay, and insufficient bandwidth for pre-recorded streaming media by sending packets simultaneously from multiple senders to a single receiver. For interactive and live streaming applications, the path diversity framework allows a single sender to send packets simultaneously on both default and redundant paths to the receiver. To create a redundant path, the sender sends packets to the appropriate relay node, which then forwards the packets to the receiver. The relay node selection algorithm is designed to ensure that packets traveling through the relay node take a different underlying physical path than that of the default path between the sender and receiver, hence, providing redundancy and protection against network outages and congestion.

We also develop a transport protocol to synchronize simultaneous media streaming to receiver via multiple routes. In particular, the protocol employs the rate allocation and packet partition algorithms. The rate algorithm determines the sending rate on each route in order to minimize the packet loss, while the packet partition algorithm ensures that each packet is sent by one and only one sender and at the same time, minimizes startup delay.

From a channel coding perspective, we show theoretically and experimentally that using Forward Error Correction (FEC) in streaming the media simultaneously over multiple mostly independent routes at appropriate sending rates is more effective than using FEC with the traditional uni-path streaming. We also provide the optimal strategy for using FEC under various network conditions.

From a source coding point of view, we design a network adaptive matching pursuits based multiple description video coding scheme for our proposed path diversity framework. Multiple description coding is an error resilient source coding scheme that generates multiple encoded bitstreams of the source with the aim of providing an acceptable reconstruction quality of the source when only one description is received, and improved quality when multiple descriptions are available. Our network adaptive multiple description matching pursuits scheme is designed to optimally split the source into descriptions that are adapted to the network characteristics of each route, hence providing superior visual quality.

Professor Avideh Zakhor
Dissertation Committee Chair

For my mom, Phung Thi Ngoc Hieu,
My constant source of love and encouragement,
For my late dad, Nguyen Huu Thong,
The guiding light of my life.

Contents

List of Figures **v**

List of Tables **viii**

1 Introduction **1**
1.1 Best-Effort Internet . . . 2
1.2 Approaches to multimedia streaming over the Internet . . . 6
1.2.1 Asynchronous Transfer Mode . . . 6
1.2.2 Integrated and Differentiated Services . . . 9
1.2.3 Multimedia streaming over best-effort networks . . . 11
1.3 Thesis Contributions . . . 16

2 Path Diversity Media Streaming Using Multiple Senders **19**
2.1 Assumptions . . . 21
2.2 Protocol Overview . . . 23
2.2.1 Transport Protocol . . . 23
2.2.2 Bandwidth Estimation . . . 24
2.3 Rate Allocation Algorithm . . . 25
2.4 Packet Partition Algorithm (PPA) . . . 28
2.4.1 Basic Description of PPA . . . 28
2.4.2 Practical Considerations with Packet Partition Algorithm . . . 33
2.5 Simulations and Experimental Results . . . 37
2.5.1 NS Simulations . . . 38
2.5.2 Internet Experiments . . . 45
2.6 Conclusions . . . 49

3 Rate Allocation with Forward Error Correction **50**
3.1 Introduction . . . 50
3.2 Assumptions . . . 52
3.3 Erasure Codes . . . 53
3.4 Network Model . . . 54
3.5 Optimal Rate Allocation . . . 58

3.5.1 Numerical Characterization 61
3.5.2 Sensitivity Analysis of Optimal Sending Rate 64
3.6 Simulation and Experiment Results 68
3.6.1 NS Simulations 68
3.6.2 Internet Experimental Results With Artificially Induced Packet Loss 71
3.6.3 Actual Internet Experiments 74
3.7 Conclusions 80

4 Path Diversity System (PDS) for Single Sender 81
4.1 Introduction 81
4.2 Related Work 83
4.3 Path Diversity System (PDS) 85
4.3.1 System Description 85
4.3.2 Redundant Path Selection 89
4.4 Simulation Results 94
4.4.1 Simulation Results for Hop Counts, Disjointness and Latency of Redundant Path 94
4.4.2 System Performance using Forward Error Correction 101
4.4.3 System Performance using Multiple Description Coding 107
4.4.4 Internet Experiments 111
4.5 Conclusions 113

5 Matching Pursuits Based Multiple Description Coding for Lossy Environments 118
5.1 Introduction 118
5.2 MP and MP-MDVC 122
5.2.1 Overview of MP 122
5.2.2 Overview of MP-MDVC 122
5.3 Fast algorithm MP-MDVC 124
5.3.1 Problem Formulation 124
5.3.2 Solution to Optimization Problem 127
5.3.3 Practical Implementation 130
5.4 Experimental Results 132
5.5 Conclusions 135

6 Summary and Future Work 138
6.1 Summary 138
6.2 Recommendations for Future Work 141
6.2.1 Scalable Video 141
6.2.2 Requesting Additional Servers 142
6.2.3 Peer to Peer Streaming 143

Bibliography **145**
A Chapter 2 Appendix 154
A.1 Appendix:Proof of optimality for rate allocation algorithm . . 154
A.2 Protocol Responsiveness 156
A.3 Throughput Reduction due to Delay Differences between Senders157
B Chapter 3 Appendix 162
B.1 Procedure for computing $P(m, k, N_m)$ 162
B.2 Online Estimation of Network Parameters 164

List of Figures

2.1 *Edge architecture.* . . . 20
2.2 *Multiple sender architecture.* . . . 21
2.3 *High level description of distributed streaming framework.* . . . 23
2.4 *Sending rate allocation for four senders.* . . . 26
2.5 *Illustration of packet partition algorithm.* . . . 33
2.6 *Illustration of choosing synchronization sequence number.* . . . 37
2.7 *Simulation configuration.* . . . 39
2.8 *Loss rate of two senders in adaptive scenario one.* . . . 42
2.9 *Send rate of two senders in adaptive scenario one.* . . . 42
2.10 *Throughputs of two senders in adaptive scenario one.* . . . 43
2.11 *Throughputs of two senders in non-adaptive scenario two.* . . . 43
2.12 *Loss rate of scenario two over that of scenario one.* . . . 44
2.13 *MSE for adaptive and non-adaptive scenarios.* . . . 44
2.14 *Loss rate versus time of simultaneous streaming from Sweden and Indiana to Berkeley.* . . . 47
2.15 *Throughput versus time of simultaneous streaming from Sweden and Indiana to Berkeley.* . . . 47
2.16 *Histogram of the difference in sequence number of consecutive received packets.* . . . 48
2.17 *Histogram of the delay difference between consecutive packets.* . . . 48

3.1 *Two-state Markov model.* . . . 55
3.2 *Hong Kong to U.C. Berkeley, (a) rate = 200kbps; (b) rate = 400kbps; (c) rate = 800kbps* . . . 57
3.3 *Probability of irrecoverable loss for various of FEC protection levels as the function of average bad times of route B, using optimal partition for two senders for (a) scenario X and (b) scenario Y.* . . . 64
3.4 *Optimal partition for various FEC protection levels as the function of average bad times of route B for (a) scenario X and (b) scenario Y;N_A denotes the number of packets per 30 sent on route A.* . . . 65

3.5 *Ratio of irrecoverable loss probability of sending all packets using only one sender to that of using two senders as the function of average bad times of route B for (a) scenario X and (b) scenario Y.* 65
3.6 *(a) Irrecoverable probability as the function of different partitions of sending rate between two senders; (b) irrecoverable loss probability ratios between sending all packets on route A over 50-50 and optimal rate splits.* 67
3.7 *Simulation setup for various number of senders* 69
3.8 *NS simulation showing number of irrecoverable loss events for differnet number of senders (a) one sender; (b) two senders; (c) three senders; (d) four senders* 70
3.9 *Internet simulations showing the number of lost packet per FEC block of 60 packets vs. packet sequence for (a) streaming from Belgium alone; (b) streaming from Belgium and Sweden; (c) Throughputs of two senders; (d) Histogram of variation in packet order.* 74
3.10 *Actual Internet experiments showing the benefits of path diversity video streaming over conventional approach.* 77
3.11 *Typical images at an irrecoverable loss event for (a) conventional video streaming and (b) path diversity video streaming.* 78

4.1 *System architecture; bold dashed and solid lines denote virtual and physical paths; The thin vertical solid lines connecting circles and squares represent the correspondence between virtual nodes and physical routers.* 88
4.2 *Two-level hierarchical topology* 95
4.3 *Flat topology* 96
4.4 *Percentage of shared links between the redundant and the default paths.* 98
4.5 *Latency of redundant path over the latency of default path.* 99
4.6 *Number of hops of the redundant path over the number of hops of the default path.* 101
4.7 *Cumulative distribution of shared links for various network topologies.* 102
4.8 *Simulation configuration for two disjoint redundant and the default paths.* 103
4.9 *Sending packets using traditional default path.* 104
4.10 *Sending packets using both redundant and default paths without shared link between them.* 105
4.11 *Sending packets using both redundant and default paths with one shared link between them.* 106
4.12 *The ratio of average loss rates using one path over that of using both redundant and default paths with various number of shared links between them.* 106
4.13 *PSNR versus the loss rate per link for MDC on 2 paths and SDC on single path (a) Coast, (b) Foreman, and (c) News.* 110
4.14 *PSNR versus the loss rate for SDC with uni-path and multi-path.* 111

4.15 *Internet configuration for experiment one.* 114
4.16 *Foreman sequence (a) Delay as a function of time, (b) PSNR as a function of time, and (c) Zoom in portion of the delay* 115
4.17 *Internet configuration for experiment two.* 116
4.18 *Foreman sequence (a) Delay as a function of time, with zero delay denoting actual packet loss; (b) PSNR as a function of time.* 116
4.19 *Typical images at a loss event for (a) SDC with uni-path streaming and (b) MDC with path diversity streaming.* 117

5.1 *Three loop structure of MP-MDVC.* 124
5.2 *Actual and predicted PSNR for Foreman and Hall.* 133
5.3 *PSNR0 and PSNR1 for Foreman and Hall.* 134
5.4 *PSNR0 and PSNR1 for Hall as a function of outage probability.* 135
5.5 *Average number of G atoms for Hall as a function of outage probability.* 136
5.6 *Expected PSNR as a function of outage probability; (a) Foreman and Hall; (b) Mom and News.* 137

A.1 *Protocol responsiveness for various values of* γ*; (a)* $\gamma = 20$*; (b)* $\gamma = 40$*; (c)* $\gamma = 100$*; (d)* $\gamma = 200$. 160
A.2 *Illustration of throughput reduction* 161
B.1 *Illustration of network parameters estimation; 0 and 1 indicate received and lost packets, respectively.* 167
B.2 *Protocol responsiveness for* $\lambda = 0.85$*; (a) sending rate triggered at the receiver; (b) corresponding throughputs at the receiver.* 167
B.3 *Protocol responsiveness for* $\lambda = 0.9$*; (a) sending rate triggered at the receiver; (b) corresponding throughputs at the receiver.* 168
B.4 *Protocol responsiveness for* $\lambda = 0.96$*; (a) sending rate triggered at the receiver; (b) corresponding throughputs at the receiver.* 168
B.5 *Protocol responsiveness for* $\lambda = 0.98$*; (a) sending rate triggered at the receiver; (b) corresponding throughputs at the receiver.* 169

List of Tables

2.1 *Notations for packet partition algorithm.* 31
2.2 *NS Simulation parameters* 39

3.1 *Notations for rate allocation algorithm.* 59
3.2 *Parameters chosen for numerical characterization in two scenarios X and Y.* 62
3.3 *Irrecoverable loss reduction using two senders for various sending rates and FEC levels.* 79

4.1 *Information for various topologies* 96

5.1 *Optimization notation* 125

A.1 *Simulation parameters for various* γ. 157
B.1 *Notations for computing* $P(m, kN_m)$. 162
B.2 *Network and protocol parameters.* 166

Acknowledgements

This dissertation is not possible without the love and support from my family. When I was a boy, I often annoyed my neighbors with the intolerable sounds of my violin practice, but my mom was always there to defend and encourage her young violinist. When I left my full-time job for graduate school, it was her unwavering support that enabled me to complete this dissertation. Whether I was in elementary school or college, my mom has always been there to encourage me to follow the unexplored paths. For that, I am deeply grateful to my mom. I also would like to thank Bac Phuong for believing in me and for his constant support during my years at Berkeley. Even as he was recovering from the surgery, his first words were of my progress. I would like to thank my brother, Thang Nguyen, his wife, NgocAnh Nguyen, and her mom, Bac Diem, for their love, support, and the delicious weekend foods. My heartfelt thanks also go to my sister Huyen Nguyen and her husband John Martin for their words of advice and encouragement throughout my graduate studies. My many thanks go to my two younger brothers, Thieu and Thuc Nguyen for their companionship.

I am deeply indebted to my advisor, professor Avideh Zakhor for her support, guidance, and encouragement throughout my years at Berkeley. She has given me the freedom to do research in my own way, yet she has taught me the importance of rigorous scientific approach. I am thankful for her insightful suggestions, critical discussions, and great attention to details. Without them, this dissertation would not be possible. I am also thankful for the countless hours she spent on editing our papers and helping me to become a better writer.

I would like to thank professor Kannan Ramchandran, professor Ion Stoica, and professor Stanley Klein for their helpful suggestions during my qualifying examination and throughout the course of my research.

I am grateful to friends at Berkeley for many fruitful and interesting discussions. In particular, I would like to thank Samson Cheung for the campus lunch walks full with conversations ranging from our research to physics and philosophy. I have been blessed to be around many bright and interesting friends in our Video and Image Processing Lab: Vito Dai, Dan Tan, Christian Frueh, Dave Harrison, Siddarth Jain, Minghua Chen, Wei Wei, Puneet Mehra, Ali Lakhia, and Russel Sammons. I also would like to thank Sang Kang and Christophe De Vleeschouwer for helping me with the remote accounts for my experiments. My heartfelt thanks go to Ruth Gjerde in the EECS graduate office for all her helpful advices.

I would like to acknowledge the financial support for my research from NSF grant CCR-9979442 and AFOSR contract F49620-00-1-0327.

Finally, I am most grateful to my girlfriend, Sea Chu for an infinite number of reasons. Most important of all, I have learned that with her love and support, all dreams are realizable.

Chapter 1

Introduction

Recent years has witnessed phenomenal growth of the Internet. According to Internet Software Consortium, number of hosts has reached 171 millions in January 2003 versus 71 millions in January 2000 and 5.8 millions in January 1995. Part of this explosive expansion is the proliferation of multimedia data such as image, audio, and video on the Internet. The current “best-effort” Internet, however, does not guarantee quality of service such as minimum bandwidth, packet loss rate, and delay which are critical to many multimedia applications. As such, many significant challenges remain to design and deploy delay sensitive multimedia applications over the Internet effectively. To understand these challenges, in Section 1.1, we review the current best-effort Internet model and its pitfalls with respect to multimedia streaming. In Section 1.2, we describe several approaches to alleviate these problems. In Section 1.3, we outline our thesis contributions to media streaming over best-effort networks.

1.1 Best-Effort Internet

The Internet is a global communication system composed of computer networks based on the Internet Protocol (IP), a common interface between communication hosts. The Internet was conceived by the Advanced Research Projects Agency (ARPA) of the U.S. government in 1969, and was first known as the ARPANET. The original goal of ARPANET was to allow research computers at universities to exchange information among each other, and maintain their communications in the event of a military attack or other disaster. Thus, the ARPANET was designed based on distributed communications using "packet-switched" concept, the idea first proposed by Paul Baran of the RAND Corporation in the 1960s [1]. Today, the ARPANET has evolved into the Internet, a self-sustaining mesh-like packet-switched network connecting hundreds of millions of computers worldwide. The success of the Internet can be attributed to its design based on "best-effort" principle. We now provide a brief introduction to the existing "best-effort" designs from the network infrastructure to the Internet protocols.

Since computers are connected by point-to-point links in a mesh-like network, a number of important questions needed to be answered in order to sustain the growth and ensure effective use of network resources [2]:

- *Addressing:* How to assign address for each computer on the network?
- *Routing:* How and which path between two computers should be chosen to send data?
- *Flow control:* How to deal with congestion in the network?
- *Error control:* How to deal with transmission errors?

Early architects and designers of the Internet realized that in order to build a network that enables growth, *interoperability* among heterogeneous networks and *robustness* are critical. Therefore, they devised an uniform communication interface to facilitate *interoperability* among networks, and defined the minimal functionality required of individual networks to achieve *robustness* [3][4]. Today, this uniform communication interface is the Internet Protocol (IP), which supervises *addressing* and *routing* of packets, and the minimal functionality designs result in simple and extremely fast routers[1]. The consequence of this design is that routers are not sophisticated enough to automatically retransmit the lost packets per link, or to guarantee certain requirements such as the bandwidth, delay, and loss rate along the links between the sender and the receiver. This preference of simplicity and speed over sophisticated functionality has been the guiding principle behind the "best-effort" design since the early 1980's.

In addition to the *interoperability* and *robustness*, successful deployment of the

[1]routers are the intermediate physical nodes in the network that are responsible for packet processing and routing.

Transmission Control Protocol (TCP) over IP for *flow control* and *error control* enables a tremendous growth in number of hosts on the Internet, by providing stability and effective use of network resources [5][6]. Together the Internet Protocol and Transmission Control Protocol (TCP/IP) are used to deliver most of the traffic on the Internet, such as web traffic and emails.

As noted earlier, the "best-effort" design results in routers with simple functionality, therefore hop-by-hop reliability is not possible. Although hop-by-hop reliability is arguably more efficient in terms of network usage, the cost of implementing complex routers, and therefore less scalable networks, make the efficiency argument less attractive. On the other hand, the mechanisms for end-to-end reliability are implemented only at the two end hosts, and therefore, end-to-end reliability is arguably more cost effective from a system design perspective [7]. As a result, TCP provides end-to-end reliability by retransmitting lost packets, and carries 89% of the current traffic on the Internet.

However, one consequence of providing end-to-end reliability by retransmitting lost packets is the associated increase in delay and delay-jitter. Delay is the time interval between the sending time of the packet at the sender and the arrival time of the same packet at the receiver. Delay-jitter characterizes the amount of variation in delay between successive packets, i.e. higher packet delay-jitter indicates higher variation in packet delay. For delay insensitive data such as FTP and emails, TCP which uses automatic retransmission of lost packets, is used to transport data since the reliability is much more important than low delay and delay-jitter. However, for delay sensitive applications such as video or audio streaming, an increase in delay or delay-jitter may result in unacceptable quality of the media. For this reason, TCP is typically not used in multimedia applications. To support delay sensitive and loss tolerant applications, the Internet designers provided User Datagram Protocol (UDP) which avoids automatic retransmission of lost packets in order to reduce packet delay at the expense of reliability. Consequently, UDP has become the building block of many multimedia protocols [8][9][10]. With UDP, the application directly handles lost packets since the application *knows* the importance of each lost packets, hence it can use appropriate techniques to correct or reduce errors. This follows directly from the ALF (Application Level Framing) principle which states that *data should be organized into units (packets) that make the most sense for the application* [11]. As early as the 1970s, researchers on audio transport over packet switch networks have observed that the application can effectively conceal errors caused by lost audio packets. On the other hand, automatic retransmissions of lost audio packets by transport layer often result in lower quality due to delay and delay-jitter [12][4]. Although using UDP reduces delay and delay jitter for multimedia applications, UDP is still not suitable for worldwide deployment since it does not provide *congestion control.* In other words, UDP does not provide a mechanism for adjusting the sending rate appropriately in presence of network congestion. This lack of rate control can potentially disrupt TCP flows, and lead to congestion collapse, where only a small percentage of packets are successfully transmitted. In Section 1.2, we review several approach for dealing with multimedia streaming over the Internet.

1.2 Approaches to multimedia streaming over the Internet

There are many approaches to multimedia streaming ranging from network to protocol, to source and channel coding. In this section, we review the advantages and disadvantages of each approach. We first begin with a network infrastructure approach, particularly, Asynchronous Transfer Mode (ATM).

1.2.1 Asynchronous Transfer Mode

The design of ATM was initiated by AT&T. Its design philosophy is based on virtual circuit switching. In traditional circuit switching networks such as the telephone network, the network sets up a fixed circuit between the two telephone terminals for a duration of the conversation and releases the circuit when the call terminates. This approach guarantees certain criterion, e.g., minimal delay and bandwidth for the active call during its duration. However, the drawback of this approach is that resources cannot be shared by any other communication during the call duration. Similar to traditional circuit switching, ATM uses virtual circuit switching, in which every connection must be set up and controlled by the network in order to reserve the resources that each connection needs. The network blocks a new connection when network resources are not available to accommodate the new connection request. This is referred to as admission control. However, virtual circuit switching does not reserve sufficient resources for all connections to send packets simultaneously at their peak transmission rates, rather, virtual circuit switch systems characterize the aggregate demand and over-subscribe resources to the extent possible while retaining statistical guarantees for bandwidth, delay, and loss rate. To show the difference between the traditional and virtual circuit switching approaches, consider a 100 Mbps link. Using traditional circuit switching, only 50 connections with peak rate of 2 Mbps are allowed through the link at any given time. On the other hand, using virtual switching circuit approach, the network may allow more than 50 connections, depending on the average rate of each connection. Assuming the average rate of each connection is 1 Mbps, the network may allow up to 80 connections through the 100 Mbps link and still achieve low loss rate since it is unlikely that the sum of the transmission rates of all 80 connections exceeds 100 Mbps long enough for the link to lose packets. The bandwidth gain of the virtual circuit switching in this example is called *statistical multiplex gain* and is the main benefit for using packet-switched networks. Based on the above discussion, the virtual circuit switching approach provides both the *statistical multiplex gain* of best-effort packet-switched networks and the tight control of resource allocation for quality of service in circuit switched networks.

Tight control over resource allocation in ATM network allows a wide range of service classes for different applications. The services classes are based on their *quality of service* (QoS) requirements. ATM service models are divided into many service classes: constant bit rate (CBR), real-time Variable Bit Rate (rt-VBR), non-real-time Variable Bit Rate (nrt-VBR), Available Bit Rate (ABR), and Unspecified Bit Rate (UBR). CBR and rt-VBR services are intended for real time applications such as audio and video conferencing, nrt-VBR service are for non-real time streaming applications, ABR and UBR services are for "best-effort" applications such as email and FTP. Based on the above discussion, one may conclude that ATM can effectively solve the multimedia streaming problem by tightly controlling the resource allocation at the network level. However, four issues have limited the wide deployment of ATM: *incompatibility*, *complexity*, *scalability*, and *unreliability*. Most Internet applications use TCP/IP and are incompatible with ATM in the sense that TCP/IP has no interface to directly exploit the benefits of ATM. Hence, the wide deployment of ATM is unlikely in the near future. Due to the tight control of resource allocation, the ATM switch must be aware of the connections and control protocols, making each ATM switch complex, and thus costly to scale. Because of the lack of scalability, the existing ATM switches are usually used in the Internet backbone only, where the ATM operations are performed on a per-aggregate of flows rather than per-flow basis. In addition, the complexity of ATM switches makes them difficult to implement and operate correctly, and thus makes them unreliable. A failure of an ATM switch is likely to have a large impact on many connections since each ATM switch stores the states for many connections. Many also argue that a fast mean and lean IP network is better since complexity of ATM network necessarily results in a slower network.

1.2.2 Integrated and Differentiated Services

Due to the shortcomings of ATM networks as described previously, the Internet Engineering Task Force (IETF) has considered a number of architecture extensions to the best-effort Internet in order to provide QoS for multimedia streaming applications. The first architecture extension is the Integrated Service (IntServ) model [13] which attempts to provide end-to-end QoS guarantees in terms of bandwidth, packet loss rate, and delay, on a per-flow basis. IntServ communicates a connection's QoS requirements using Resource Reservation Protocol (RSVP) [14]. When a host requests a specific QoS for its data stream, RSVP is used to deliver the request to each router along the path of the data stream and to maintain router and host states. The maintenance of router and host states in RSVP is necessary for the admission control modules at each hop along the path to allow or block the new request based on the available resources. A new end-to-end QoS request is granted only when the QoS is met at each hop along the path. Although, IntServ using RSVP-based service architecture is attractive for providing QoS over best-effort networks, it is still highly complex and costly due to maintenance of router and host state for per-flow QoS and as such, does not scale to large number of flows. As a result, IntServ model has not been widely deployed. This eventually led the IETF to consider the Differentiated

Service (DiffServ) model [15].

The DiffServ model is specifically designed to achieve low complexity and easy deployment at the cost of less stringent QoS guarantee than IntServ. Under DiffServ, service differentiation is no longer provided on a per-flow basis, rather QoS is provided on a per-class basis. DiffServ divides flows into classes with flows belonging to the same class, sharing the same QoS requirements. Hence, the maintenance of router and host states on a per-flow basis is no longer required, reducing the complexity and cost for deployment of DiffServ. To provide QoS on per-class basis, each packets contains a tag indicating the level of its QoS. Each router in the DiffServ networks then provides the same treatment for the packets having the same tags, regardless of where they originate. As an example, all the video packets have the same tag, and therefore, are given the same preferential treatment in terms of delay and loss as compared to FTP packets. At a glance, DiffServ seems to be a promising approach for multimedia streaming over the Internet. However, three short-comings of DiffServ have limited its wide deployment. First, as mentioned previously, DiffServ uses preferential treatment for low complexity, rather than strict admission control policy for QoS at each hop, and thus end-to-end QoS per flow is often not guaranteed. The second short-coming of DiffServ is interaction among different ISPs through Service Level Agreements (SLAs). Each ISP has different implementations and definitions for the same QoS, making it hard to maintain the QoS guarantee for a flow crossing ISPs. Finally, although far less intrusive than ATM or IntServ, the DiffServ model requires modification to network routers, making its proliferation less likely in the near future.

Due to the above reasons, most of multimedia applications over the Internet today still rely on the best-effort networks, and the quality of the streamed media depends on the application's treatment of packet loss and delay. In Section 1.2.3 we describe several techniques for multimedia streaming over best-effort networks.

1.2.3 Multimedia streaming over best-effort networks

We first begin with multimedia protocols. As discussed previously, TCP is typically not used in multimedia applications. Instead, most multimedia protocols use UDP [8][9][10]. Thus, the application manages packet loss appropriately. The IETF proposed RTP (Real Time Protocol) for transporting audio and video data based on UDP. To allow the applications to respond to network conditions, RTP also reports packet loss rate to the applications [16]. Other promising protocol for multimedia streaming is the TFRC (TCP-friendly Rate Control) proposed in [8]. Unlike TCP, TFRC does not reduce the sending rate by half when encountering a packet loss, rather, it calculates the sending rate as a function of smooth loss rate and round trip time. As a result, the sending rate of TFRC is typically much smoother than TCP, hence, more suitable for multimedia streaming. There are also other multimedia protocols such as RAP [9] which increases and decreases the sending rate by a small amount according to the observed loss rate and round trip time. These protocols also produce smoother sending rates than TCP. Since multimedia protocols are not reliable as they use UDP as the building blocks, applications need to handle the packet loss. For delay sensitive applications, retransmission of lost packets is often not used due to additional retransmission delay. Instead, channel and source coding techniques are used to recover the packet loss or to conceal its effect.

From the channel coding approach, FEC has been widely employed for transporting audio and video over the Internet [17]. Unequal error protection techniques which provide protection levels according to the importance of the video bits have been proposed [18]. For example, motion vectors in the video bitstream are generally considered more important than the residual bits, hence, they are protected with stronger FEC code. FEC techniques are also used in multicast streaming for situations with one sender and many receivers. Multicast reduces the network bandwidth by not sending duplicate packets on the same physical link [19]. The use of FEC in a multicast scenario is to avoid the NACK implosion associated with retransmission methods. The NACK implosion is caused by the flood of loss acknowledgment messages sent from a large number of the receivers in a multicast group to the sender. This scenario is likely to occur since there are typically hundreds to thousands of receivers in a multicast group, and at any given point in time, there can be a large number of loss acknowledgment messages from different receivers, making the sender unable to respond effectively.

From the source coding approach, error-resilient audio and video codecs have been proposed to conceal the effect of packet loss at the expense of coding efficiency [20][21]. The error-resilient codecs typically rely on the temporal and spatial similarities between and within frames to extrapolate the missing information in the video bitstream due to packet loss. To deal with time-varying bandwidth of the Internet, there has been work on scalable video codecs that adapt the bit rate, i.e., quality, to the current available bandwidth [20][21]. Also, Multiple Description Coding (MDC) techniques have also been proposed to combat packet loss [22][23][24][25][26]. MDC is an error resilient source coding scheme that creates multiple descriptions of the source with the aim of providing an acceptable reconstruction quality when only one description is received, and improved quality as more descriptions become available.

From the network architecture perspective, Content Delivery Network (CDN) companies such as Akamai use the edge architecture to achieve better load balancing, lower latency, and higher throughput. The edge architecture reduces latency by moving content to the edge of the network in order to reduce round-trip time and to avoid congestion in the Internet. Companies such as FastForward Networks and Akamai strategically place a large number of the servers around the Internet so that each client can choose the server that results in shortest round-trip time and least amount of congestion.

All of these approaches assume a single fixed route between the receiver and the sender throughout the session. If the network is congested along that route, video streaming suffers from high loss rate and jitter. Even if there is no congestion, as the round-trip time between the sender and the receiver increases, the TCP throughput may reduce to unacceptably low levels for streaming applications. Furthermore, authors in [27][28] have revealed the ill-behaved systematic properties in Internet routing, which can lead to sub-optimal routing paths.

Based on these, it is conceivable to make content available at multiple sources so that the receiver can choose the "best" sender based on bandwidth, loss, and delay. In a way, this is what Akamai does by moving the server closer to the client. However, if no sender can support the required bit rate needed by the application, it is conceivable to have multiple senders or to use multiple mostly independent routes to simultaneously stream video to a single receiver in order to effectively provide the required throughput. Having multiple senders or routes, is also a diversification scheme in that it combats unpredictability of congestion in the Internet. If the route between a particular sender and the receiver experiences congestion during streaming, the receiver can redistribute streaming rates among other senders,thus resulting in smooth video delivery. This is the motivation for our proposed *path diversity* framework.

Many diversity schemes have been proposed in wireless literature, ranging from frequency and time, to spatial diversity [29]. In wired networks, path diversity was first proposed in [30] and the theoretical work on information dispersion for security and load balancing was proposed in [31]. Recently, there have been other works dealing with simultaneous downloading of data from multiple mirror sites. If the data is not delay sensitive, it is possible to use multiple TCP connections to different sites, with each TCP connection downloading a different part of the data. For example, the authors in [32] use Tornado codes to download data simultaneously from multiple mirror sites. More recently, Digital Fountain has used an advanced class of linear-time codes to enable the receivers to receive any N linear-time coded packets from different senders, so as to recover N original data packets. There has also been study on the throughput of TCP connection using multiple paths in [33]. In [34], the authors proposed *CoopNet*, a tree structure for delivering video to the receivers from multiple servers. Also Peer-to-Peer (P2P) file sharing system such as *Kazaa* allows downloading the media files simultaneously from multiple participating members. Another Peer-to-Peer system with adaptive layered streaming has also been proposed in [35]. Also, the performance gain of using multiple servers together with MDC over traditional single server approach is compared and analyzed in [36][37]. Recently, rate distortion optimization in the framework of path diversity has also been explored in [38]. Finally, authors in [39] have shown substantial improvement for real-time voice communication over the Internet using path diversity together with MDC, and sophisticated playback schedule.

In this dissertation, we advocate a novel approach for multimedia streaming over best-effort networks using *path diversity*, i.e, multiple paths. In subsequent chapters, we show the path diversity approach offers substantial improvement over traditional uni-path approach. We now begin with the outline of our thesis contributions in the next section.

1.3 Thesis Contributions

In this dissertation, we develop a *path diversity* framework for concurrent media streaming to a receiver using multiple routes. Without requiring QoS, our framework improves the quality of the streamed media via multiple routes created using either multiple senders or relay nodes, in order to increase available bandwidth, reduce packet loss and delay. Our *path diversity* framework combines many approaches from the network architecture and protocols, to source and channel coding to improve the overall quality of the streamed media.

In Chapter 2, we present a *path diversity* framework for multimedia streaming using multiple senders in order to achieve higher throughput, and to increase tolerance to loss and delay due to network congestion. In our framework, multiple senders simultaneously stream the media, e.g., video to a receiver via mostly disjoint paths. The advantage of this framework is that it combats unpredictability of congestion in the Internet. If the route between a particular sender and the receiver experiences congestion during streaming, the receiver can redistribute streaming rates among other senders, thus resulting in smooth video delivery. Another advantage is that, using multiple routes can potentially provide higher bandwidth than single route, hence, higher quality media can be streamed. Within this framework, we propose a receiver-driven protocol in which, the receiver coordinates simultaneous transmissions from multiple senders through the control packets sent to all senders from the receiver. The proposed protocol employs two main algorithms: the rate allocation and packet partition algorithms. The rate algorithm determines the sending rate on each route in order to minimize the packet loss, and to also share bandwidth fairly with other traffic; the packet partition algorithm ensures that each packet is sent by one and only one sender, and at the same time, minimizes startup delay. We then examine the feasibility of such a framework and the performance of the proposed protocol.

In Chapter 3, we extend the rate allocation algorithm in Chapter 2 to incorporate FEC to combat bursty packet loss on the Internet. We show theoretically and experimentally that using FEC in streaming the media simultaneously over multiple mostly independent routes at appropriate sending rates is more effective than using FEC with the traditional uni-path streaming. We also provide the optimal strategy for using FEC under various network conditions.

The path diversity system using multiple senders proposed in Chapter 2 cannot be used for live streaming or interactive applications. In Chapter 4, we propose a path diversity system that allows single a sender to send packets simultaneously on both default and redundant paths to the receiver. To create a redundant path, the sender sends packets to the appropriate relay node, which then forwards the packets to the receiver. The relay node selection algorithm is designed to ensure that packets traveling through the relay node take a different underlying physical path than that of the default Internet path between the sender and receiver, hence providing redundancy and protection against network outages and congestion. Experiments and simulations demonstrate that our path diversity system can improve the streamed media quality significantly over the traditional uni-path streaming techniques.

For certain scenarios, e.g. long network outages, using MDC techniques could potentially be better than FEC. In Chapter 5, we design a network adaptive matching pursuits based multiple description video coding scheme for our path diversity system. Our network adaptive multiple description matching pursuits scheme is designed to optimally partition the source into descriptions that are adapted to the network characteristics of each route, hence providing superior visual quality.

Chapter 2

Path Diversity Media Streaming Using Multiple Senders

In this chapter, we propose a *path diversity* framework for streaming video from multiple senders simultaneously to a single receiver in order to achieve higher throughput, and to increase tolerance to loss and delay due to network congestion. Our solution combines approaches from different perspectives including system architecture and transport protocols. From the systems perspective, we expand the edge architecture as shown in Figure 2.1 to allow simultaneous video streaming from multiple senders. Figure **??** shows our approach of using multiple senders, in contrast to the edge architecture where only one server is responsible for streaming video to its nearest clients. From protocol perspective, we use a TCP friendly protocol to coordinate simultaneous transmission of video from multiple mirror sites to a single receiver effectively. There are two main algorithms in our protocol: The *rate allocation* and *packet partition* algorithms. The rate allocation allocate the sending rate among senders to minimize the packet loss while the packet partition algorithm prevents duplicate packets and minimizes the startup delay. Although, this work focuses primarily on video streaming, our protocol and architecture can accommodate other multimedia components such as audio as well.

The rest of this chapter is organized as follows. In Section 2.1, we state the assumptions and the scenarios for our path diversity framework. In Section 2.2, we describe our transport protocol, rate allocation, and packet partition algorithms. We show that under certain assumptions, our rate and packet allocation algorithms are optimal in terms of packet loss and delay. Next, we describe the simulation setup and discuss results in Section 2.5. Finally, we conclude in Section 2.6.

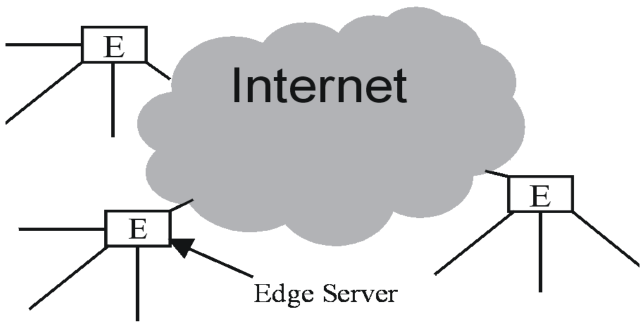


Figure 2.1: *Edge architecture.*

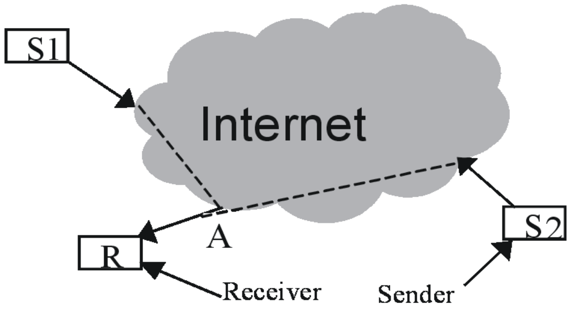


Figure 2.2: *Multiple sender architecture.*

2.1 Assumptions

To successfully stream video from multiple senders, we assume that the available aggregate bandwidth from all the senders to the receiver exceeds the required video bit rate. As an example, consider a network configuration shown in Figure 2.2. The available bandwidth from router A to receiver R, sender $S1$ to router A, and sender $S2$ to router A is assumed to be 2Mbps, 0.8Mbps, and 0.4Mbps, respectively. Router A can be an edge router of a particular network domain of a university or a company, and is assumed to be the only router receiver R is connected to. In this scenario, it is not possible to stream a 1Mbps video from sender $S1$ to receiver R since the available bandwidth for streaming video from $S1$ to R is only 0.8Mbps. However, it is possible to stream a 1Mbps video simultaneously from both senders $S1$ and $S2$ to receiver R since the aggregate bandwidth from both senders $S1$ and $S2$ to receiver R is 1.2Mbps which exceeds the required bit rate of 1Mbps. On the other hand, if the available bandwidth from router A to receiver R is only 0.9Mbps, then the link between A and R becomes a bottleneck, and video streaming at 1Mbps becomes infeasible both for multiple sender and single sender scenarios. We also assume that the routes from a client to the senders do not share the same congestion link. If there is congestion on a shared link between two senders, then changing the sending rate of one sender may adversely affect the traffic conditions of the other one. In this work, we assume that changing the sending rate of one sender does not affect the route conditions of others. As will be discussed in Section 2.3, this assumption prevents potential oscillations and instabilities in the sending rates computed by our rate allocation algorithm.

Based on the above assumptions, in streaming situations when the bottleneck is in the last hop, e.g., due to the physical bandwidth limitation of dial-up modem, our proposed distributed streaming approach is of little use. Indeed if a client is connected to the Internet through a low bandwidth connection, it is preferable to download the entire video in a non-real time fashion before playing it back. Our premise is that, asymptotically, as broadband connection to the Internet such as DSL or cable modem becomes prevalent, the limiting factor in streaming is packet loss and delay due to congestion along the streaming path, rather than the physical bandwidth limitations of the last hop. Finally, there has been work on detecting the shared congestion points of different routes [40] based on the correlation of packet loss and delay between routes. These correlations can be used ahead of time to improve the performance of our approach. 2.5. Finally, we conclude in Section 2.6.

2.2 Protocol Overview

2.2.1 Transport Protocol

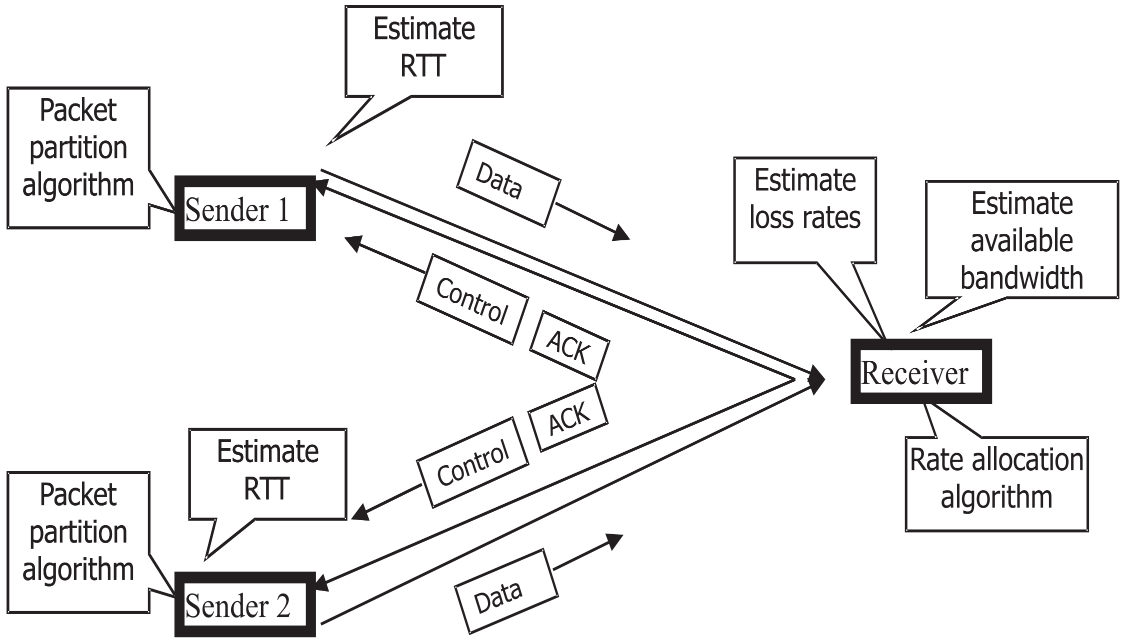


Figure 2.3: *High level description of distributed streaming framework.*

Our transport protocol is a receiver-driven one, in which the receiver coordinates transmissions from multiple senders based on the information received from the senders. Each sender estimates and sends its round trip time to the receiver. The receiver uses the estimated round trip times and its estimates of sender's loss rates to calculate the optimal sending rate for each sender. When the receiver decides to change any of the sender's sending rates, it sends an identical control packet to each sender. The control packet contains the synchronization sequence number and the optimal sending rates as calculated by the receiver for all senders. Using the specified sending rates and synchronization sequence number, each sender runs a distributed packet partition algorithm to determine the next packet to be sent. Figure 2.3 shows the block diagram of an example of a system deploying our approach.

2.2.2 Bandwidth Estimation

In our protocol, the receiver estimates available bandwidth for each sender based on the TCP-friendly rate control algorithm (TFRC) proposed in [8]. The TFRC protocol is designed to be fair with TCP traffic, and results in less fluctuation in sending rate than TCP does. It calculates the available bandwidth according to the following equation:

$$B = \frac{s}{R\sqrt{\frac{2p}{3}} + T_{rto}(3\sqrt{\frac{2p}{3}})p(1+32p^2)} \tag{2.1}$$

where B denotes the current available TCP-friendly bandwidth between each sender and the receiver, T_{rto} is TCP time out, R is the estimated round-trip time in seconds, p is the estimated loss rate, and s is the TCP segment size in bytes. The estimated round trip time is computed using the moving average of round-trip times over a fixed time interval. Similarly, the estimated loss rate is the ratio of number of lost packets over the total number of packets sent during a fixed time interval. The estimated bandwidth B forms an upper bound for the TCP-friendly sending rate.

2.3 Rate Allocation Algorithm

In our protocol, the receiver computes the optimal sending rate for each sender based on its loss rate and estimated available bandwidth. The problem of allocating optimal sending rate to each sender can be stated as follows. Let N be the total number of senders, and $L(i,t)$ and $S(i,t)$ be the estimated loss and sending rates, respectively for sender i over an interval $(t, t+\delta)$. Our goal is to find $S(i,t)$, $i = \{1...N\}$, that minimize the total lost packets during interval $(t, t+\delta)$ given by

$$F(t) = \sum_{i=1}^{N} L(i,t)S(i,t) \tag{2.2}$$

subject to

$$\begin{aligned} &0 \leq S(i,t) \leq B(i,t) \\ &\sum_{i=1}^{N} S(i,t) = S_{req}(t) \end{aligned} \tag{2.3}$$

where S_{req} is the required bit rate for the encoded video during the interval $(t, t+\delta)$, and $B(i,t)$ is the TCP-friendly estimated bandwidth for sender i during the interval $(t, t+\delta)$.

This optimization problem can be solved using the following algorithm. At time t, we sort the senders according to their estimated loss rates from lowest to highest. We start with the lowest loss rate sender and assign its sending rate to be its TCP-friendly estimated bandwidth as described in Equation (2.1). We then continue to assign to each sender its available bandwidth, beginning with the ones with lower loss rates and moving to the ones with higher loss rates, until the sum of their available bandwidths exceeds the bit rate of the encoded video. As an example, Figure 2.4 shows the rate allocation for four senders with the loss rates increasing from left to right. The height of each cylinder shows the available TCP-friendly bandwidth for that sender. As seen, since sender four has the highest loss rate and since sum of available bandwidth from sender one, two, and three is sufficiently large, sender four can afford to operate at "below capacity." Intuitively, the algorithm assigns more packets to senders with

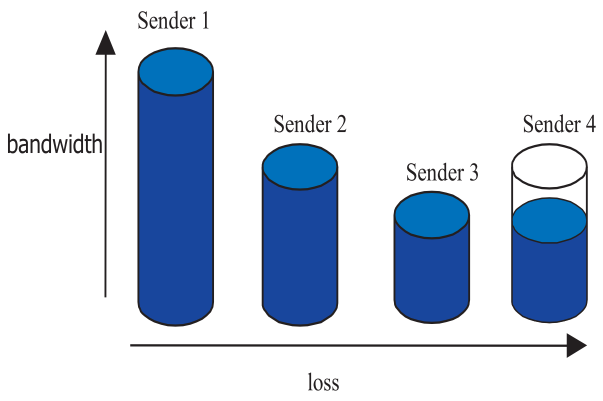


Figure 2.4: *Sending rate allocation for four senders.*

lower loss rates and fewer packets to the senders with higher loss rates. At the same time, each sender also satisfies the bandwidth constraints in Equation (2.3) in order to share bandwidth fairly with other TCP traffic. The algorithm minimizes $F(t)$, the number of lost packets during the interval $(t, t+\delta)$, given instantaneous feedback, and assuming that the estimated loss rate and TCP-friendly available bandwidth are accurate. The proof for this is included in the Appendix A.1.

In practice, the network delay introduces errors into the estimated loss rate and bandwidth, making the algorithm approximately optimal. This algorithm also assumes that changing the sending rate of one sender does not affect the loss rates of other senders. This is a reasonable assumption if senders are located strategically across the network. On the other hand, if the assumption is not true, then sender's estimated loss rates may fluctuate after each rate assignment, leading to oscillating sending rates. Also, the estimated bandwidth $B(i,t)$ using Equation (2.1) can vary rapidly, even though the average bandwidth of other traffic remains constant. This is due to bandwidth overshooting and undershooting of TCP connections, which result in large fluctuations of the estimated bandwidth of TFRC connections. If $B(i,t)$ changes rapidly, $S(i,t)$ needs to be recomputed more frequently, and more control packets have to be sent from the receiver to all the senders, resulting in inefficiencies. Therefore, we use a hysteresis window to constrain the minimum interval during which, the sending rates must remain constant. The idea of the hysteresis window is as follows. We periodically sample the estimated bandwidth at a fixed interval ϕ. If $B(i,t)$ is greater than $S(i,t) + w$, where w is a small percentage of $S(i,t)$, then we add one to the variable *count*. At each sampling time, if the estimated bandwidth is smaller than $S(i,t) - w$, then we subtract one from the variable *count*. Otherwise, the variable *count* remains the same. The parameter *count* can be interpreted as the number of times that the sampled bandwidth is outside the window. If at some point the variable count is greater than γ, a fixed threshold, or smaller than -γ, then the rate allocation algorithm is run to update the sending rates. The parameters

w, *count*, ϕ, γ, and *count* determine the minimum interval during which $S(i,t)$ must remain constant, and the responsiveness of $S(i,t)$ to network conditions. If a protocol responds too slowly to decreasing network available bandwidth, it occupies more than its fair share of the bandwidth, and vice versa. The simulations in Section 2.5 show that on average, our algorithm based on hysteresis window shares bandwidth fairly with other TCP traffic. In Appendix A.2, we also discuss the responsiveness of our protocol for different values of count threshold γ.

2.4 Packet Partition Algorithm (PPA)

2.4.1 Basic Description of PPA

In previous section, we describe the optimal rate allocation scheme for specifying rates for each sender. In this section, we address the issue of packet selection for each sender. As described in Section 2.2.1, after receiving the control packet from the receiver, each sender immediately decides the next packet in the video stream to be sent, using the Packet Partition Algorithm (PPA). All the senders simultaneously run this algorithm in a distributed fashion in order to ensure that, all packets are sent by one and only one sender, and also to minimize the startup delay.

To show the advantages of our proposed PPA over other PPAs, we first briefly describe a PPA in P2P file sharing systems such as *Kazaa*. *Kazaa* file sharing system allows a single member to download a media file simultaneously from multiple participating members. To decide which packets to be sent by which sender, each sender is assigned to send a contiguous block of data of length proportional to its sending rate. For example, suppose there are two senders, the allowable sending rate for the first and second senders are 100 and 80 packets per second, respectively, and total playback rate is 180 packets per second. In this case, the first sender is assigned to send the first 100 packets and the second sender send the next 80 packets. Once, a sender finishes sending all its assigned packets, the receiver sends a control packet to instruct the sender to send the next block of data starting at position x and ending at position y of a media file. Typically, control packets in these systems are not sent often and the length of the block of data is long, e.g on the order of minutes for *Kazaa* to send a data block. Because of this, the receiver has to wait until the entire block of data is received before attempting to playback since its playback rate is larger than the sending rate of the first sender. Clearly, this strategy avoids duplicating packets between senders, however, it incurs unnecessary startup delay.

The main objective of our PPA is to ensure that the received packets arrive in an interleaved fashion from multiple senders, so as to reduce the startup delay. The algorithm can be described as follows. Each sender receives a control packet from the receiver through a reliable protocol at the beginning of a session or whenever the receiver determines there should be a change in any of the sending rates. The control packet contains two-byte fields $S(1)$-$S(5)$ to specify the sending rate in packets per second for each sender, and one-byte fields $D(1)$-$D(5)$ for the estimated delay from each sender to the receiver in multiples of 2 milliseconds. For most experiments, we use only two senders, however, we provide rooms for five senders as a possible extension. Also, the quantized value of 2 milliseconds is chosen to specify up to 512 milliseconds using only one byte, and at the same time being accurate enough for most practical purposes. The control packet also contains *Sync* number which is used as the starting sequence number that all senders use in the PPA to determine the next packet to send, immediately upon receiving the control packet. Note that here, we only list the essential information in the control packet to describe our packet partition algorithm in a broad sense. Clearly, information such as packet size, the amount of FEC, should also be included for system flexibility. We do not envision that control packets will incur too much overhead since they are rarely sent. The entire copy of the video is assumed to reside at all the senders. To describe the PPA in details, we use the notation in Table 2.1.

If the reference time, $T_{k'} = 0$, is conceptually chosen to be the departure time of the control packet from the receiver, the estimated arrival time of the k^{th} packet sent by sender j is $n_{j,k,k'}\sigma(j) + 2D(j)$. This is because it takes $D(j)$ for the control packet to arrive at the sender j, $n_{j,k,k'}\sigma(j)$ for the k^{th} packet to be sent by sender j, and $D(j)$ for it to arrive at the receiver. Since $P_{k'}(k)$ is the playback time of the k^{th} packet with respect to $T_{k'}$, the expression $A_{k'}(j,k) = P_{k'}(k) - [n_{j,k,k'}\sigma(j) + 2D(j)]$ can be interpreted as the estimated time difference between arrival and playback time of the

k'	Sequence number *Sync* in the control packet which all senders use to initialize the PPA
$T_{k'}$	Time at which control packet with sequence number sync k' is sent by the receiver
N	Number of senders
$P_{k'}(k)$	Playback time for k^{th} packet with respect to $T_{k'}$
$S(j)$	Sending rate for sender j
$\sigma(j)$	Sending interval between packets for sender j
$n_{j,k,k'}$	Number of packets already sent by sender j since packet k', and up to packet k
$D(j)$	Estimated delay from the sender j to the receiver

Table 2.1: *Notations for packet partition algorithm.*

k^{th} packet, if sender j is its originator.

The basic idea in our packet partition algorithm is that among all senders $j = 1, ...N$, the one that maximizes $A_{k'}(j,k)$ is assigned to send the k^{th} packet. Specifically, each sender computes $A_{k'}(j,k)$ for each packet k, for itself, and all other senders, and only sends the k^{th} packet if it discovers that $A_{k'}(j,k)$ is at a maximum for itself. If $A_{k'}(i,k)$ is not at a maximum for sender i, it will increase k by one, and repeats the procedure until it finds the packet for which $A_{k'}(j,k)$ is at a maximum among all other senders. Note that if $A_{k'}(j,k)$ is positive, the k^{th} packet is on time, otherwise, the k^{th} packet is late. In a way, by choosing sender j to send the k^{th} packet with maximum $A_{k'}(j,k)$ also results in lower probability of kth packet being late.

Each sender effectively keeps track of all the values of $A_{k'}(j,k)$ for all N senders and updates every time a packet is sent. The values evolve in the same way at all senders even though they are computed at different locations. The reasons for this are that all the senders (a) receive the same control packet from the receiver, (b) only use the information in the control packet to update and (c) use the same equation to do so. Therefore there is no need for information exchange among the senders in order to determine who needs to send the next packet. There is also no need to synchronize all the senders'clocks to a global time. To illustrate the algorithm, we show a simple example in which, there are two senders, each sending packets at same rate. As shown in Figure 2.5, the *Sync* sequence number is 10. The top line with vertical bars denotes the playback time for the packets. The middle and the bottom lines indicate the time to send packets for senders 1 and 2, respectively. In this scenario, packet 10 will be sent by sender 1 since the estimated difference between playback and its receive time for packet 10 is greater than that of sender 2. Next, packet 11 will be sent by sender 2 for the same reason. In our implementation of a reliable protocol for the control packets, a batch of 5 identical control packets are sent with 5 milliseconds spacing between them whenever the receiver determines there should be a change in sending rates. If none of the control packets is acknowledged within two round-trip time from a particular sender, a new batch of control packets are sent to all the senders until the control packets are acknowledged by all senders. The interval of 5 milliseconds between the control packets is chosen in order to reduce potential loss of all 5 control packets due to burst loss, while ensuring that the control packets do not arrive at one particular sender too late. This value does not affect the overall performance of the system since control packets are rarely sent, e.g., on the order of several seconds to

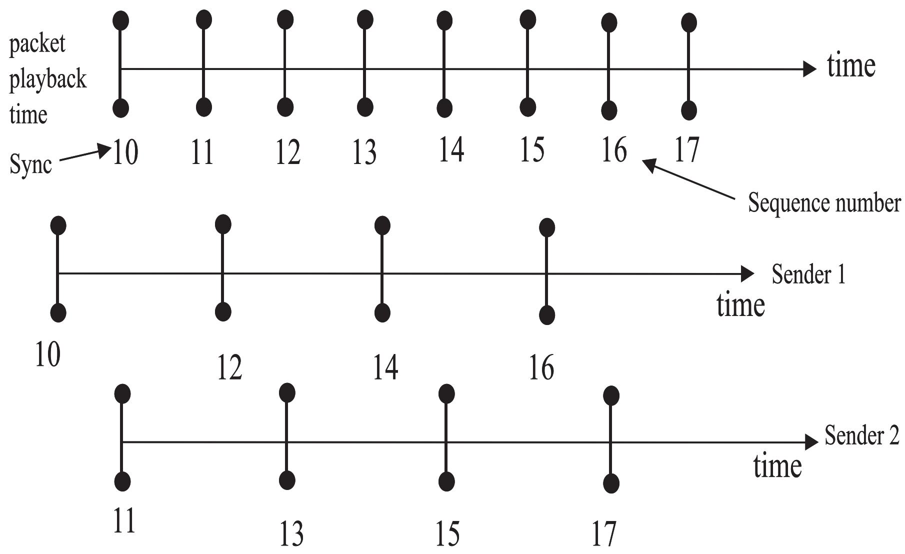


Figure 2.5: *Illustration of packet partition algorithm.*

minutes, and the overall loss probability of all control packets is very small.

2.4.2 Practical Considerations with Packet Partition Algorithm

A practical consideration is the choice of the synchronization sequence number, k', in the control packet to signal the new sending rates or change in delay. At any point in time, the receiver knows the sequence number for the last packet sent by each sender. Thus, a reasonable strategy is to choose k' more or less to be around the sequence numbers that the senders are actually sending when they receive the control packets.

As an example, consider scenario in Figure 2.6. The already sent packets by either sender are denoted by crosses, while the packets to be sent by circles. Suppose the receiver sends out control packets which arrive at sender one before sender two. Thus, by the time the control packet arrives, sender one's most recently sent packet is 6, and that of sender two is 11. If k' is chosen to be 11, then sender one will either have to skip packets 8 and 10, or send them temporarily faster before adapting to the new rate. On the other hand, if k' is chosen to be 8, then senders one and two, will run the PPA beginning from packet 8 in order to determine who sends what and the process proceeds smoothly. If k' is significantly smaller than the packet sequence number a particular sender is processing at the time it receives the control packet, it is possible that some packets are sent twice during this short interval, i.e. are duplicated at the receiver.

Based on the above discussion, one strategy is for the receiver to set $k^{'} = \min_j k''(j)$, where $k''(j)$ is the estimated sequence number for the latest packet sender j has just sent, before receiving the control packet. Since the receiver knows about the last packet received from each sender, and the round trip time between itself and each sender, it can arrive at a fairly accurate estimate for $k''(j)$. In particular, a reasonable estimate for $k''(j)$ is $k^*(j) + 2D(j)S$, where $k^*(j)$ is the sequence number for the last packet receiver has received from sender j, and S is the total sending rate in packets per second. This estimate of $k''(j)$ is reasonable since $2D(j)S$ is the approximate number of packets sent by all senders during round trip time of sender

j, i.e. during $2D(j)$ interval. Another strategy is to choose $k' = \max_j k''(j)$, and other "behind" senders have to send all the packets with sequence numbers between their current sending sequence numbers and the "sync" number temporarily faster before adapting to the new rates. The "min" strategy of setting $k' = \min_j k''(j)$ results in higher number of duplicate packets as compared to the strategy of setting $k' = \max_j k''(j)$. The "max" strategy increases the total sending rate temporarily. This strategy also compensates for the throughput reduction during the transition to be described shortly. It is also possible that always choosing $k' = \max_j k''(j)$ may also increase the buffer size at the receiver in the long run, in that case, the receiver can choose $k' = min_j k''(j)$ to reduce the buffer.

We now discuss the throughput reduction problem when using the strategy $\min_j k''(j)$. When there is the time interval between the changing rate of the senders, the received aggregate bit rate temporarily deviates from the encoded bit rate, S, during this interval. The analysis of throughput reduction is provided in Appendix A.3. As an example, consider Figure 2.6, where two senders are assumed to send packets at same rate $S/2$, with sender one having a shorter round trip time (RTT) than sender two. The time difference $t_1 - t_0$ corresponds to half the RTT difference between sender one and two. As a result, the received bit rate during the first $RTT1 - RTT2$ is only $S/2$. Hence the receiver cannot start playing back the video at rate S after $RTT1 - RTT2$. To remedy the situation, the receiver can use buffer to absorb this difference before playing back the video at rate S.

However, if control packets are sent often and the “min” strategy is used, this can result in eventual depletion of the receiving buffer due to duplication of packets. One way to fix this problem is for the receiver to set $k^{'} = \max_j k''(j)$ as to temporarily increase or decrease the total sending rate depending the fullness of its buffer. Each sender then sends all the packets with sequence number between its current sending packet and the “sync” number within a prespecified time.

Since control packets sent to all senders are identical, each sender is aware the RTT of all the other ones and hence the sender with the shorter RTT can temporarily increase their bit rates for a short amount of time so as to compensate for the lower aggregate throughput within a prespecified amount of time after the rate change. Yet another alternative to deal with this problem is for the receiver to slow down playback S[41].

We have implemented both the “min” and “max” strategies in our testbed. The difference in packet loss and delay between the two strategies for the typical Internet streaming scenarios are negligible. In terms of the length of the required startup buffer at the receiver, the “min” strategy without temporarily increasing or decreasing the sending rates, requires slightly larger buffer than the “max” strategy or the “min” strategy with adjusting k' to absorb the throughput reduction. If the amount of memory at the receiver is limited, such as the set top box, the “max” strategy or the “min” strategy with adjusting k' is preferred. However, if the delay and the amount of memory at the receiver is not crucial, the “min” strategy without temporarily increasing or decreasing the sending rates is preferred due to its simplicity, i.e., senders do not have to adjust their sending rates to compensate for the throughput reduction.

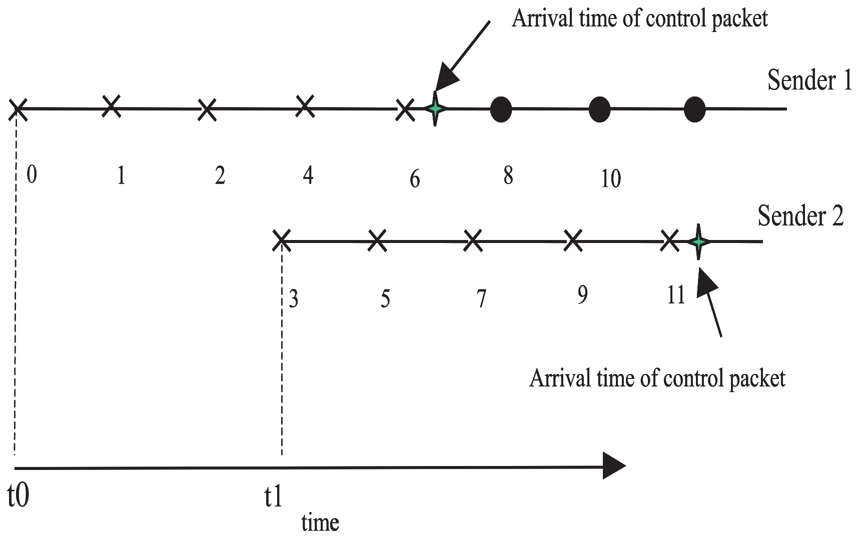


Figure 2.6: *Illustration of choosing synchronization sequence number.*

2.5 Simulations and Experimental Results

We now demonstrate the effectiveness of our approach by showing that simplistic distributed video streaming without using our protocol results in lower video quality than would be achieved otherwise.

2.5.1 NS Simulations

The first scenario involves a single receiver simultaneously receiving video from two senders using our protocol. Via the same network configuration, in the second scenario, receiver also receives video from two senders but without using our protocol to adapt to the channel conditions. The network configuration is shown in Figure 2.7. The delays from senders one and two to the receiver are 40 and 50 milliseconds, respectively, the link capacities from senders one and two to the router R are 60Mbps and 30Mbps, respectively, and the local link from router R to the receiver is 240Mbps. Senders one and two simultaneously stream a video to the receiver. The video is a MPEG-1 video sequence taken from MPEG-7 test suite which is then decoded and recoded using H.263 encoder with error-resilient option at 880kbps. At the receiver, we use a simple error-resilient technique to combat packet losses. Basically, the error-resilient technique replaces the lost group of block (GOB) of the current frame with GOB of the previous frame, and copies the motion vectors of the lost GOB from the GOB above it. The parameters for our experiment are shown in Table 2.2.

To provide a compromise between responsiveness to network conditions and oscillations of sending rates, we find that the parameters shown in Table 2.2 provide a reasonable trade-off for many experiments using the same network topology but with different link rates and number of TCP sources. Using these parameters, it takes about 20 to 30 seconds for our protocol to respond to network conditions.

At time $t = 0$ seconds, 100 TCP sources that share the same route with sender

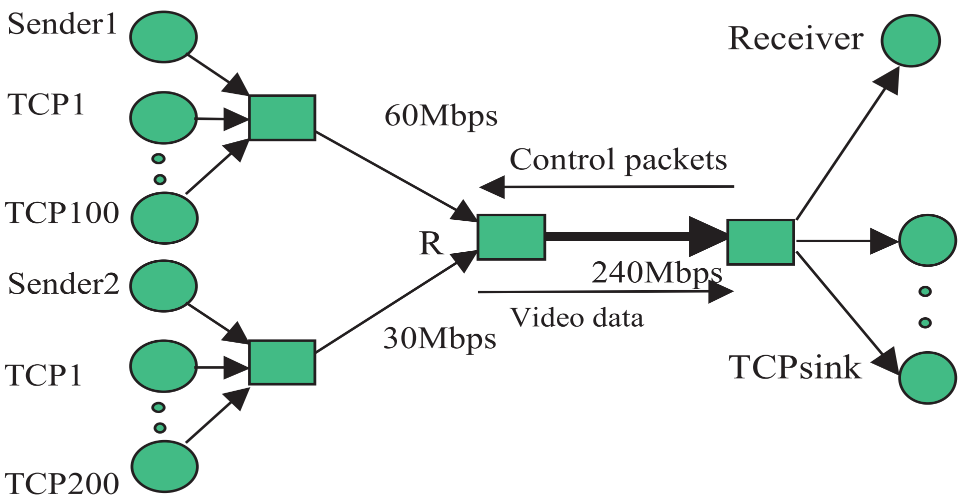


Figure 2.7: *Simulation configuration.*

w(width of the hysteresis window)	$0.1S(i)$
ϕ(sampling interval for the hysteresis window)	100 milliseconds
γ(count threshold)	40
$PWIN$(time window for estimating loss)	128RTT
P(packet size)	500 bytes

Table 2.2: *NS Simulation parameters .*

one, and 200 TCP sources that share the same route with sender two, start transmitting data. At $t = 2$ seconds, the receiver starts sending control packets to coordinate the video transmission from senders one and two. Initially, the receiver does not know the fair bandwidth for each sender, so 880kbps is divided equally between the two senders. Figure 2.8 shows the estimated loss rate for each sender. At $t = 25$ seconds, both senders begin to send data at the rates according to the Equation (2.1) based on their loss rates and round-trip times as shown in Figure 2.9. Between $t = 25$ and 200 seconds, the average bandwidth of sender one is 76.7 KBytes/s, which is close to the fair bandwidth of 77.8 KBytes/s per connection obtained by dividing 60Mbps among 101 connections. At $t = 200$ seconds, 25 of 100 TCP sources that share the same route with sender one stop sending data; our protocol responds by increasing sender one's rate by approximately a quarter of its current rate, and reducing the sender two's rate such that the total bandwidth is 880kbps. At $t = 400$ seconds, 10 new TCP sources that share the same route with the sender one start and stop at random times, varying the available bandwidth for sender one. As shown in Figure 2.9, our algorithm adjusts the sending rates for both senders appropriately. Figure 2.10 shows the total throughput and the throughput of each sender. As seen, the variations in Figure 2.10 reflect closely those in Figure 2.9. In scenario one, both senders use our adaptive protocol to adjust their sending rates accordingly so as to be TCP friendly, and at the same time, to reduce the overall loss rate. In scenario two, the sender two is not TCP-friendly since it sends more than the average bandwidth of other TCP sources while sender one under-utilizes its available bandwidth. Figure 2.11 shows the throughput for each sender in scenario two, and Figure 2.12 shows the loss rate ratio of scenario two over that of scenario one. As seen, the loss rate of scenario two can be 2.5 times larger than that of scenario two. Next, we compare the mean squared error (MSE) of pixel values between the sent frame and the received frame. Since there are quite a large number of frames, we further average the MSE for every video frame over a period of 5 seconds. Higher MSE represents lower fidelity of the video due to lost packets. As shown in Figure 2.13, most of the time the MSE for scenario one is lower than that of scenario two. The average MSE for the entire simulation for scenarios one and two are 13.70dB and 16.16dB, respectively. The difference in MSE between the two scenarios is most significant from $t = 200$ to $t = 400$ seconds. During this period, there are fewer active TCP sources, and therefore, sender one has more available bandwidth, triggering our protocol to allocate more packets to sender one. Since sender one has lower loss rate, the overall loss rate decreases, resulting in smaller MSE. Beyond MSE arguments, it is worthwhile to mention that while our protocol allows the distributed senders to compete for bandwidth fairly with existing TCP connections, in scenario two, TCP connections sharing bandwidth with distributed senders suffer unfairly.

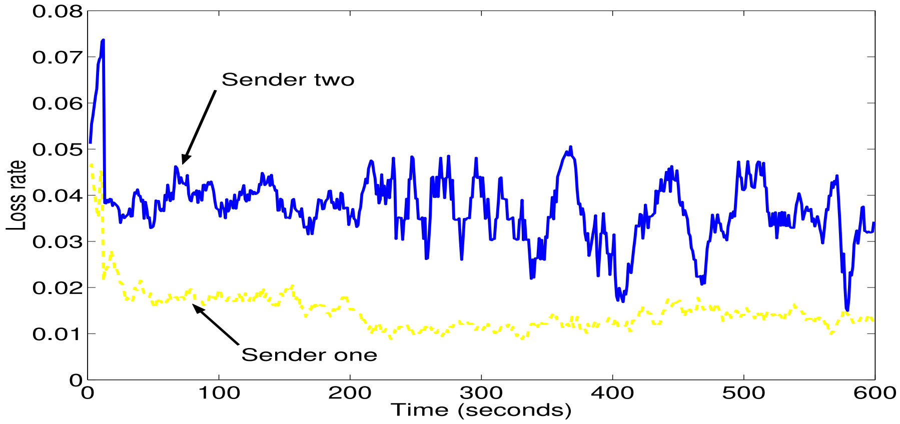


Figure 2.8: *Loss rate of two senders in adaptive scenario one.*

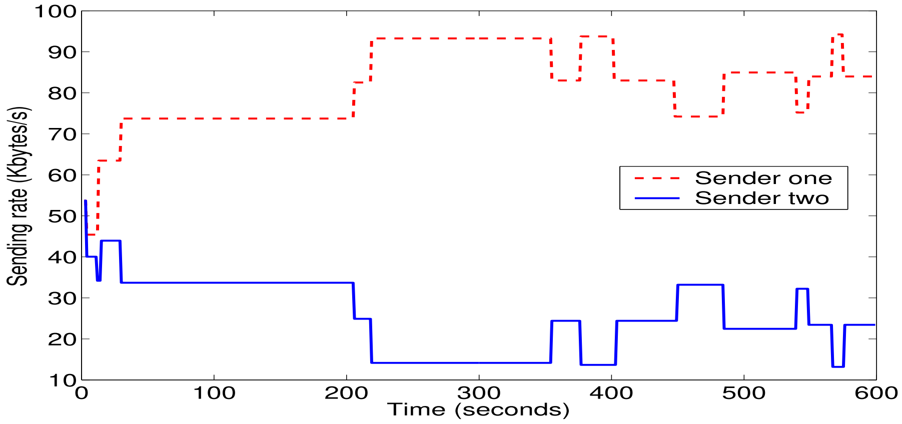


Figure 2.9: *Send rate of two senders in adaptive scenario one.*

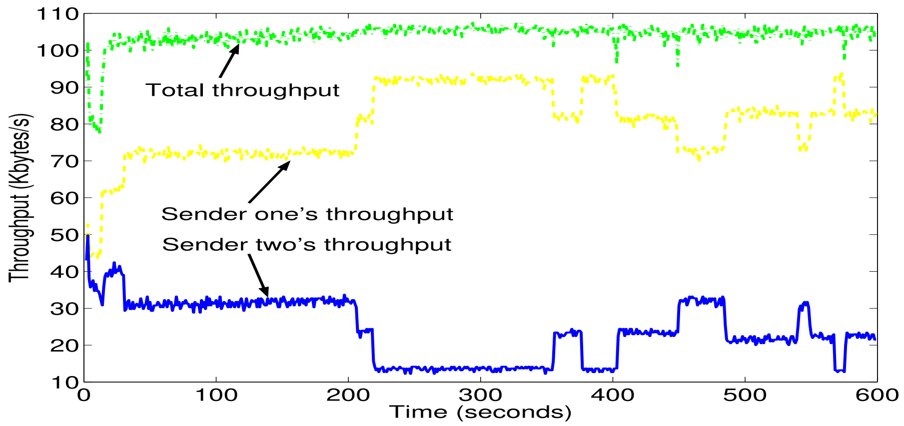


Figure 2.10: *Throughputs of two senders in adaptive scenario one.*

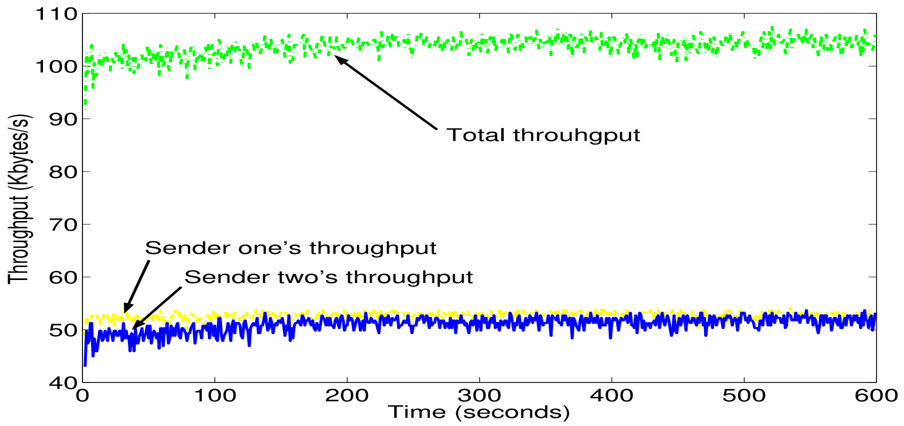


Figure 2.11: *Throughputs of two senders in non-adaptive scenario two.*

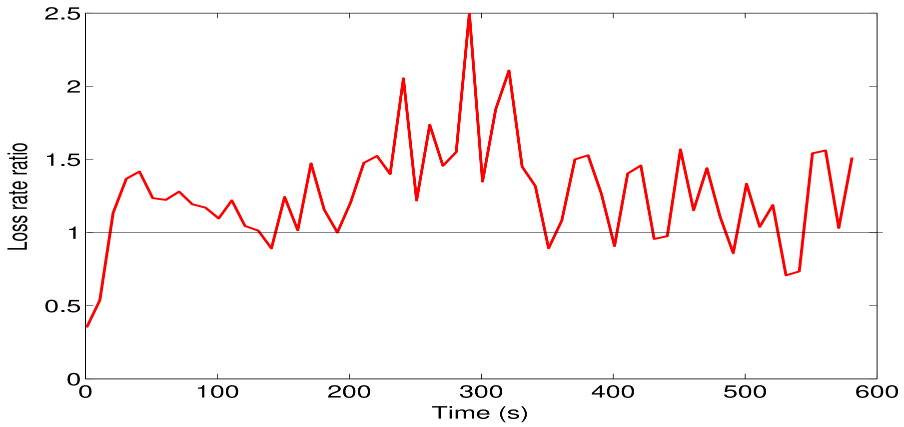


Figure 2.12: *Loss rate of scenario two over that of scenario one.*

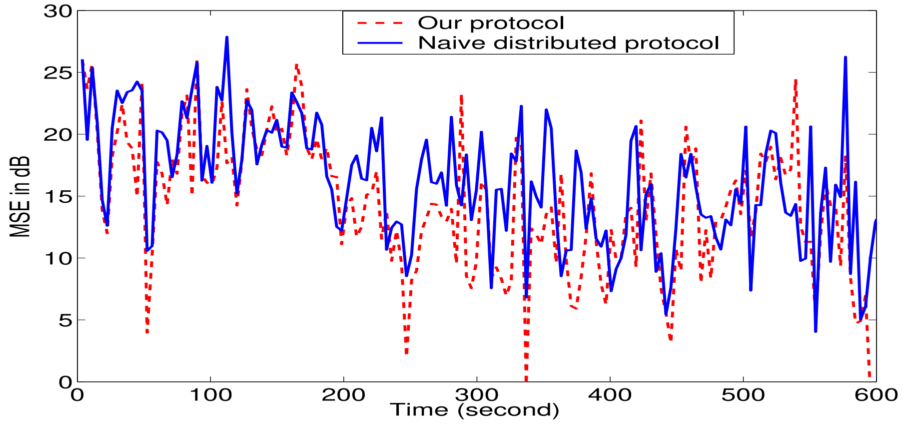


Figure 2.13: *MSE for adaptive and non-adaptive scenarios.*

2.5.2 Internet Experiments

We have also implemented our protocol for actual experiments over the Internet. In our Internet experiment, packets of 500 bytes are sent simultaneously at an aggregate rate of 880kbps or 220 packets per seconds from Sweden and Belgium to U.C. Berkeley. Since the loss rates on these routes are extremely small, packets are artificially dropped at the senders to simulate the congestion effect. Figure 2.14 shows the initial simulated loss rates of 6% and 3% for senders at Belgium and Sweden, respectively. Initially, the receiver does not know loss rates of two senders, so the total sending rate is divided equally between two senders at 440kbps each. After 10 seconds of monitoring network characteristics, the receiver sends the control packets to both senders to adjust the sending rate appropriately. Upon receiving the control packets, senders at Sweden and Belgium start sending packets at roughly 580kbps and 300kbps, respectively, to result in throughputs shown in Figure 2.15. As expected, Sweden sender has lower loss rate, hence, is assigned to send packets at higher rate. However, Sweden sender cannot send packets at higher than 580kbps due to the TCP-friendly bandwidth constraint. Therefore, Belgium sender sends the remaining 300kbps to accommodate the 880kbps video. From $t = 100$ to 200 seconds, the loss rate of Belgium sender drops from 6% to 0.3%. Approximately 20 seconds later, our rate allocation algorithm reassigns all 880kbps to Belgium sender since it has lower loss rate, and its fair estimated bandwidth is more than 880kbps. From $t = 200$ to 300 seconds, the overall loss rate is 0.3% rather than $.3\%(300/880) + 3\%(580/880) = 2\%$

which would have been the loss rate had the sending rates not been adapted to channel conditions properly. At $t = 200$ seconds, Belgium sender's loss rate increases back to 6%, triggering our rate allocation algorithm to reassign the sending rates for the two senders properly as seen in Figure 2.15.

Next, we show the results of our packet partition algorithm for the same path diversity streaming experiment. As discussed previously, the goal of the packet partition algorithm is for senders to send packets in an interleaved fashion such that (a) packets are not duplicated and (b) arrive at the receiver more or less in playback order. During this experiment, there are a total of five control packets sent. We use the approach of setting the synchronization number k' in the control packet to $\min_j^{k''(j)}$ as discussed previously in Section 2.4.2. Using this approach, only 0 to 3 duplicates packets are observed every time control packets are sent. Figure 2.16 shows the histogram of the sequence number difference of the consecutive received packets. As seen, most packets have sequence numbers within 5 of each other. This indicates the effectiveness of the proposed PPA to send closely interleaved packets. Figure 2.17 also shows the histogram of the time interval between packets with consecutive sequence numbers. Ideally, the histogram should be a single column at $1/220 = 4.5$ milliseconds which represents the sending interval between consecutive packets. However, network jitter, packet loss, and delay difference of the two paths result in a spread-out histogram. However, majority of packets with consecutive sequence numbers are still received within 10 milliseconds of each other. Again, this indicates the effectiveness of the proposed PPA to send closely interleaved packets as to reduce the startup delay.

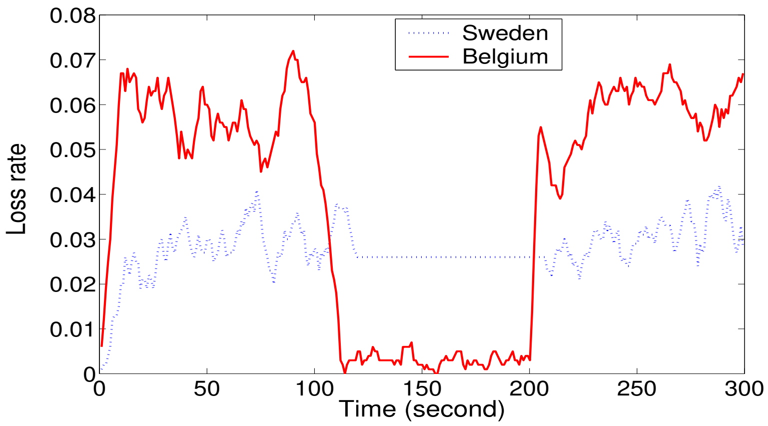


Figure 2.14: *Loss rate versus time of simultaneous streaming from Sweden and Indiana to Berkeley.*

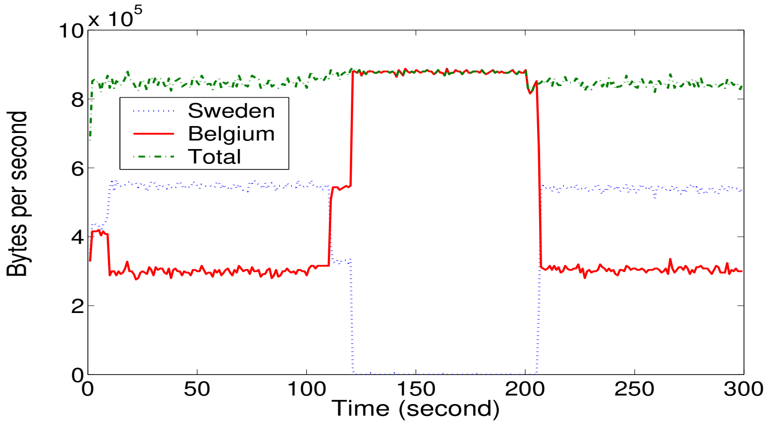


Figure 2.15: *Throughput versus time of simultaneous streaming from Sweden and Indiana to Berkeley.*

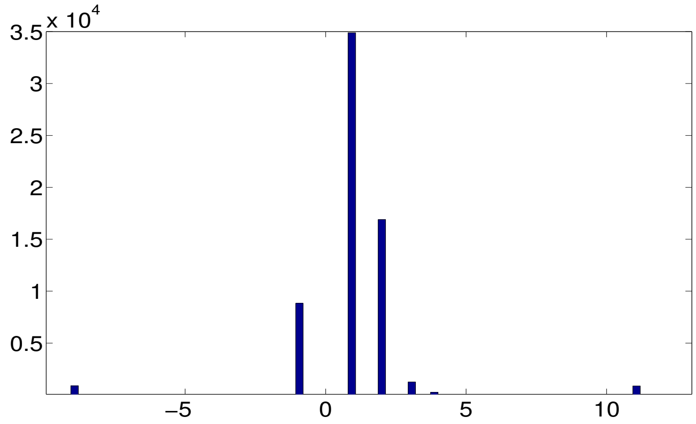


Figure 2.16: *Histogram of the difference in sequence number of consecutive received packets.*

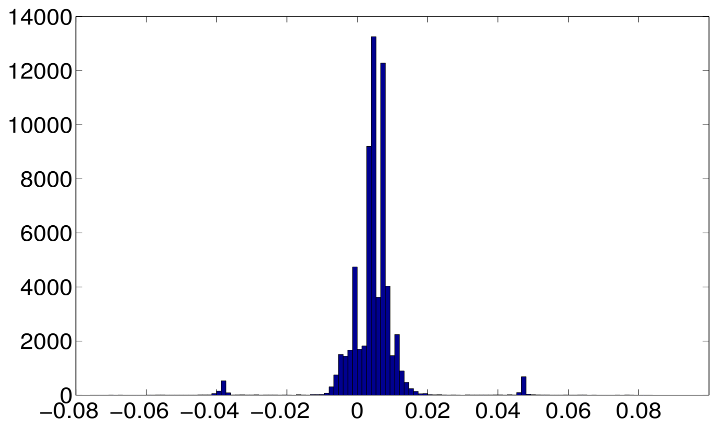


Figure 2.17: *Histogram of the delay difference between consecutive packets.*

2.6 Conclusions

In this chapter, we have proposed a *path diversity* framework for simultaneously streaming video from multiple senders to a single receiver in order to achieve higher throughput, and to increase tolerance to loss and delay due to congestion. We exploit a TCP-friendly protocol and propose rate allocation and packet partition algorithms to minimize the packet loss, and startup delay. We have shown that path diversity video streaming using our protocol results in lower distortion than path diversity video streaming in a simplistic, non-adaptive fashion.

Chapter 3

Rate Allocation with Forward Error Correction

3.1 Introduction

Our proposed *path diversity* media streaming protocol introduced in Chapter 2, consists of a rate allocation algorithm that minimizes the packet loss in random loss environments, without using Forward Error Correction (FEC). On the other hand, on many parts of the Internet, packet loss is rather bursty, and for many delay sensitive media streaming applications, FEC are often employed to combat packet loss and retransmission delay. Based on these, in this chapter, we propose a new rate allocation algorithm to be used with FEC in bursty loss environments in order to minimize the irrecoverable loss probability.

In general, FEC has been shown to be an effective tool in combating packet loss in low delay multimedia communications[17]. It can be argued however, that for streaming applications, retransmissions together with large buffer sizes would obviate the need for FEC. In practice though, retransmissions result in large start up delay, and limit interactive VCR-like functionalities such as fast forward and rewind. Based on delay considerations, in this chapter, we propose a new rate allocation for our *path diversity* media streaming protocol to be used with FEC.

A well known drawback of FEC though is that it results in bandwidth expansion, and hence reduces the amount of available bandwidth for the actual media bit stream. Since the level and burstiness of packet loss in the Internet fluctuates significantly, incorporating the optimal amount of FEC in any single route streaming application is a difficult task; too little redundancy cannot effectively protect the media bit stream, and too much redundancy consumes too much bandwidth unnecessarily. Thus FEC level has to be closely matched to channel characteristics for it to be effective in single route streaming applications. In this chapter, we show that FEC combined with path diversification via multiple senders, can combat bursty loss behavior in the Internet more effectively than in single sender case. Specifically the above sensitivity mismatch between FEC level and network characteristics for single route streaming applications is significantly reduced in *path diversity* media streaming applications due to the additional redundancy introduced by multiple routes.

This chapter is organized as follows. In Section 3.2, we list specific assumptions for our rate allocation algorithm. In Section 3.3, we provide a brief overview of FEC, particularly, Reed-Solomon code. We then discuss the Internet packet loss model in Section 3.4. In Section 3.5, we propose a rate allocation algorithm that minimizes the irrecoverable loss probability given the network characteristics and a fixed amount of FEC. Next, we present the simulations and actual Internet experimental results using the proposed rate allocation algorithm in Section 3.6. Finally, we conclude in Section 3.7.

3.2 Assumptions

We now list specific assumptions for our rate allocation algorithm.

1. The total needed video rate, S, is available from the aggregate bandwidth of all the senders. We make this assumption for sake of simplicity, if a scalable video stream is used to adjust the source bit rate, this assumption can be relaxed.

2. The amount of FEC is fixed. We make this assumption to simplify the analysis of optimal sending rate.

3. Video is pre-coded at a fixed rate and residing at all servers.

4. Packet loss rate is independent of the sending rate. This is a direct consequence of our network model, i.e. two-state continuous Markov chain to be described shortly. This assumption holds if the video bit rate is only a small part of the

total traffic, i.e. there is a large degree of statistical multiplexing. Since speed of routers on the Internet are on the order of hundreds of megabits to tens of gigabits per second, and typical streamed video is less than 1Mbs, we believe this assumption is justified.

5. Packet loss between two routes are independent. We make this assumption to simplify the analysis of rate allocation. In general, the distributed streaming framework still provides benefits over traditional uni-sender scheme if packet loss correlation between routes is small.

6. Packet size is fixed, hence FEC can be applied.

3.3 Erasure Codes

Erasure codes are a form of FEC [42] used for communication between senders and receivers through a lossy medium. When decoding the encoded data using erasure codes, the receiver is assumed to know the exact location of the lost packets, while this information is not needed in a general FEC technique. Erasure codes are typically used for sending packets through the Internet since the receiver can detect the location of the lost packets by noting the skipped packet sequence number. In a typical erasure code, sender encodes redundant packets before sending both the original and redundant packets to the receiver. Receiver can reconstruct the original packets upon receiving a fraction of the total packets. Standard erasure codes such as the (N, K)

Reed-Solomon erasure codes, take K original packets and produces $(N-K)$ redundant packets, resulting in a total of N packets. If K or more packets are received, then all the original packets can be completely reconstructed. Hence, larger $\frac{N}{K}$ ratio leads to higher level of protection for data. Throughout this chapter, we use Reed-Solomon codes in our analysis, simulations, and experiments.

3.4 Network Model

An accurate model for packet loss over the Internet is quite complex. Instead, we model our network as a simple two-state continuous-time Markov chain shown in Figure 3.1, which has been shown to approximate the behavior of packet loss over the Internet fairly accurately [17][18][43]. A two-state continuous Markov chain with state at time t denoted by X_t where $X_t \in \{g, b\}$, is characterized by μ_g and μ_b. μ_g and μ_b can be thought of as rates at which the chain changes from 'good" to "bad" state and vice versa. When the chain is in good state, the probability of having lost packets is much smaller than that of when the chain is in bad state. A further simplified model assumes that a packet transmitted at time t is successfully received if $X_t = g$, and is lost otherwise. The stationary probabilities of lost packet and received packet are π_b and π_g where $\pi_b = \frac{\mu_b}{\mu_g+\mu_b}$ and $\pi_g = \frac{\mu_g}{\mu_g+\mu_b}$.

We now provide intuition for using a two-state continuous-time Markov model

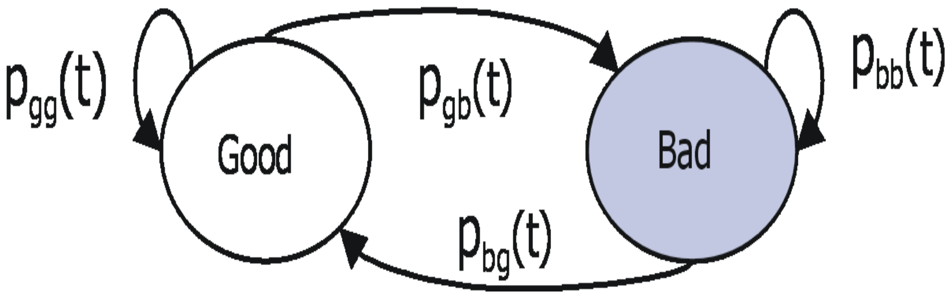
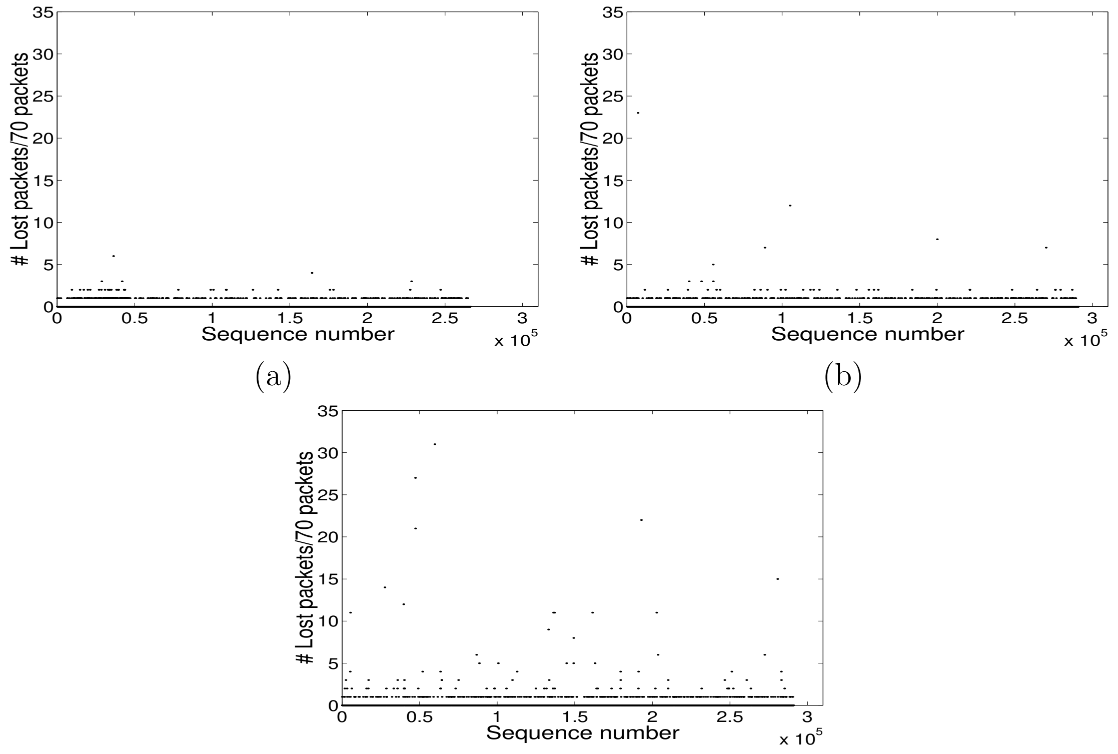


Figure 3.1: *Two-state Markov model.*

to approximate the packet loss behavior of the Internet traffic. It is known that lost packets in the Internet often happen in bursts. This loss of successive packets is due to the way packets are dropped at the network routers during congestion. Most routers employ First-In-First-Out (FIFO) policy in which, the successive arrived packets are dropped if the network buffer becomes full. Hence, a Markov chain which models a bursty loss environment, approximates the Internet traffic reasonably well. Also the average congestion period during which, the amount of aggregate traffic exceeds the link's capacity, can be thought of as $1/\mu_b$, i.e. the average time that the Markov chain is in the bad state. Clearly, sending more packets during congestion results in larger number of lost packets.

To support the validity of the Markov model, we have performed a number of streaming experiments over the Internet, the results of which are shown in Figure 3.2. The experiments show results of sending packets from Hong Kong University to U.C. Berkeley at the rates of 200, 400, and 800$kbps$, respectively. The vertical axis shows the number of lost packets out of every 70 packets, and the horizontal axis shows the packet sequence number. As expected, larger sending rates result in longer loss bursts when the route is in congestion. The largest numbers of lost packets out of 70 packets for 200, 400, and 800$kbps$ are 6, 23, and 31 packets, and the average loss rates are 0.13%, 0.18%, and 0.21%, respectively. Figure 3.2(a) indicates that if $RS(70, 64)$ codes are used to protect the data, then all packets sent at the rate of 200$kbps$ will be completely recovered, while there will be 5 and 14 instances of irrecoverable loss for sending packets at the rates of 400$kbps$ and 800$kbps$, respectively. Irrecoverable loss occurs when the number of lost packets out of N encoded packets exceeds $N - K$ or 6 in these experiments. These experiments also indicate that even though the average loss rate is small, a substantial amount of redundancy in erasure codes is needed to recover the data completely. For instance, based on Figure 3.2(c), a strong $RS(70, 39)$ code is needed to completely recover all the data. In general, FEC codes do not combat bursty loss efficiently. Given the same ratio N/K and the bursty loss characteristics, the efficiency of $RS(N, K)$ code increases as N and K increase. However, the encoding and decoding computational complexities and delay also increase. To alleviate this drawback, interleaving techniques [44] are often used, even though they result in increased delay.

The above experiments provide an insight on how to combat bursty loss efficiently without introducing increased delay and complexity. Instead of streaming a video at 800$kbps$ from the sender at Hong Kong University to a client at U.C. Berkeley, we can use 4 senders, each streaming the video simultaneously at 200$kbps$ to the client



(a) (b)

(c)

Figure 3.2: *Hong Kong to U.C. Berkeley, (a) rate = 200kbps; (b) rate = 400kbps; (c) rate = 800kbps*

at U.C. Berkeley. The senders are assumed to be located in such a way that the packet loss along the routes between each sender and the client are uncorrelated. Assume that the routes between all the senders and the client have the same loss behavior as the route between Hong Kong University and U.C. Berkeley; then we expect that the number of instances of irrecoverable loss to be fewer than streaming a video at 800*kbps* from only one sender. The effect of sending packets at a lower rate on multiple independent routes transforms the bursty loss behavior into a more uniform loss behavior, thus increasing the efficiency of FEC techniques. Naturally, given a number of routes each with a different loss behavior, the video bit rate, and the total amount of FEC protection, there should be an optimal partition of sending rates for each route. We address this rate allocation problem in the next section.

3.5 Optimal Rate Allocation

To simplify analysis, we replace the two-state continuous-time Markov chain with an equivalent two-state discrete one. To see the equivalence, note that since we are only interested at the instances when a packet is sent, equations for computing the transition probabilities for the discrete one can be derived as:

$$p_{gg} \triangleq P(X_{n+1} = g | X_n = g) = \pi_g + \pi_b e^{-(\mu_g + \mu_b)\tau} \tag{3.1}$$

$$p_{gb} \triangleq P(X_{n+1} = g | X_n = b) = 1 - p_{bb}(\tau) \tag{3.2}$$

$$p_{bb} \triangleq P(X_{n+1} = b | X_n = b) = \pi_b + \pi_g e^{-(\mu_g + \mu_b)\tau} \tag{3.3}$$

$$p_{bg} \triangleq P(X_{n+1} = b | X_n = g) = 1 - p_{gg}(\tau) \tag{3.4}$$

where τ is the sending interval. With the discrete model, the discrete time step corresponds to the event of sending a packet. The process of the discrete Markov chain undergoing n discrete time steps is equivalent to the process of sending n packets through the network. To further simplify analysis, we only consider the case of two senders, both assumed to be sending packets to the receiver along two routes with independent packet loss. The extension of analysis to the case with more than two senders is straightforward. Our goal is to find the sending rates for the two senders in

N	Total number of packets in a FEC block
K	Number of data packets in a FEC block
B_m	Estimated available bandwidth for sender m in packets per second
(μ_g^m, μ_b^m)	Network parameters for route between sender m and the receiver
$\lambda = \frac{N}{S_{req}}$	Interval between successive transmitted FEC blocks in seconds
N_m	Number of packets transmitted by sender m during λ seconds

Table 3.1: *Notations for rate allocation algorithm.*

order to (a) minimize the probability of irrecoverable loss for a given protection level of FEC, and (b) to ensure that each sender sends packets only at available bandwidth. To formally state our rate allocation problem, we use the notation in Table 3.1. The rate allocation problem can now be stated as follows:

Given S_{req}, N, K, B_m, (μ_g^m, μ_b^m), we want to find N_m for $m \in \{A, B\}$ so as to minimize the probability of irrecoverable loss given by

$$C(K, N_0, N_1) = \sum_{j=N-K+1}^{N_A+N_B} \sum_{i=0}^{j} P(A, i, N_A) P(B, j-i, N_B) \tag{3.5}$$

subject to

$$N_A + N_B = N, \quad \tfrac{N_A}{\lambda} \leq B_A, \quad \frac{N_B}{\lambda} \leq B_B \tag{3.6}$$

where $P(m, i, N_m)$ denotes the probability that i packets are lost out of the N_m packets sent by sender m, and $C(K, N_A, N_B)$ denotes the probability that more than $N-K$ packets are lost out of a total $N_A + N_B$ packets sent by both senders. Since we assume independent packet loss along the two routes, the probability of j lost packets out of $N_A + N_B$ packets sent by both senders can be written as $\sum_{i=0}^{j} P(A, i, N_A) P(B, j-$

$i, N_B)$. Therefore, the probability of more than $N-K$ lost packets out of N_A+N_B packets sent by both senders, or equivalently the irrecoverable loss probability, is $\sum_{j=N-K+1}^{N_A+N_B}\sum_{i=0}^{j} P(A,i,N_A)P(B,j-i,N_B)$. As indicated in inequality constraints (3.6), $\frac{N_m}{\lambda}$ is the sending rate of sender m, which is required to be less than or equal to the available bandwidth. Since the sum of the sending rates equals to the required sending rate for the video, we have $N_A+N_B=N$. We also assume that sum of the available bandwidth of all senders is always greater than or equal to the video bit rate. When the bandwidth constraints in (3.6) are not satisfied due insufficient network bandwidth, a scalable video bit stream can be used to stream the video at a lower bit rate. The procedure to compute the $P(m,i,N_m)$ is shown in Appendix B.1. Using $P(m,i,N_m)$, we search over all possible values of N_A and N_B such that the constraints (3.6) are satisfied, and $C(K,N_A,N_B)$, the probability of irrecoverable packet loss is minimized [1]. This search is fast since for the case of two senders, only N comparisons are required. For M senders, the straightforward exhaustive search has complexity $O(N^{M-1})$. This is indeed problematic if N and M are large. Another approach is to precompute optimal sending rates for typical channel conditions in a table, and adapt to these sending rates when the actual channel conditions are similar to the ones stored in the table. Also, we believe that from an implementation point of view, having more than 10 connections, might make the coordination of the senders more difficult.

[1]Another approach is perhaps to consider the joint channel as a product of two Markov chains and avoid exhaustive search.

Note that in our rate allocation formulation, the amount of FEC is assumed to be known in advance. The constraints on this amount of FEC are the available bandwidth, complexity, and delay associated with decoding and encoding a FEC block. For applications that can tolerate high delay, more FEC can be used, provided that (a) the available bandwidth is greater than aggregate bandwidth of the application and FEC overhead, and (b) sufficient computing power for encoding and decoding exists at the sender and receiver.

3.5.1 Numerical Characterization

In this section, we present numerical results for various FEC protection levels and network characteristics using the optimal rate allocation between two senders versus that of using one sender.

In the two sender scheme, we send packets on two "loss independent" routes A and B while in "one sender" scheme, packets are sent only on route A. Since end-to-end packet loss characteristics on the Internet vary widely depending on the locations of sender and receiver, we show the results for two common scenarios X and Y with the parameters shown in Table 3.2. We choose these parameters to match with those reported in [43][45]. Note that in both scenarios X and Y, route A has lower packet loss rate than route B. The aggregate sending rate of both "two senders" and "one sender" scheme is $800kbps$, and the packet size is set to 500 *bytes*. We assume that available bandwidth for each route is greater than 800kbps in order to make a fair

Scenario	Average good time in seconds		Average bad time in seconds		Packet loss probability in the good state		Packet loss probability in the bad state		Average packet loss rate p	
	A	B	A	B	A	B	A	B	A	B
X	0.01	0.01	0.01	0.01-0.05	0	0	1	1	1%	1%- 5%
Y	5	5	0.5	0.5-2	0	0	0.2	0.2	1%	2%- 6%

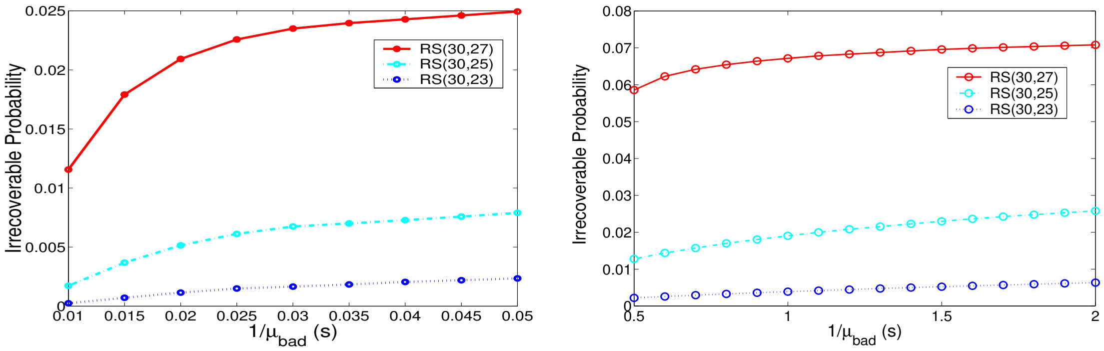
Table 3.2: *Parameters chosen for numerical characterization in two scenarios X and Y.*

comparison for single route versus multiple route streaming.

Figures 3.3(a) and (b) show the probability of irrecoverable loss for scenarios X and Y, respectively, as a function of average bad times of route B for three FEC protection levels: $RS(30,27)$, $RS(30,25)$, and $RS(30,23)$. As the average bad time of route B increases, as expected, the irrecoverable loss probability also increases for all three levels of FEC protection. Furthermore, a small increase in FEC protection level from $RS(30,25)$ to $RS(30,23)$ can reduce the optimal irrecoverable probability by a large factor such as 3. Even though the packet loss probability in the bad state is larger in scenario X, the average time in bad state is much higher in scenario Y. As seen in Figures 3.3(a) and 3.3(b), this results in the overall irrecoverable loss probability to be considerably larger in scenario Y than X.

Figures 3.4(a) and (b) show N_A, the optimal number of packets out of 30 that should be sent on route A as a function of the average bad time of route B for scenarios X and Y, respectively. As the average bad time of route B increases, more packets are sent on route A for all three levels of FEC protection. At the same average bad time of route B, the number of packets sent on the route A decreases with increased FEC protection level. This indicates that when stronger FEC protection is employed, more packets should be sent on the "bad" route. This is intuitively plausible by considering that in the reverse scenario in which weak FEC protection is employed, more packets should be sent on the "good" route. Specifically, in the extreme case where no FEC is used, no losses can be recovered by FEC, and hence, it is advantageous to primarily rely on the "good" route.

Figures 3.5(a) and (b) show the ratio of irrecoverable loss probabilities between the cases when all packets are sent on route A and when the optimal rate allocation is employed to send packets on both routes A and B for various FEC protection levels. As seen, the irrecoverable loss probability improves by as much as a factor of 15, depending on network conditions and the amount of FEC. Also as expected, the curve corresponding to $RS(30, 23)$ is above that of $RS(30, 25)$ which in turn, is above that of $RS(30, 27)$. This indicates that the optimal rate allocation scheme is more effective with stronger FEC protection. Also, The above results show that using multiple senders can reduce the irrecoverable packet loss over that of single sender in two typical loss scenarios in the Internet. We should also emphasize that our framework for dividing the rates among senders to reduce irrecoverable packet loss is also applicable to other "bursty" network models which do not necessarily follow the exact "two-state" Markov model. We note that under certain network characteristics and levels of FEC, the optimal solution may result in sending all the packets on one



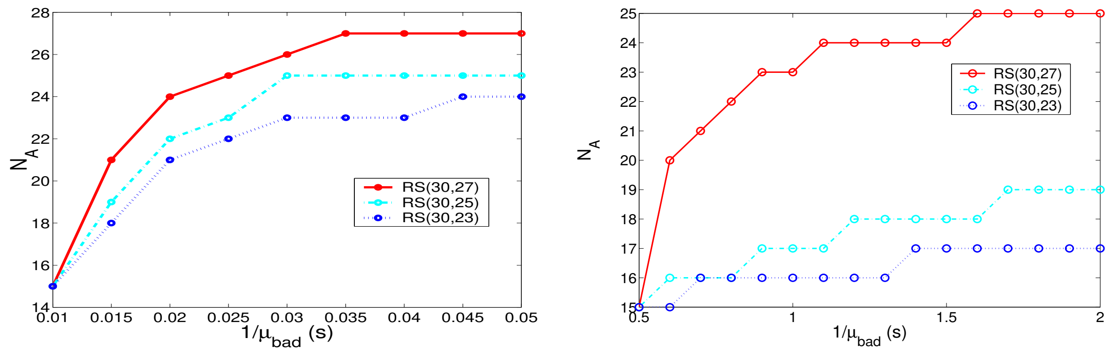
(a) (b)

Figure 3.3: *Probability of irrecoverable loss for various of FEC protection levels as the function of average bad times of route B, using optimal partition for two senders for (a) scenario X and (b) scenario Y.*

route.

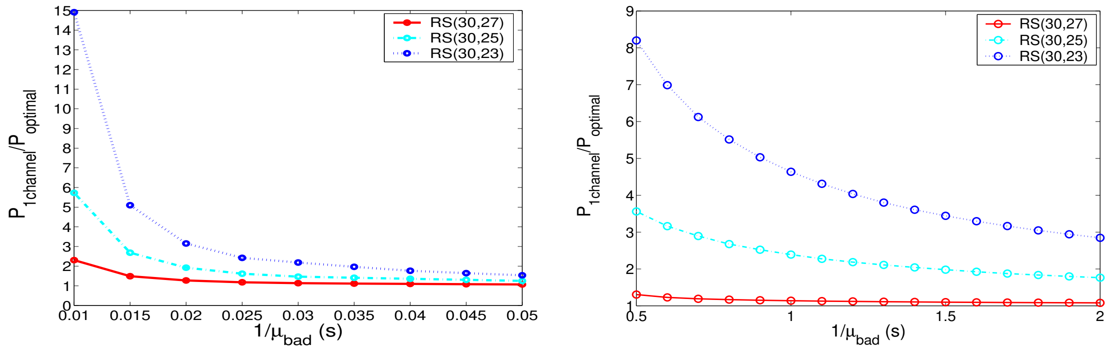
3.5.2 Sensitivity Analysis of Optimal Sending Rate

In this section, we analyze the sensitivity of optimal sending rate, i.e. how much loss probability increases if packets are not sent at the optimal sending rates for each route. This situation can arise due factors such as inaccurate estimates of route parameters or the limitation of the system to react to the changing network conditions fast. Figure 3.6 shows the irrecoverable probabilities for various rate allocations between two senders when routes A and B have identical average good times at 1 second, while the average bad time of route A and B are 10 and 20 milliseconds, respectively. The optimal sending rate for each FEC protection level is at the minimum of each corresponding curve. Assuming inaccurate estimation of average bad time of route B as shown in the brackets, then resulting sending rate on the x-axis varies around



(a) (b)

Figure 3.4: *Optimal partition for various FEC protection levels as the function of average bad times of route B for (a) scenario X and (b) scenario Y;* N_A *denotes the number of packets per 30 sent on route A.*

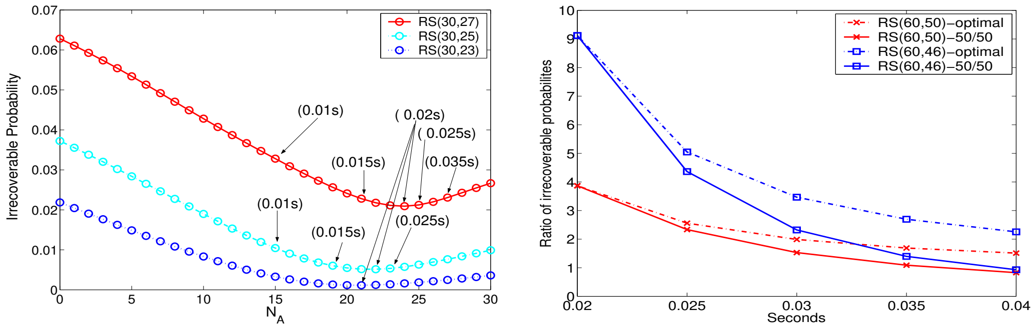


(a) (b)

Figure 3.5: *Ratio of irrecoverable loss probability of sending all packets using only one sender to that of using two senders as the function of average bad times of route B for (a) scenario X and (b) scenario Y.*

the optimal rate as shown in Figure 3.6. Note that the curves with stronger FEC protection levels are more flat at the center than those of weaker FEC protection levels. This indicates that when strong FEC protection is used, a slight variation in sending rates around the optimal values will result in a smaller change in irrecoverable probability than when weaker FEC protection is used. Hence, if delay and bandwidth requirements are satisfied, it is preferable to use stronger FEC protection level to be robust to slight deviations from the optimal sending rate values.

To further characterize these, Figure 3.6(b) shows the irrecoverable loss probability ratios of sending all packets on the better route A over the optimal and over 50-50 rate split between the routes. The average good times of routes A and B are now set to 1 second while the average bad time of route A is set to 20 milliseconds and that of route B varies from 20 to 40 milliseconds as shown in the x-axis of Figure 3.6(b). As expected, the optimal rate split results in the highest ratio, hence largest gain over the single route streaming. As seen, the 50-50 rate split also results in lower irrecoverable loss probability than sending all packets on single route A, especially when strong FEC is employed, and route B is not substantially worse than A. As expected, the more asymmetric the two routes become, the larger the gap between the optimal and 50-50 splits. The actual Internet experiments in Section 3.6.2 set further light on these results. Naturally, the question arises as to why it is beneficial to also use the "worse" route to send any packet on. By "better" and "worse" routes, we mean one with shorter and longer average bad times, respectively. Note that sending



(a) (b)

Figure 3.6: *(a) Irrecoverable probability as the function of different partitions of sending rate between two senders; (b) irrecoverable loss probability ratios between sending all packets on route A over 50-50 and optimal rate splits.*

more packets on a route while it is in the bad state will result in larger number of lost packets. Therefore, when the “better” route A is in the bad state and the “worse” route B is in good state, sending all packets in route A will result in larger number of lost packets than splitting the packets between the two routes. It is true that when route A is in good state and route B is in bad state, splitting packets between the two routes is likely to cause larger number of lost packets than sending all packets in the “better” route A. However, if an appropriately small fraction of packets are sent in route B while it is in bad state, the total number of lost packets per 30 packets is likely to be small enough to be recovered by FEC. Generally, when at least one of the routes is in good state, splitting packets appropriately between two routes will provide a good chance for recovering all the lost packets using FEC. If both routes A and B are in bad state at the same time, then FEC is not likely to be able to recover the lost packets; the probability of this however is quite small. Therefore in most situations, it is advantageous to split the sending rates between the routes.

3.6 Simulation and Experiment Results

In this section, we describe NS simulations and actual Internet experiments to show that our distributed streaming protocol using the proposed rate allocation algorithm results in fewer lost packets, leading to higher visual quality for the streamed video.

3.6.1 NS Simulations

In this section, we simulate our optimal rate allocation scheme using the network simulator NS [46] to demonstrate the benefits of distributed video streaming. In particular, the goal of these NS experiments is to show that larger number of senders results in lower packet loss. The experiment setup is shown in Figure 3.7. There are four identical 500 *kbps* links connecting the senders $S1$, $S2$, $S3$, and $S4$ to the router, and a 5*Mbps* link connecting the router to the receiver. The delays for all the links are 50 *milliseconds*. The queue size of the router is 250 *Kbytes*. The bursty background traffic from all senders to the receiver are simulated using bursts of UDP traffic which follow the exponential distribution with peak rate of 1 *Mbps*, burst and idle times of 200 *milliseconds* and 5 *seconds*, respectively. Since the physical bandwidth of the link between the router and the receiver, 5 *Mbps*, is larger than the aggregate physical bandwidth of all the links between the senders and the router, our assumption on no congestion at the last hop to the receiver holds. For the following experiments, we encode packets of 1000 *bytes* using $RS(100, 90)$. and compare the performance using various number of senders. Since, the traffic characteristics on all

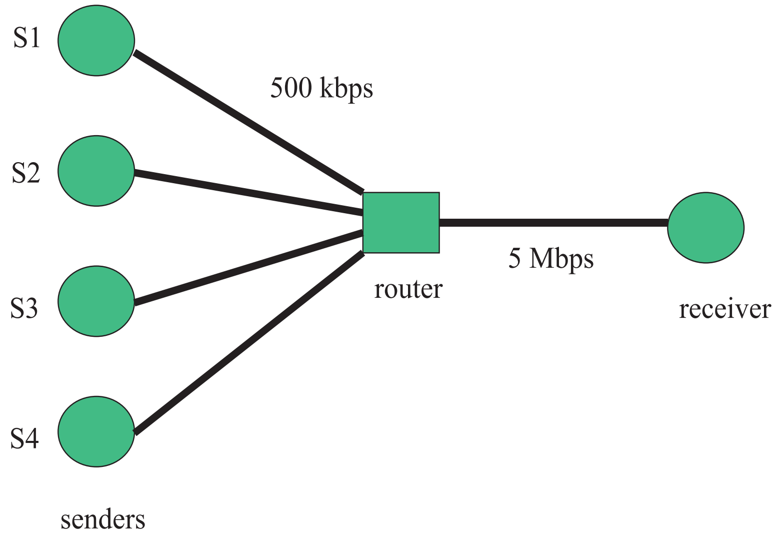


Figure 3.7: *Simulation setup for various number of senders*

four links between the senders and the router are identical, the optimal sending rates from each sender to the receiver should be the identical and equal to the total sending rate divided by the number of senders used to stream the video. In these experiments, the total sending rate is 400 *kbps*. Figures 3.8(a),(b), (c), and (d) plot the number of lost packets per 100 sent packets for one, two, three, and four senders. All the points above the horizontal line represent irrecoverable loss events. As seen, the number of irrecoverable loss events for one, two, three, and four senders are 20, 8, 2, and 2.

This indicates that the number of irrecoverable loss event decreases as the number of senders increases. However, further simulations reveal that increasing number of senders beyond a certain point is no longer beneficial in terms of minimizing number of irrecoverable loss events, as shown in cases of three and four senders.

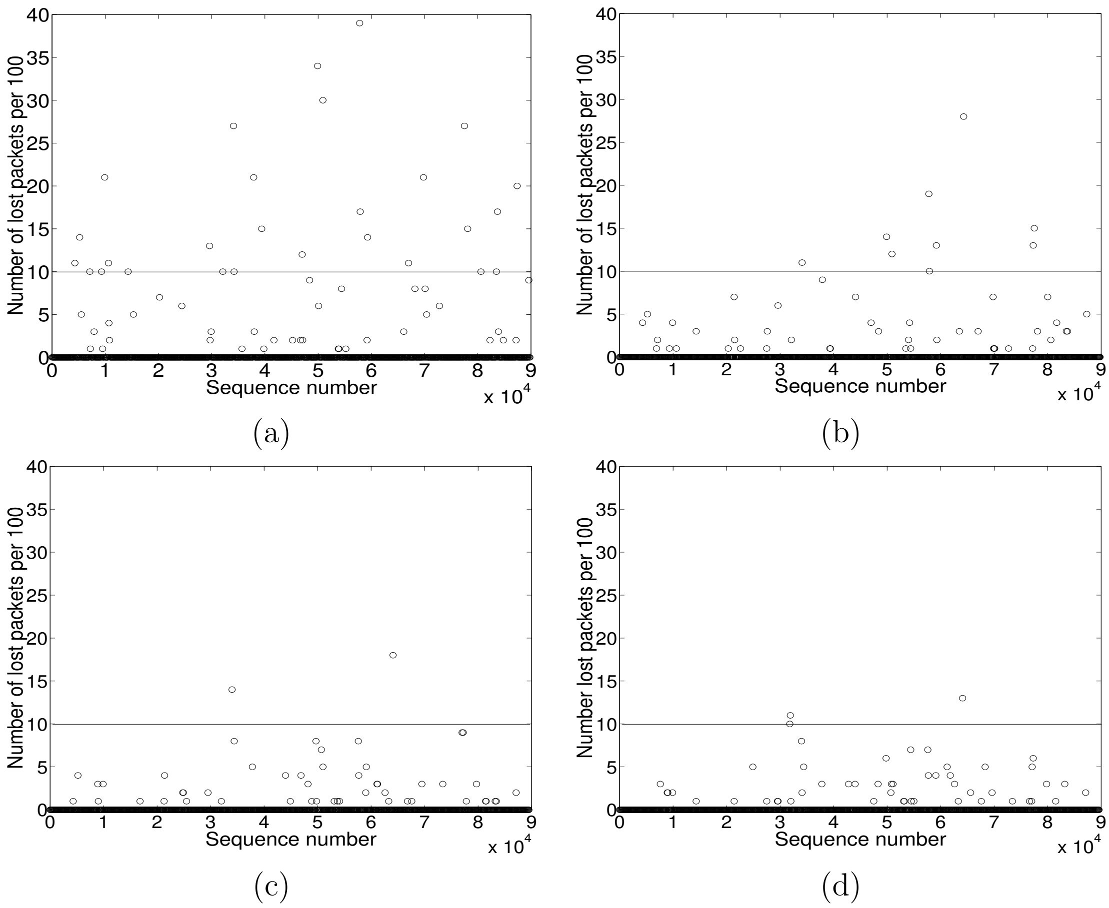


Figure 3.8: *NS simulation showing number of irrecoverable loss events for differnet number of senders (a) one sender; (b) two senders; (c) three senders; (d) four senders*

3.6.2 Internet Experimental Results With Artificially Induced Packet Loss

We have developed a system for distributed video streaming with real-time FEC decoding and display capabilities. We now demonstrate the effectiveness of optimal rate allocation scheme in reducing packet loss. Before describing the experiments, we emphasize that the reduction of packet loss using distributed video streaming scheme over the traditional single path scheme is attributed entirely to the rate allocation between senders, which reduces the bursty loss, hence, increasing the error correction capability of FEC. In essence, the packet partition algorithm in Chapter 2 provides the necessary machinery and logistics, to make the overall system, including the rate control algorithm to function properly, and hence is not directly responsible for reducing packet loss. Also the goals of PPA and rate allocation algorithms are different and in some sense complementary, and hence even though the two interact with each other, they cannot be combined. Therefore, we do not anticipate any improvements by combining the two, and examine their performance separately.

We perform the following experiments to compare traditional uni-sender streaming with distributed streaming using multiple senders. In experiment one, we use one sender in Belgium to stream all packets to U.C. Berkeley. In experiment two, one sender at Belgium and the other at Sweden simultaneously stream packets to the receiver at U.C. Berkeley. In both experiments, we use $RS(60, 46)$ with packet size of 500 bytes, and set the total sending rate to 200 packets per second, i.e. 800 kbps. In both experiments, we artificially induce packet loss at the senders using the Markov chain model mentioned earlier; however, other factors such as round trip time remains faithful to the characteristics of the routes. The observed average round trip times between Belgium and U.C. Berkeley is 152 milliseconds, and that between Sweden and U.C. Berkeley is 199 milliseconds. Throughout the experiments, we assume that 800kbps bandwidth is available at all times for a single sender to stream the packets.

For both experiments, initially, the average good and bad times of both connections are set at 1 and 0.02 seconds, respectively, and the packet loss probability in bad state is set to $p = 1$. These parameters result in an average packet loss rate of 2%. During the first 200 seconds in experiment one, all packets are sent using only Belgium sender, resulting in 8 irrecoverable loss events as indicated by the number of circles above and to the left of the horizontal and vertical lines in Figure 3.9(a). During the first 200 seconds in experiment two, sending rates are split equally between Belgium and Sweden senders, resulting in no irrecoverable loss events as seen in Figure 3.9(b). This is due to the reduction in burst loss by sending packets at lower rates on both routes, and hence, the increased error correction capability of FEC. From $t = 200$s to $t = 600$s, we increase the average bad time of Belgium connection to 0.04 seconds, resulting in 66 irrecoverable losses for experiment one. On the other hand, for experiment two, there are only 6 irrecoverable losses during $t = 200$s to $t = 400$s where sending rates are still split equally. This indicates that splitting sending rates between senders sup-optimally, can still provide some degree of robustness to sudden change for the worse in network characteristics.

Further reduction in packet loss can be achieved by sending packets at the optimal rates. In particular, for experiment two, during the period from 400 to 600 seconds, the Belgium and Sweden senders adjust their sending rates to optimal levels, i.e. 60 and 140 packets per second for Belgium and Sweden, respectively. During this period, there is no irrecoverable loss events as shown in Figure 3.9(b). This suggests that if accurate network parameters are available, then sending packets at the optimal rates results in further packet loss reduction as compared to dividing packets roughly evenly especially as the loss difference between the two routes widens. Figure 3.9(c) shows the throughputs of two senders in experiment two. As seen, the variation in throughputs is mainly due to packet loss. In particular, we have found the throughput dips of Belgium connection during the period from t = 200s to t = 400s to be on average larger than that of Sweden sender due to higher number of lost packets for Belgium sender during this period [2]. From t = 400s to t = 600s, since the sending rate of Belgium sender reduces to 60 packets per second, these throughput dips become smaller than before, providing FEC a better chance to recover the lost packets.

For completeness, Figure 3.9(d) shows the histogram of difference in sequence number of consecutive received packets. As seen, most of the packets are within 5 packets of each other. This indicates the effectiveness of the rate allocation and packet partition algorithms to send closely interleaved packets.

[2] This is best seen on the color printout of this plot.

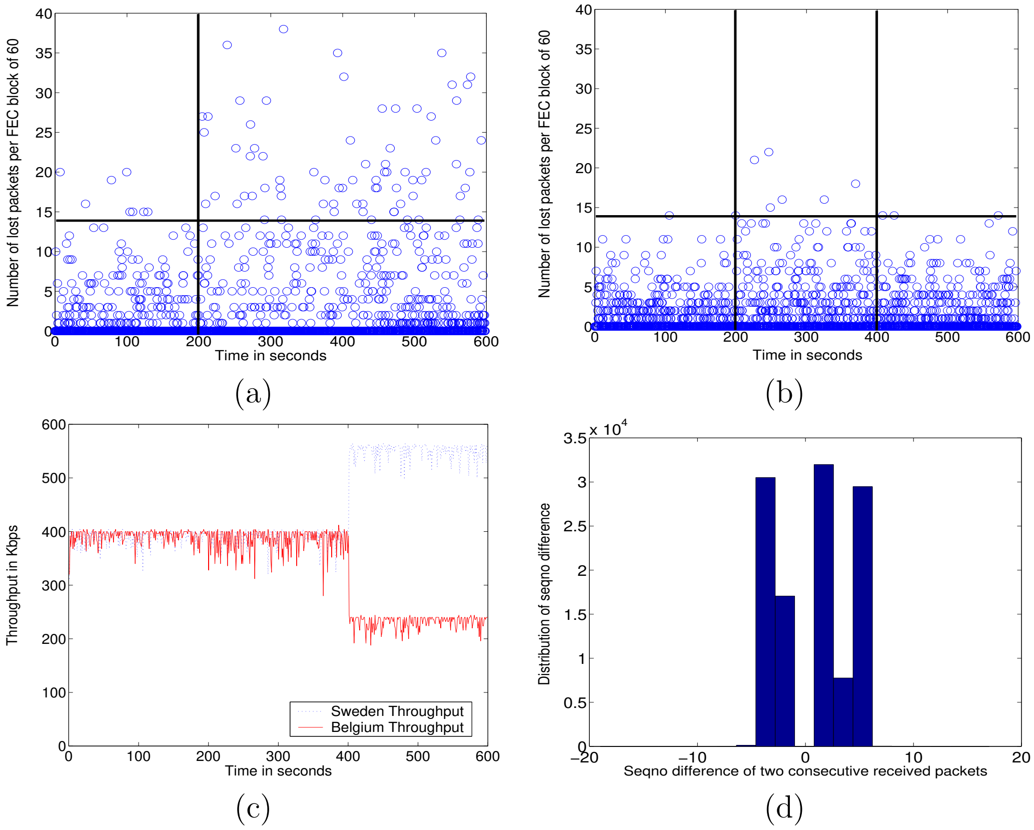


Figure 3.9: *Internet simulations showing the number of lost packet per FEC block of 60 packets vs. packet sequence for (a) streaming from Belgium alone; (b) streaming from Belgium and Sweden; (c) Throughputs of two senders; (d) Histogram of variation in packet order.*

3.6.3 Actual Internet Experiments

We also perform actual experiments over the Internet. In experiment three, a sender at Purdue University streams a H.263 encoded video to a receiver at U.C. Berkeley at the rate of 200 packets per second. In experiment four, the same video is also streamed at 200 packets per second from a sender at Sweden to a receiver at U.C. Berkeley. In experiment five, both senders at Sweden and Purdue University simultaneously stream the video to a receiver at U.C. Berkeley with the optimal rates of 80 and 120 packets per second, respectively. In all three experiments, the streamed H.263 video has constant bit rate of $720kbps$ and is packetized into 500 bytes packets which are then protected using $RS(100, 90)$ code. To compute the optimal sending rate, we estimate the network parameters for Sweden-Berkeley and Purdue-Berkeley routes, using a Hidden Markov Model inference algorithm [47] on the traces of packets over many hours offline. An online algorithm has also been developed and discussed in Appendix B.2. To capture the bursty characteristics of the network, we ignore all the single loss events, and only use consecutive burst loss of 2 or more packets in our estimates of network parameters. Although many single losses may change the network characteristics, we observe that in our experiments, there are rarely multiple single losses within a FEC block, hence these losses can be recovered by FEC most of the times. Therefore, we only consider bursty characteristics to determine the sending rates. In our experiments, the average congestion intervals for Sweden-Berkeley and Purdue-Berkeley are estimated to be approximately 39 and 33 milliseconds while the average good times are 6.1 and 6.9 minutes, respectively.

Figure 3.10 plots the number of lost packets per 100 for experiments three, four, and five denoted with squares, circles, and crosses, respectively. The points above horizontal line represent irrecoverable loss events. Since we are using $RS(100, 90)$, irrecoverable loss happens when there are more than 10 lost packets per 100 sent packets. As seen, there are 5 instances of irrecoverable loss for experiments three and four where only one sender is used to stream video to receiver. On the other hand, in experiment five where both senders at Sweden and Purdue university stream video simultaneously to the receiver at U.C. Berkeley, all the lost packets are successfully recovered by FEC. Note that even though the average packet loss rates for experiments three, four, and five are 0.05%, 0.09%, and 0.08%, i.e. well below 10%, $RS(100, 90)$ code in experiments three and four cannot recover all the lost packets due to the bursty loss nature of Internet. We have also found uni-path schemes result in 13 to 20 dB instantaneous drop in PSNR at the time of irrecoverable loss events as compared to distributed video streaming scheme. For visual comparison, we show the typical resulting images for both schemes at an irrecoverable loss events in Figures 3.11(a) and 3.11(b).

We have also performed experiments using different pairs of sending rates for distributed streaming from Purdue University and Sweden: (100, 100), (104, 96), and (112, 88) packets per second. We have found the difference in irrecoverable packet loss between the above sending rates and the optimal rates to be small. We conclude that in situations where senders exhibit small and roughly identical loss characteristics, splitting packets approximately equally among the senders is good enough. To further show the robustness of the distributed streaming, we perform many uni-sender and and multi-sender experiments for different sending rates and two different levels of FEC protection: RS(60,46) and RS(60,50). The senders in the experiments are at Sweden and Hong Kong, and the receiver is at U.C. Berkeley[3].

[3]This Sweden site is not the same as the Sweden site in experiment four. Both sites planetlab-1.it.uu.se at Sweden and s1_803.ie.cuhk.edu.hk at Hong Kong are part of PlanetLab sites.

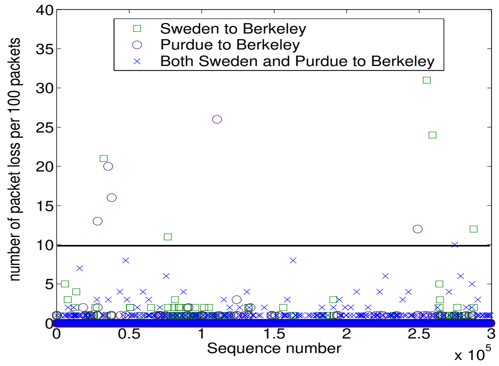


Figure 3.10: *Actual Internet experiments showing the benefits of path diversity video streaming over conventional approach.*

Packet size is set to 500 bytes and total sending rates is set to 220 packets per seconds. All the experiments are performed between 1 and 4 PM PST, and the duration of each experiment is set to 15 minutes. The results of these experiments are shown in Table 3.3. Sending rates in packets per second are shown in columns 1 and 2. The numbers of irrecoverable loss events are shown in columns 3 and 4, and the ratios of irrecoverable losses of multi-sender over uni-sender are shown in columns 5 through 8.

As seen in Table 3.3, distributed streaming from multiple senders clearly reduces the number of irrecoverable losses, up to 10.2 times over uni-sender streaming. Even though the average packet loss rate of Sweden sender, i.e. 1.3%, is lower than the average loss rate of Hong Kong, i.e. 1.8%, we have empirically found the Sweden's loss patterns to be more bursty; hence, it is advantageous to send packets at lower rate on the Sweden route. In this scenario, if one sends all the packets on the route with lower average loss rate, namely Sweden route, the number of irrecoverable loss

(a)

(b)

Figure 3.11: *Typical images at an irrecoverable loss event for (a) conventional video streaming and (b) path diversity video streaming.*

is even larger than sending all the packets on the route with larger loss rate, namely Hong Kong route. This indicates that when appropriate amount of FEC is used, the lower average loss rate of a particular route is not necessarily a good indicator for sending packets at higher rate on that route, rather the burstiness loss patterns should be taken into account for setting the sending rates between routes.

Results in Table 3.3 further verify our earlier numerical results in Section 3.5.2 re-

Sweden	Hong Kong	Irrecoverable Loss		Gain over Hong Kong		Gain over Sweden	
		RS(60,46)	RS(60,50)	RS(60,46)	RS(60,50)	RS(60,46)	RS(60,50)
0	220	18	28	1	1	2.8	2.4
20	200	14	31	1.3	0.9	3.6	2.2
40	180	9	30	2	0.9	5.7	2.4
60	160	5	9	3.6	3.1	10.2	7.6
80	140	7	9	2.6	3.1	7.2	7.6
100	120	12	20	1.5	1.4	4.3	3.5
220	0	51	69	0.4	0.4	1	1

Table 3.3: *Irrecoverable loss reduction using two senders for various sending rates and FEC levels.*

garding graceful degradation of irrecoverable loss as we deviate from optimal partition of the packets between the two routes. Specifically, while the number of irrecoverable loss events is at its minimum for 60/160 split, deviation to 40/180 or 80/140 results in slight increase in irrecoverable loss events. This suggests that in a highly dynamic environment where network parameters change frequently, and therefore, it is hard to estimate them accurately, splitting packets sup-optimally between the senders may still provide advantages of uni-route streaming, even though optimal partitioning always results in best performance. Therefore, if enough FEC is used, control packets could be sent infrequently to avoid frequent adaptation to "noise" to achieve reasonable performance.

3.7 Conclusions

In this chapter, we proposed a novel rate allocation algorithm for use with FEC as part of the *path diversity* streaming protocol in Chapter 2. The rate allocation scheme determines the sending rate for each sender based on the available network bandwidth, amount of Forward Error Correction (FEC), and channel characteristics in order to minimize the probability of packet loss in bursty loss environments. Our rate allocation using FEC is developed based on the observation that sending packets at a lower rate on multiple independent routes transforms the bursty loss behavior into a more uniform loss behavior, thus increasing the efficiency of FEC techniques. We also provide a robustness analysis of the optimal sending rates. Using NS simulations and actual Internet experiments, we demonstrate the effectiveness of our rate allocation scheme in reducing packet loss over the traditional uni-path approach, and hence, achieving higher visual quality for the streamed video.

Chapter 4

Path Diversity System (PDS) for Single Sender

4.1 Introduction

In Chapter 2, we propose a framework for path diversity media streaming over the Internet using multiple senders. In this approach, multiple senders simultaneously stream the precoded video to a receiver using a receiver-driven protocol. While this approach increases the throughput and reduces packet loss as compared to the uni-path approach, its architecture inherently does not enable live or interactive media streaming. In this chapter, we extend our path diversity concept in previous chapters to propose a single sender, single receiver, Path Diversity System (PDS) for delay sensitive applications over packet switched networks.

The proposed PDS is essentially an *overlay network* connecting a set of participating nodes. An overlay network is a "virtual" network created on top of an existing network. Nodes on the overlay network can be equivalently thought of as routers in the physical networks. In the current Internet, a sender cannot choose which routers in the underlying physical networks for its packets to transverse to the destination since the routing of packets is accomplished by routing protocols such as BGP [48] and OSPF [2]. On the other hand, the sender can choose which "overlay" paths in the overlay network by sending packets to an intermediate node, which then forwards the packets to the destination.

In this chapter, we propose a scalable algorithm for choosing the intermediate node which creates an overlay redundant path consisting of few or no shared underlying physical links with the default Internet path. The reason for this is we have shown in previous chapters that the performance of FEC in multi-sender and multi-path scenario depends on the correlation of packet loss between paths. If the packet loss between multiple paths are highly correlated, then the advantage of using multiple paths, together with FEC for streaming is reduced. Also, the quality of video of many multiple description video coding (MDC) schemes [22][49] are higher in presence of uncorrelated than correlated loss of descriptions [36][50]. Hence, the central question is whether there exists sufficiently disjoint paths between a pair of senders and receivers on the Internet to result in mostly uncorrelated loss patterns between paths. If the paths are not entirely disjoint, then the probability of congestion on the shared links between the paths must be small in order to minimize the overall correlated loss.

Recent studies in [51] indicate that at least 60% of the Autonomous System (AS) are *multi-homed* to two or more providers, i.e. sufficiently disjoint paths to the receiver can be created using the relay nodes in different providers. Recent work [52] using RON, has shown how to route packets around all the observed outages between any pair of senders and receivers in an experimental network. Also, many Internet topological models such as *Albert-Barabasi* [53] also exhibit a high level of disjointness between the paths connecting two nodes. These suggest the existence of path redundancy between nodes on the Internet.

The remainder of this chapter is organized as follows. In Section 4.2, we present relevant work. In Section 4.3, we provide an overview of the proposed PDS and a scalable algorithm for redundant path selection. In Section 4.4, we characterize the disjointness and delay between the redundant and default paths for various Internet topologies. We then show the overall performance improvement using both FEC and MDC [22][49] techniques with the proposed PDS for streaming. Finally, we conclude in Section 4.5.

4.2 Related Work

The original motivation of overlay network was to extend the multicast functionality across area where the Internet did not natively support multicast. MBone is one of the well-known large overlay networks deployed on the Internet [54] for wide area multicast. Beside multicast, overlay networks also provide the flexibility for the application to choose which paths to send the packets on via the relay nodes. Because of this flexibility, many systems have used the overlay framework to select the optimal path between sender and receiver for load balance as well as loss rate and delay. For example, Detour [55] is an architecture that uses overlay network to dynamically route packets on different routes to optimize loss and delay. To create a redundant path in the overlay framework, the sender sends packets to a relay node, and the relay node then forwards packets to the receiver [23][52]. By selecting the relay node appropriately, the packets traveling through the relay node take a different underlying physical path than that of the Internet default path between the sender and receiver. From traffic engineering point of view, RON [52] provides alternate paths between sender and receiver using relay nodes. RON actively probes for the current "best" path based on delay and loss, and sends all packets through that path. All of the existing overlay systems assume a single "best" path for sending data. In contrast, our proposed path diversity system uses multiple paths simultaneously for sending data.

Path diversity can also be accomplished using source based routing supported at the routers. Using this approach, one can explicitly specify a set of nodes for each packet to transverse to the destination. Currently, source based routing is available only to a few autonomous systems (AS). Hence, path diversity is accomplished by sending packets on both the default path and the redundant path.

4.3 Path Diversity System (PDS)

4.3.1 System Description

In this section, we describe the system architecture for path diversity based on overlay framework [52][55]. As seen in Figure 4.1, the system consists of a set of participating nodes. Circles represent participating overlay nodes, squares denote routers, and the bold solid and dashed lines represent the underlying physical and virtual paths between the nodes, respectively. The thin vertical solid lines connecting circles and squares represent the correspondence between virtual nodes and physical routers. At any instance, a node can act simultaneously as a receiver, a sender, or a relay node. A sender can send packets to the receiver using the default Internet path or via a relay node which then forwards the packets to the receiver. By choosing an appropriate relay node, the packets traverse an underlying physical path that is different from the one used by default Internet path. Nodes can be deployed at various locations on the Internet, although redundant paths via nodes in the same Autonomous System (AS) may have larger number of shared links. Hence, nodes are preferably located in different AS to reduce correlated congestion and outages so as to improve the performance of media streaming as shown in previous chapters.

A participating node which is neither a receiver nor a sender, may still receive and forward packets on behalf of other senders and receivers. For two-way applications such as video conferencing, the nodes act simultaneously as both senders and receivers. In addition, the senders and receivers do not necessarily have to be the participating nodes. We envision the participating nodes as an overlay network to be deployed by companies or organizations that are interested in providing low delay communication services such as video streaming or conferencing over the Internet between their geographically diverse sites. Companies and organizations typically have multiple gateways from their ASes to other ASes that potentially enable large number of independent paths [56].

To set up a communication channel between the sender and receiver, the sender first executes *traceroute* [57] from itself to all the participating nodes and the receiver. The information returned from the *traceroute* includes the link latencies, and the names of the routers along the default path from the sender to the receiver and all paths between the sender and the participating nodes. The sender also sends a setup packet to all participating nodes instructing them to execute *traceroute* from themselves to the receiver. Next, all the participating nodes send the path information between themselves and the receiver obtained from *traceroute* to the sender. The sender now has the names of the routers and their associated link latencies for the default path, the paths between itself to all participating nodes, and the paths between participating nodes to the receiver. This information is used to compute the redundant path as described later in Section 4.3.2.

After the redundant path via a chosen relay node is selected, the sender sends a setup packet to the selected relay node, instructing it to forward packets to the receiver on behalf of the sender from then onward. The setup packet contains a flow ID, IP address, and the port number of the receiver. Upon receiving the setup packet, the relay node stores an entry containing a flow ID, an IP address, and a port number of the receiver in a table. This table is used to forward packets on behalf of different senders to their receivers. Each packet sent from a different sender to the relay node contains a different flow ID in its header. The relay node forwards a packet to the right IP address and port number of the receiver based on its flow ID. Note that all the setup messages and executions of *traceroute* are done only at the start of the session. Also, the information returned from trace route for earlier session can be saved for future session.

Due to delay consideration, we choose to only use one relay node to forward packets between a pair of sender and receiver. The reason is that the delay of a redundant path via multiple participating nodes in series is likely to be larger than that of the redundant path via only one node. However, our proposed PDS can be easily extended to the case where there are multiple redundant paths via multiple relay nodes.

We use UDP to send all packets between participating nodes since TCP is not appropriate for real-time data transmission due to its variable throughput, and lack of precise rate control. In addition, one of the main concerns with using multiple paths to send packets is that packets likely arrive at the destination out of order. For UDP traffic, this is not a concern since a reordering buffer at the receiver can be used. For TCP traffic, however, out-of-order packets are treated as packet loss, and TCP *fast retransmit* algorithm continually resends packets which are merely in transit, reducing the connection bandwidth.

One major difference between our PDS and RON [52] is that RON dynamically determines the "best" path to send all packets on, while our PDS sends packets simultaneously on multiple paths, similar to Detour [55]. Detour, however, uses the multi-path at the router level whereas we accomplish multi-paths at the application level, allowing the flexibility for the application to decide which packets should be sent on which paths.

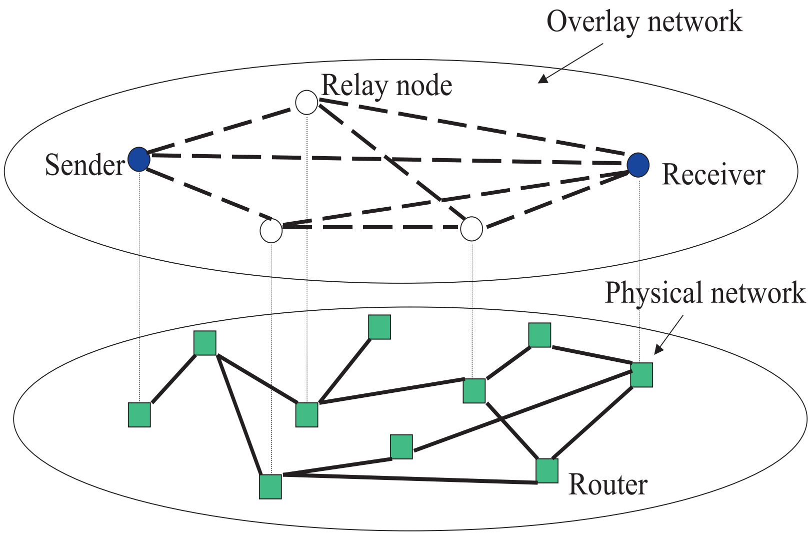


Figure 4.1: *System architecture; bold dashed and solid lines denote virtual and physical paths; The thin vertical solid lines connecting circles and squares represent the correspondence between virtual nodes and physical routers.*

4.3.2 Redundant Path Selection

In this section, we propose a scalable, heuristic method for finding a redundant path via a participating relay node. Selecting an optimal path between a pair of nodes on the Internet at any one instant is difficult and complex. If the traffic conditions of the path vary rapidly, the problem becomes almost infeasible. For scalability reasons, the Internet domain routing is handled primarily by the Border Gate Protocol (BGP) [48]. BGP only exchanges and updates summarized information between ASes, ignoring the link usages and topologies within ASes. Hence, accurate path information such as number of links along the path and their associated latencies can not be obtained through BGP. Other link state routing protocols such as Open Shortest Path First (OSPF) [2] periodically probe the links between the routers for updated information such as latency, bandwidth, and link failures. OSPF can provide these information reasonably accurately, however, it is not scalable and therefore, is only used within ASes.

To measure end-to-end latency, bandwidth, and packet loss between nodes at various locations on the Internet, probing tools [58][59][60] can be used at the expense of bandwidth expansion due to sending probed packets.

Another approach is to use passive probing in which the application packets themselves are used as the probing packets, to determine the packet loss rate, latency, and bandwidth. The advantage of this approach is lack of bandwidth expansion; however, its drawback is that the measurement process depends on the application sending rates. If an application sends packets at a slow rate, the measurement resolution is low, resulting in inaccurate end-to-end estimates of packet loss rate and bandwidth.

In contrast to scalable designs such as BGP in which, the routing and link state information is kept to a minimum, systems such as RON use more complex path computation algorithms to increase path performance. A RON node aggressively probes all the paths connecting itself to other participating nodes in order to estimate detailed link quality metrics such as loss rate, bandwidth, and latency. Also, each RON node can exchange complete topologies and execute complex routing algorithms to improve the path performance, and to respond rapidly to path outages. All these benefits come at the expense of scalability, as the number of RON nodes is usually limited to fewer than 50 [52].

Our approach is to use a simple, but suboptimal technique for selecting redundant paths. In particular, we argue that finding two paths with absolute lowest loss rates for the proposed PDS may not be needed in practice due to two reasons: (a) complexity increases due to active monitoring of probed packets and maintaining the link state information associated with all the paths i.e. scalability reasons, and (b) sending packets on two paths with the absolute lowest loss rates may not be necessary to achieve reasonable performance in PDS. This can be justified considering that as long as packet loss does not happen simultaneously on both redundant and default paths, by using MDC, reasonable video quality at the receiver can be reconstructed using the received description from other good path.

Based on the above discussion, we propose a scalable, heuristic scheme to select the redundant path via a participating node using path information from *traceroute* tool [57]. *Traceroute* can provide the names of the routers and round trip delays of links between the two communicating nodes on the Internet.

Let us formally denote a network topology as directed graph $G = (V, E)$ consisting of the vertices $v_i \in V$ and edges $e = (v_i, v_j) \in E$. Vertices v_i's can be thought of as routers or domains, and the path $p(v_1, v_n) = [v_1, v_2, ..., v_n]$ as the physical path from v_1 to v_n. A redundant path $p'(v_1, v_k, v_n)$ from v_1 to v_n, via node v_k is then $p(v_1, v_k) \cup p(v_k, v_n)$. Associated with every pair of vertices (v_i, v_j) is a weight $w(v_i, v_j)$. This weight can be thought of as delay, bandwidth, or loss rate associated with the physical link between v_i and v_j. Hence, the weight $w(p(v_k, v_l))$ associated with a path from v_k to v_l is the sum of the weights of the individual physical links joining v_k and v_l. In this chapter, weights denote the latencies between participating nodes since they are readily available from *traceroute*. Let $O = [v_m, ..., v_n]$ be the set of participating nodes; then the relay node k' for creating the redundant path between vertices u and v is computed in a two-step procedure.

1. First, we compute a set of relay nodes O' that result in the minimum number of joint links between the default Internet path and all the redundant paths via a node in O, namely,

$$O' = \arg\min_{k} p'(u, k, v) \cap p^*(u, v)$$

where $k \in O$, $p'(u, k, v)$ denotes the redundant path via node k, and $p^*(u, v)$

denotes the default Internet path. Note that the set O' can have more than one element since there could potentially be two or more nodes in O that result in the same minimum number of joint links between the default and redundant paths. Since the average number of links between two nodes on the Internet is 16 [28], the operation $p'(u,k,v) \cap p^*(u,v)$ can be done fast. To find $\arg\min_k p'(u,k,v) \cap p^*(u,v)$, exhaustive search for the set of nodes O' that results in minimum number of joint links between the redundant and default paths can be done in $O(N)$ with N being the number of participating nodes.

2. Next, we choose the node k' that results in minimum weight associated with the corresponding redundant path, namely,

$$k' = \arg\min_{l} w(p'(u,l,v))$$

where $l \in O'$.

Note that it is the sender that runs the redundant path selection algorithm, and that the input to the algorithm is the path information, specifically, the names of routers and the associated link latencies of the default path $p(u,v)$ and the redundant paths $p'(u,l,v)$. The names of the routers are used in the first step to compute the number of shared links between the redundant and default paths while the latencies are used in the second step to find the path with minimum delay. As mentioned previously in Section 4.3.1, the link latencies and names of routers along the path $p(u,v)$ are readily available using *traceroute* from node u to v. The latency and router information for redundant path $p'(u, k, v)$ via node k can be obtained by executing *traceroute* twice, once from nodes u to k, and another time from nodes k to v. The information returned from the two *traceroute* executions is then concatenated to form the path $p'(u, k, v)$. Note that sender either runs this algorithm at the beginning of the new session, or it can use the stored paths provided from previous session since the paths are relatively stable. This will reduce the overhead of executing *traceroute* unnecessarily.

Intuitively, the path selection scheme first finds a set of redundant paths that are as disjoint as possible from the default path. Within this set of redundant paths, it then selects the one that results in minimum latency. An alternative might seem to be to select the redundant path based on traffic characteristics of each link along the path between two nodes. However, since we do not have knowledge of loss rates and bandwidths for individual links, we choose the redundant path to be maximally disjoint with the default Internet path so as their losses are uncorrelated; this results in the PDS to be effective in minimizing correlated loss.

In case the latency of the redundant path exceeds the the desired delay, that redundant path is ignored, and the same two-step procedure is applied to all the remaining relay nodes $k \in \{O \setminus O'\}$ to select a new redundant path. After each iteration, the number of shared links for the redundant path increases.

Note that our PDS does not aggressively send out probing packets to monitor the current loss rates and bandwidths between participating nodes. Instead, it simply uses the route information between participating nodes, and only invokes *traceroute* at the start of the session. Hence, our solution is scalable, even though its performance is potentially lower than that of non-scalable systems [52] with aggressive probing for network conditions.

One of the drawbacks of PDS is that its performance depends on the information provided by *traceroute*. This information can be incomplete or inaccurate. For example, *traceroute* can only differentiate between routers and not switches. Two paths with completely different routers may share the same backbone switches. Hence, if that backbone switch is overloaded, packet loss between routes can be correlated. Although, packet loss rate in backbone switches is small, it is still a concern.

Another drawback is that some ASes do not report accurately or deliberately hide information about their networks; only certain routers are visible to outside and therefore, complete information is not available.

4.4 Simulation Results

4.4.1 Simulation Results for Hop Counts, Disjointness and Latency of Redundant Path

In this section, we characterize the disjointness, hop counts, and latency of the redundant path using the proposed scheme for various network topologies. We use the Internet topology generator software Brite [61] to generate various *Albert-Barabasi* topologies for all the simulations. *Albert-Barabasi* model has been shown to approximate the Internet topology reasonably well [53][61]. In particular, we generate two, two-level hierarchical *Albert-Barabasi* models, and one flat *Albert-Barabasi* model. All topologies contain 1500 nodes each. The top level of the two-level hierarchical topology models the collection of ASes while bottom level models the routers within an AS. The two two-level *Albert-Barabasi* models are meant to model wide area Internet topology consisting of many ASes with various degrees of interconnectivity, and the flat *Albert-Barabasi* model represents the router interconnectivity within an AS as shown in Figures 4.2 and 4.3, respectively . The relevant information for each simulated topology is listed in Table 4.1. *H-Albert-Barabasi* stands for hierarchical *Albert-Barabasi* model. To estimate the average latency, hop counts, and the degree

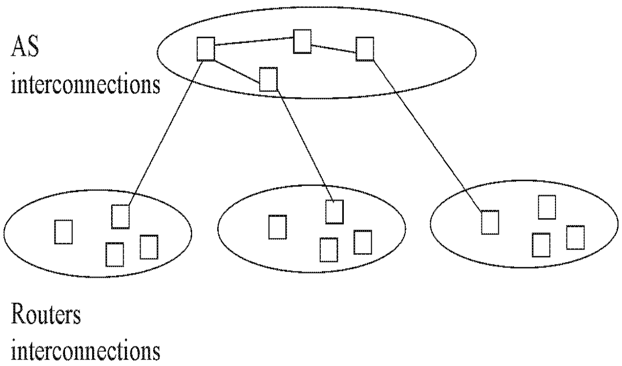


Figure 4.2: *Two-level hierarchical topology*

of disjointness between the redundant path and the default Internet path, we perform the following three steps in our simulations. In step one, we randomly choose a set of participating nodes. In step two, we randomly choose a pair of receiver and sender in the set of participating nodes. In step three, we use our scheme in Section 4.3.2 to

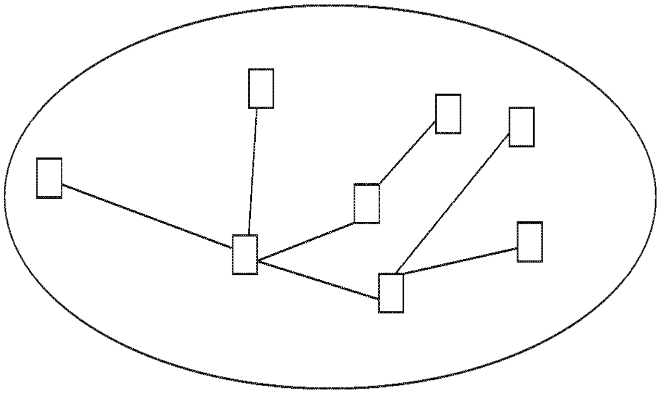


Figure 4.3: *Flat topology*

find the redundant and the default paths for a given configuration of sender, receiver, and participating nodes. The default path in our simulations is assumed to be the one with smallest latency or equivalently the shortest path between the sender and receiver, and hence can be computed using OSPF algorithm [62] for a given topology. This assumption is not critical to our redundant path selection scheme as we merely need a way to compute a default path.

Next, we repeat step one to three over 5000 times to obtain the average latency, hop counts, and the number of shared links between redundant and default paths. Define the jointness percentage between the default and redundant paths as the num-

Models	No. Nodes	No. Edges
Flat *Albert-Barabasi*	1500	2967
H-Albert-Barabasi I	1500	2997
H-Albert-Barabasi II	1500	4377

Table 4.1: *Information for various topologies*

ber of shared links between them over the number of links of the default path. Figure 4.4 shows the jointness percentage between the default and redundant paths for all three topologies as a function of percentage of participating nodes. As expected, the jointness percentage decreases as the number of participating nodes increases for all three topologies. This phenomenon is intuitively plausible since a redundant path is created via a participating node. Larger number of participating nodes allows more choices of redundant paths, thus producing more disjoint paths than a configuration with fewer participating nodes. As seen in Figure 4.4, on average, to achieve less than 10% shared links or less than one shared link between the default and redundant paths for all three topologies, only 2% of total nodes are required to be participating nodes. Another observation is that the percentage of shared links is smaller for *Albert-Barabasi* II model than that of *Albert-Barabasi* I. The reason is that *Albert-Barabasi* II has larger degree of interconnectivity between its nodes than that of *Albert-Barabasi* I; hence, more disjoint paths can be found. Note that the flat *Albert-Barabasi* model has the lowest ratio of all three topologies due to following reasons. As mentioned previously, the top level of the two-level hierarchical *Albert-Barabasi* topologies represents interconnectivity between AS while the bottom level represents the interconnectivity between routers within an AS. If two participating nodes are located in two different ASes, all the paths between them must go through a few AS nodes. As a result, the redundant paths may share larger number of links with the default path than with a flat topology.

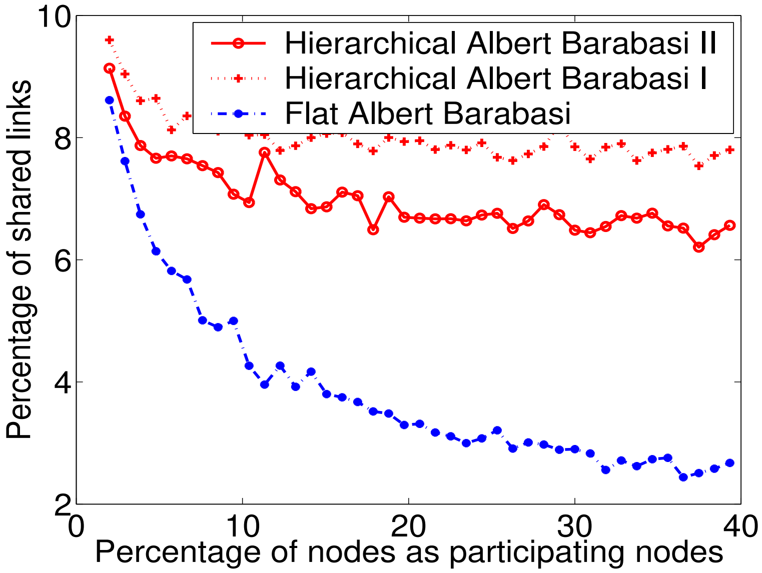


Figure 4.4: *Percentage of shared links between the redundant and the default paths.*

Figure 4.5 shows the ratio of average latency of the redundant path to that of the default path as a function of percentage of participating nodes for all three topologies. As expected, the latency decreases as the number of participating nodes increases for all three topologies. This phenomenon is intuitively plausible since as the number of participating nodes increases, there are more choices for redundant paths, one of which is likely to have shorter latency. Another observation is that the latency ratio is smaller for *Albert-Barabasi* II model than that of *Albert-Barabasi* I. The reason is that *Albert-Barabasi* II has a larger degree of interconnectivity between its nodes than *Albert-Barabasi* I, thus resulting in redundant paths with lower latencies. For small percentage of participating nodes, the latency ratio of redundant over default paths is as high as 1.7. In this case, if the round-trip time of the default path is

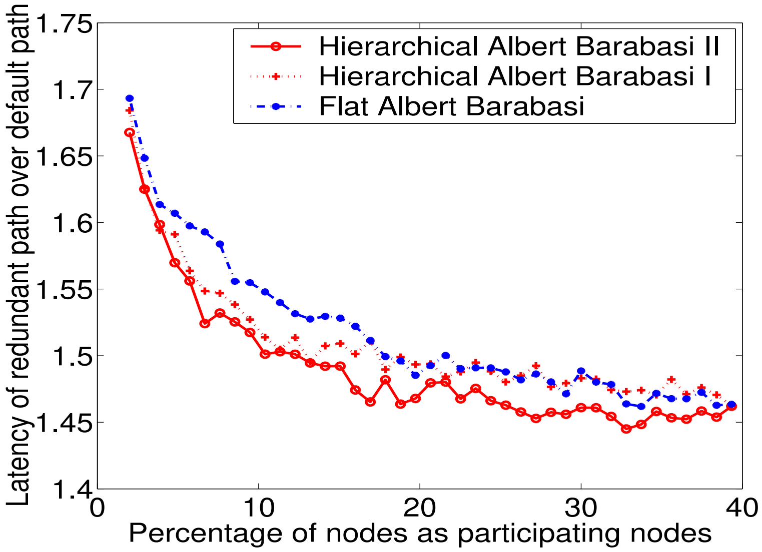


Figure 4.5: *Latency of redundant path over the latency of default path.*

100 *milliseconds*, then the average round-trip time of the corresponding redundant path is 170 *milliseconds*, thus exceeding the tolerable delay of 150 *milliseconds* for two-way interactive applications. This problem can be remedied by either increasing the percentage of participating nodes to 10%, or by allowing the redundant path to share larger number of links with the default path. With the default round-trip time of 100 *milliseconds* and with 10%, or more participating nodes, the average latency ratio of redundant over default paths is 1.5, corresponding to the delay of 150 *milliseconds* for the redundant path, thus satisfying the requirements for two-way interactive applications. The fact that typical round-trip time between two sites within North America is less than 100 *milliseconds* makes the deployment of our proposed PDS feasible from a practical point of view. For example, the average

round-trip times from Georgia Tech, Purdue university, and Duke university to U.C. Berkeley are 62, 57, and 90 *milliseconds*, respectively.

An important observation to make is that for all three topologies, the latency ratios and the jointness percentages decrease rapidly when the percentage of participating nodes is less than 20%, and decrease rather gradually beyond 20%. This suggests that it may not be all that beneficial to increase the number of participating nodes beyond a certain value.

Figure 4.6 shows the ratio of the average number of links in the redundant path to that of default path as a function of percentage of participating nodes. For all three topologies, the ratio decreases slightly at first, then stays relatively constant. This phenomenon together with plots in Figure 4.5 indicate that major reduction in latency does not result from fewer links, rather from selecting links with shorter latencies. Figure 4.7 shows the cumulative distribution of shared links between redundant and default paths for the three topologies, with 10% of the nodes participating. As seen, the probability that the redundant and default path share few links is large for all three topologies. For example, the probability of two or fewer shared links for *Flat Albert-Barabasi*, *Albert-Barabasi II*, and *Albert-Barabasi I* is roughly 100%, 90%, and 85%, respectively. Note that the average number links of the default paths for *Flat Albert-Barabasi*, *Albert-Barabasi II*, and *Albert-Barabasi I* are 6, 11, and 12, respectively. This indicates that a redundant path with few shared links can be found with high probability, and hence PDS can be deployed effectively.

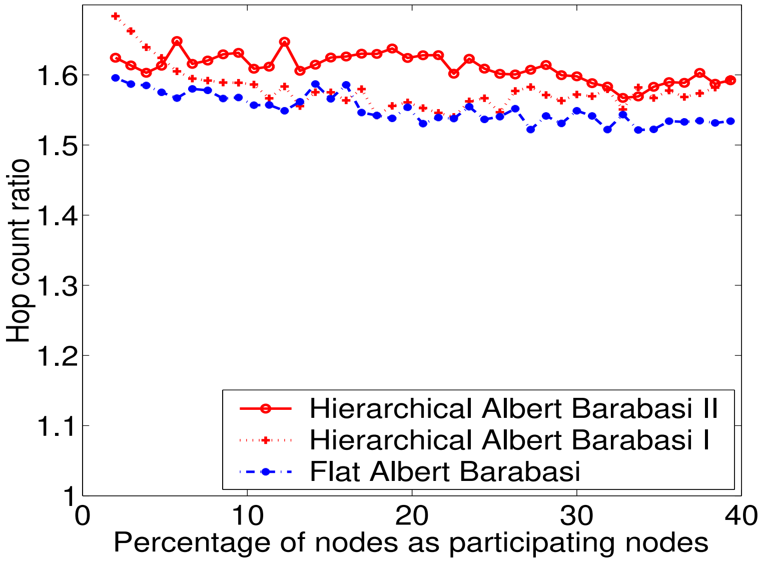


Figure 4.6: *Number of hops of the redundant path over the number of hops of the default path.*

4.4.2 System Performance using Forward Error Correction

In this section, we characterize the packet loss reduction for various number of shared links between redundant and default paths using NS [46]. The simulation topology for the redundant and default paths is shown in Figure 4.8. Based on the average number of links between two participating nodes in Section 4.4.1, the number of links for default and redundant paths are set to 11 and 17 respectively. Each link's capacity is 2Mbs with propagation delay of 4 *milliseconds*. To simulate bursty packet loss of the Internet, random exponential traffic is generated at each link with the peak rate of 1.8Mbs, average idle period of 8 seconds, and the burst period of 40 *milliseconds*. We compare the packet loss rate for following three scenarios: (1) sender streams the video to the receiver at 800kbps on the default path, (2) sender

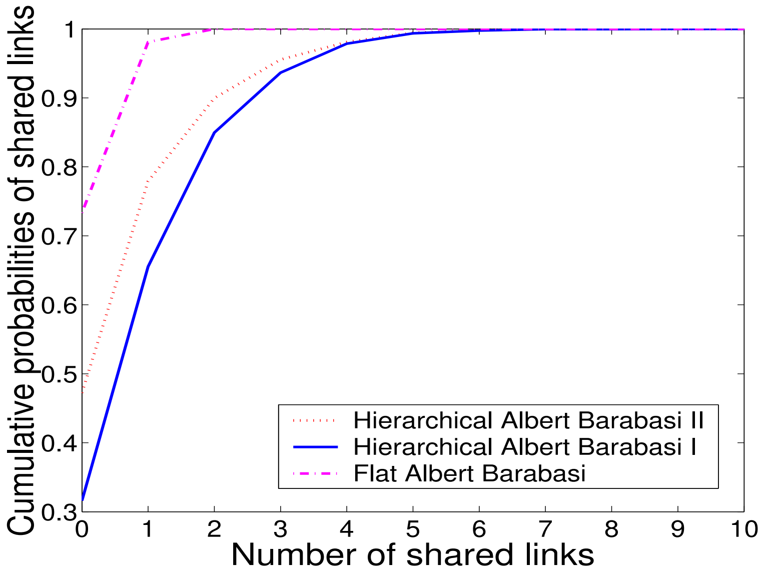


Figure 4.7: *Cumulative distribution of shared links for various network topologies.*

streams the video to the receiver on both redundant and default paths at 400kbps for each path with two paths are assumed to be completely disjoint, and (3) same as scenario 2 except there is one shared link between redundant and default paths.

In all scenarios, the video packet size is set to 500 bytes, and packets are protected using Reed-Solomon code RS(30, 23) with 23 data packets and 7 redundant packets for each FEC block. Hence, if there are more than 7 lost packets per a FEC block, then it is not possible to recover all the lost packets. As previously shown in Chapter 3, path diversity streaming framework is more effective with stronger FEC codes. However, stronger FEC results in higher overhead bandwidth and delay. In our simulations, we choose RS(30,23) for reasonable delay and bandwidth overhead introduced by FEC. Other FEC codes can also be chosen to satisfy the application's needs and

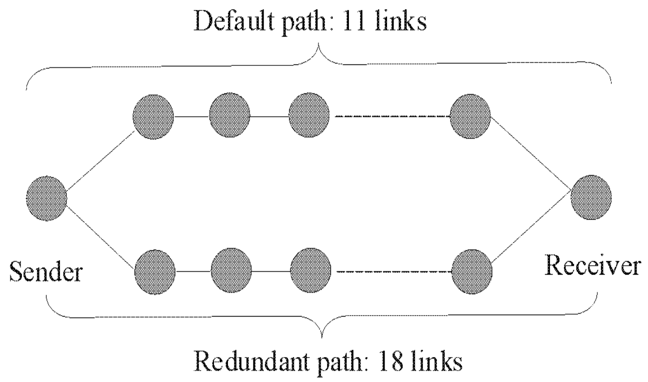


Figure 4.8: *Simulation configuration for two disjoint redundant and the default paths.*

performance.

Figures 4.9, 4.10, and 4.11 show the number of lost packets per 30 packets versus the packet sequence number for scenarios 1, 2, and 3, respectively. The points above the horizontal line represent irrecoverable loss events. As seen, there are considerably more irrecoverable loss events for scenario 1 than for 2. This is intuitively plausible since sending packets at a lower rate on multiple independent paths transforms the bursty loss behavior of a single path into a uniform loss behavior, thus reducing the burstiness, and increasing the recoverable probability. Also, the number of irrecoverable loss events for scenario 3 is larger than that of 2. This is due to the one shared link between the redundant and default paths in scenario 3. Assuming that links are independently congested, the larger number of shared links between the redundant and default paths leads to higher chance of simultaneous bursty loss on both paths, for which FEC is ineffective. One can think of scenario 1 where all packets are sent on only default path as an extreme case of path diversity in which all the links of two paths are shared.

Clearly, the recoverable probability of FEC decreases as the number of shared links between the redundant and default paths increases. To characterize the effect

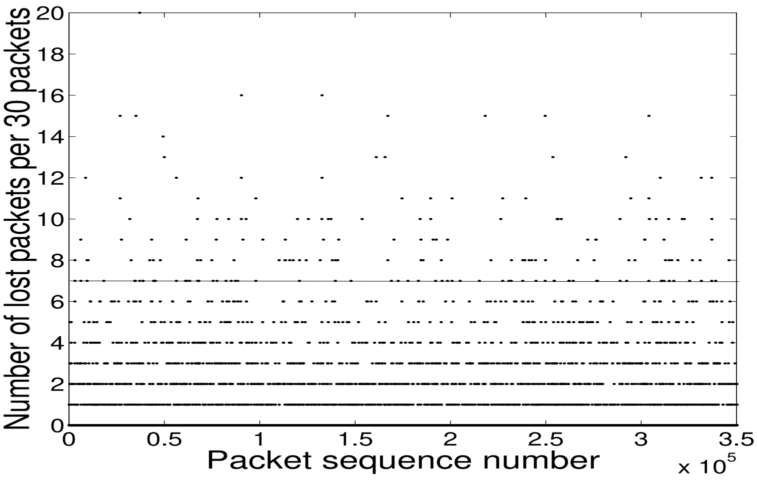


Figure 4.9: *Sending packets using traditional default path.*

of number of shared links on the loss rate, Figure 4.12 shows the ratio of the effective packet loss rate of uni-path scheme to that of dual path scheme as a function of number of shared links between them. The effective packet loss rate is the ratio between the number of irrecoverable lost packets and the total sent packets. As seen, the effective loss rate for the single path scheme is more than 7 times that of the path diversity scheme with completely disjoint redundant and default paths. The reduction in effective loss rate from using path diversity decreases as the number of shared links between the two paths increases. However, even when 3 of out 11 links are shared, the effective loss rate using path diversity scheme is twice smaller than

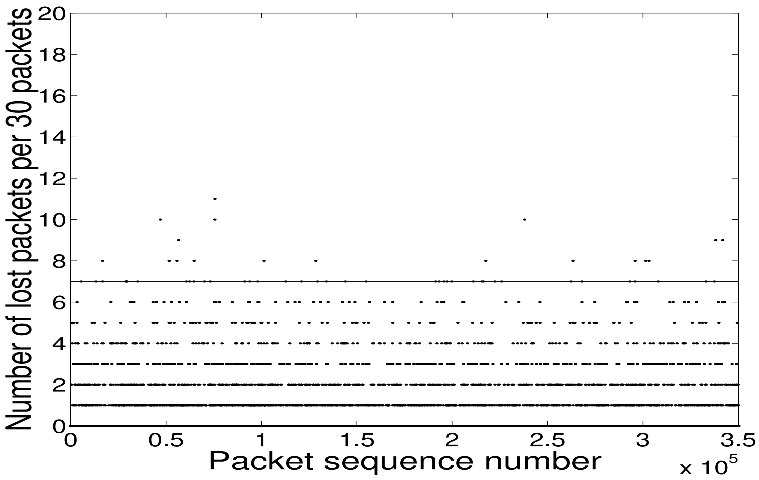


Figure 4.10: *Sending packets using both redundant and default paths without shared link between them.*

that of using uni-path scheme.

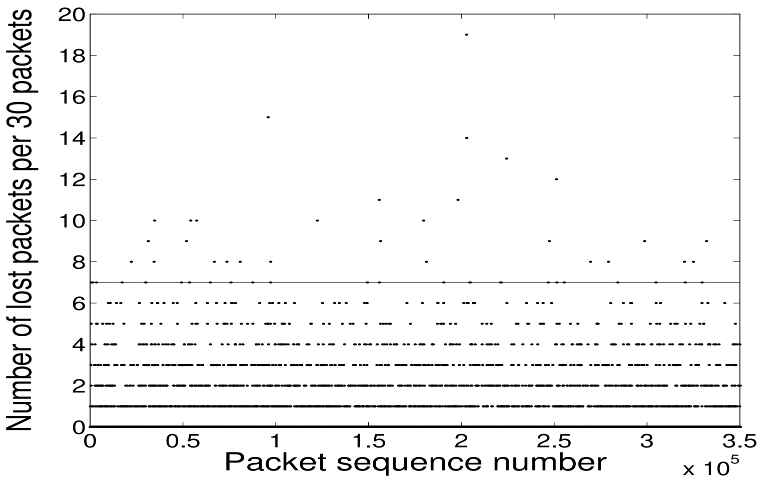


Figure 4.11: *Sending packets using both redundant and default paths with one shared link between them.*

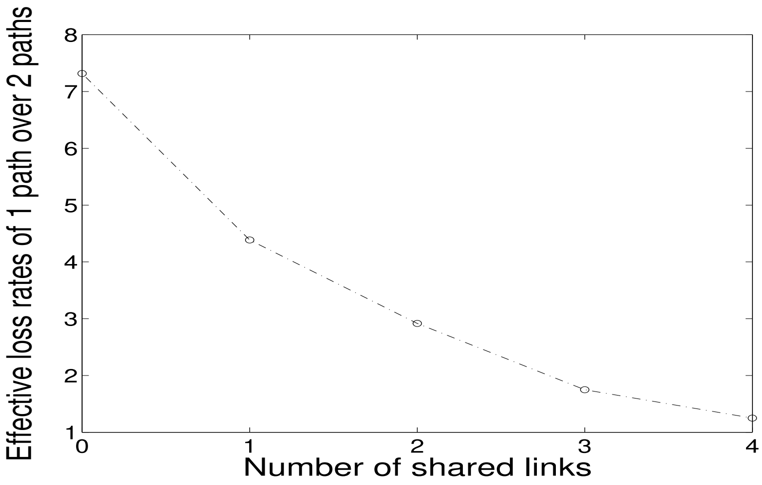


Figure 4.12: *The ratio of average loss rates using one path over that of using both redundant and default paths with various number of shared links between them.*

4.4.3 System Performance using Multiple Description Coding

To evaluate the performance gain of the PDS coupled with MDC, we run simulation to compare the average Peak Signal to Noise Ratio (PSNR) of our proposed scheme versus the traditional uni-path streaming using a single description video codec for various path characteristics, i.e number of shared links. For both single and multiple description codecs, we use matching pursuits based hybrid motion compensated technique as described in [63][64][65]. The main reason for doing so is convenience, and easy access to these codecs. The higher PSNR is assumed to correspond to higher video quality. Based on the results from Section 4.4, for the traditional uni-path single description case, packets are sent on the path consisting of 11 links. For the multi-path, multiple description case, packets are simultaneously sent on both the default and the redundant paths consisting of 11 and 17 links, respectively. For simplicity, packet loss on each link is assumed to be independent with equal loss rate. Using the experimental results reported from [66], we set the average duration of the burst loss to 120 milliseconds. In all experiments, we use standard test sequences Foreman, News, and Coast. For fair comparison, these video sequences are coded at 30 frames per second, and in such a way as to result in same bit rate for single and multiple description streams. The number of frames in a group of pictures (GOP) is set to 60 [1]. For error concealment in SDC, we use the last received frame to predict

[1]The common GOP sizes in video conferencing applications are from 60 to 300 frames

the lost frame.

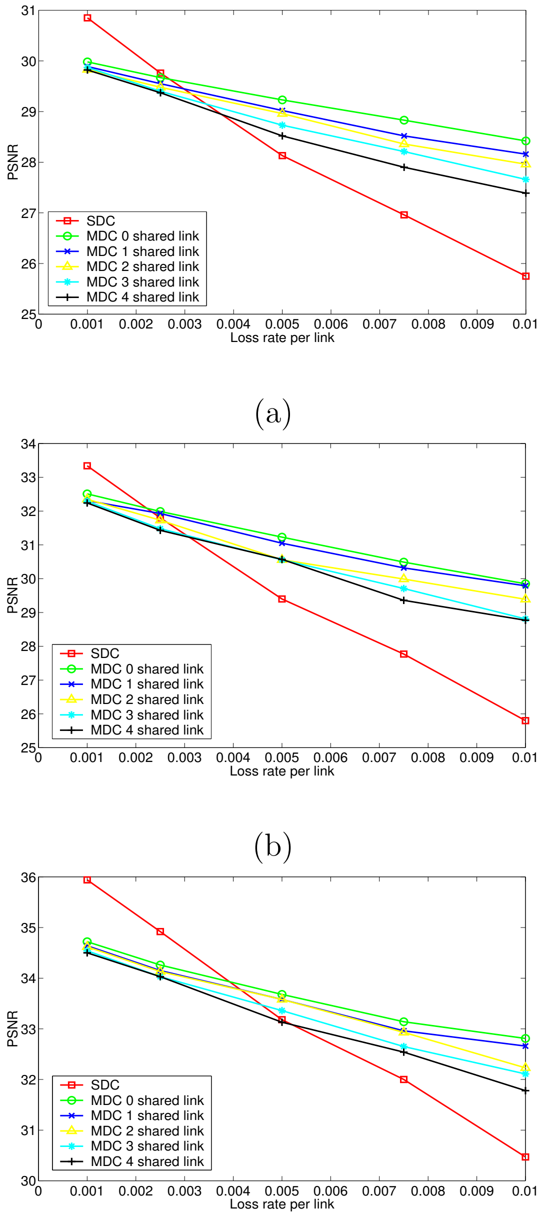
Note that another way to compare MDC with SDC is to reduce the source bit rate of SDC in order to use this additional bandwidth for FEC. However, we design these experiments to simulate interactive applications in situations where network burst loss is long, i.e. 120 milliseconds. In these scenarios, delay constraints of interactive applications require FEC codes to be short. However, short FEC codes are unable to correct long burst loss. For a reasonable performance in many scenarios, the length of FEC block is required to be at least two times the length of the burst loss. In our experiments, we set the burst loss to 120 milliseconds, and therefore the delay induced by appropriate length FEC code is 240 milliseconds, exceeding the tolerable one-way delay limit of 75 milliseconds for many interactive applications. On the other hand, in situations where the network characteristics exhibit short burst loss or where application can tolerate some delay, FEC can potentially outperform MDC.

For each sequence and loss rate per link, we send packets repeatedly for over an hour and compute the average PSNR. Figures 4.13 (a), (b), and (c) show the average PSNRs for sequence Coast, Foreman, and News as a function of loss rate per link for various number of shared links between redundant and default paths, respectively. As seen, when loss rate per link is less than 0.001, SDC results in higher average PSNR than MDC. However, when loss rate per link is greater than 0.005, MDC outperforms SDC. Also, as the number of shared links increases, the correlated packet loss increases, hence the average PSNR of MDC goes down. As shown in

Figure 4.7, a redundant path with 4 or fewer shared links can be found 98% of the time; therefore, MDC scheme via PDS is easily deployable in practice.

It's interesting to note that for the sequence News, there must be high loss rate per link, i.e. above 0.005, for MDC to outperform SDC. For Foreman sequence, MDC already outperforms SDC at low loss rate per link, i.e. below 0.001. Our hypothesis is that for low motion sequence such as News, the motion vectors between the previous and current frames are approximately identical, hence, the mismatch error caused by inaccurate motion vectors resulting from lost packets is small for SDC. For high motion sequences, the motion vectors are likely to be different from frame to frame, resulting in a high mismatch error for SDC when packet loss occurs.

It is interesting to determine whether it is beneficial to use SDC coupled with PDS, specifically, sending odd and even frames of SDC sequence on default and redundant paths. Figure 4.14 shows the PSNR as a function of loss rate for using SDC with uni-path and PDS with no shared links for Foreman sequence. A simple concealment method using the previous received block to replace the missing block is employed in the SDC case. As seen, PSNR of SDC with uni-path is consistently higher than that of SDC with PDS. The intuition for this observation is as follows. In the experimental setup, the redundant path has higher loss rate than the default path, i.e. redundant path has 17 links as compared to 11 links for default path. A single packet loss is likely to produce error propagation through the entire GOP for SDC. As a result, using SDC with PDS is likely to incur higher distortion due to higher likelihood of



(a)

(b)

(c)

Figure 4.13: *PSNR versus the loss rate per link for MDC on 2 paths and SDC on single path (a) Coast, (b) Foreman, and (c) News.*

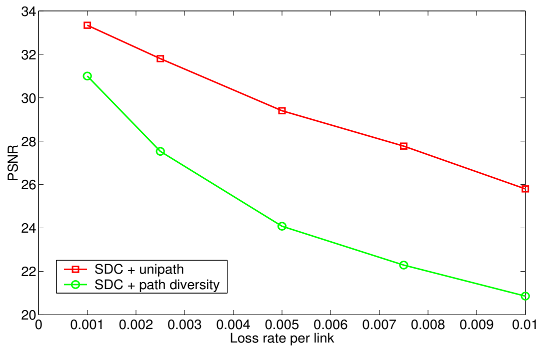


Figure 4.14: *PSNR versus the loss rate for SDC with uni-path and multi-path.*

encountering packet loss than that of using SDC with uni-path.

4.4.4 Internet Experiments

In this section, we show the results of two actual Internet experiments using MDC together with PDS. The setup for experiment one is shown in Figure 4.15 with the source at U.C. Berkeley, the destination at an AT&T cable modem subscriber in Berkeley, and the relay node for the redundant path at Stanford University. It is interesting to note that the average delay propagation of the redundant path via Stanford University is 16 milliseconds, 7 milliseconds shorter than that of the default path. From *traceroute*, we note that most of the time, packets are in *Level3* network for default path. In our experiment, we send MDC Foreman sequence repeatedly at 185 kbps and 30 fps with one video frame per packet. The the size of typical P-frames of each description ranges from 400 bytes to 600 bytes. Figure **??**(a) shows the frame delay as a function of time for two descriptions on two paths. Since the upper limit of two-way delay for many interactive applications is 150 milliseconds, we assume that packets with the one-way delay of more than 75 milliseconds are considered lost, i.e. points above the horizontal line in Figure 4.16(a) are considered lost frames. For the duration of 200 seconds of the experiment, we observe that no packets are dropped by the network and delays are relatively uncorrelated between default and redundant paths. However, as seen in Figure 4.16(a), there are clearly three peaks of unusually high delay or outage. Two of these outages occur on the default path while one occurs on the redundant path. These high packet delays result in almost no degradation in the corresponding PSNR of MDC stream as shown Figure 4.16(b). However, if the same packet trace are used to simulate PSNR for streaming SDC on either default or redundant path, the corresponding PSNRs drop significantly at these outages as shown on Figure 4.16(b). Note that retransmission scheme would not work in this experiment, since the outage can be on the order of seconds as shown in Figure 4.16(c), a zoom-in portion of outage from the redundant path in Figure 4.16(b). We also note that when there is no outage, the average PSNR of SDC is higher than that of MDC. However, in our experiment, the resulting MDC video is much more pleasant to view since there are neither pauses nor significant drops in quality from one frame to the next. For the resulting SDC video, we experience a two second pause, followed by highly distorted frames.

We perform another experiment to further validate the results of our approach. The setup for experiment two is shown in Figure 4.17. As shown in Figure 4.17, the default and redundant paths are quite disjoint, and the corresponding delays are 46 and 50 milliseconds, respectively. In this experiment, we use MDC to send Foreman sequence repeatedly at 300 kbps, 30 fps, and one video frame per packet. The the size of typical P-frames of each description ranges from 600 bytes to 900 bytes. Figure 4.18(a) shows the frame delay as a function of time for two descriptions on two paths. Since the delay of the actual lost frames, i.e. frames dropped by the networks, is infinite, for convenience, we set them to zero in the plot in of showing these lost frames in the plot in Figure 4.18(a). As seen, the default and redundant paths both exhibit packet loss and high delay; however, they are generally independent since the packets of the two paths spend most of their times, traveling through different networks, i.e. QWest and Cogento. Figure 4.18(b) shows the corresponding PSNRs for MDC and SDC as a function of time. Similar to experiment one, sending SDC on a single path results in significant PSNR reduction at times of packet loss or high delays, while using MDC coupled with PDS results in only a slight PSNR reduction. For visual comparison, Figures 4.19(a) and 4.19(b) show the typical resulting images of SDC and MDC coupled with path diversity at a loss event, repsectively for the sequence News.

4.5 Conclusions

In this chapter, we have proposed a *path diversity* system to be used with MDC and FEC techniques for delay sensitive applications over packet switched networks. In the

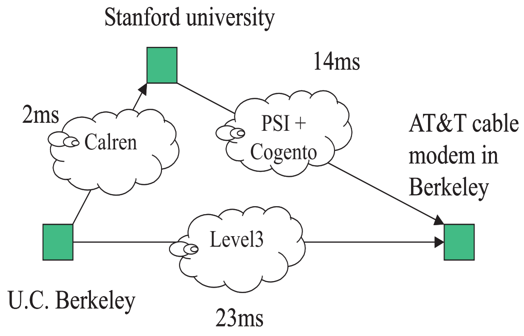
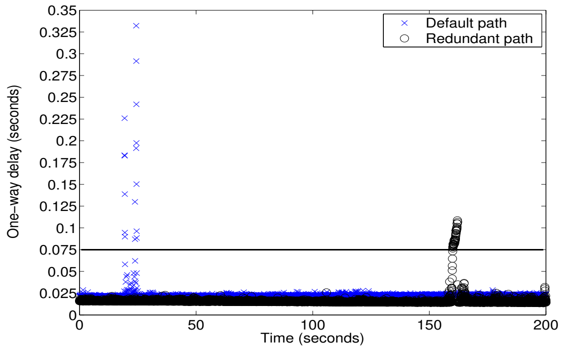
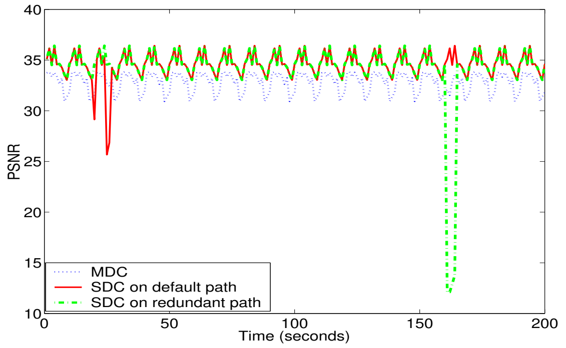


Figure 4.15: *Internet configuration for experiment one.*

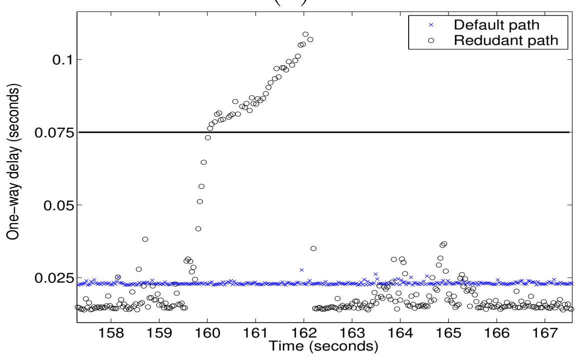
proposed PDS, the disjoint paths between a sender and a receiver are established using a collection of relay nodes. A scalable, heuristic scheme for selecting a redundant path has been proposed, and the resulting redundant path's lengths and disjointness for various Internet-like topologies have been characterized. Our simulations demonstrate that, for various Internet-like topologies, only 10% of participating nodes are required for the proposed redundant path selection scheme to effectively find a redundant path sharing two or fewer links with the default path, and hence, the proposed PDS can be realized in practice. Using simulations and actual Internet experiments, we demonstrate that coupling FEC or MDC techniques with the PDS result in superior visual quality over the traditional uni-path approach.



(a)



(b)



(c)

Figure 4.16: *Foreman sequence (a) Delay as a function of time, (b) PSNR as a function of time, and (c) Zoom in portion of the delay*

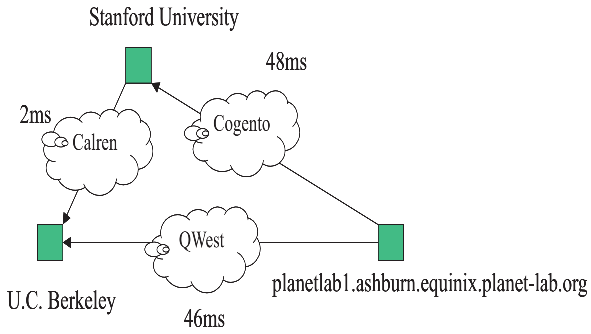
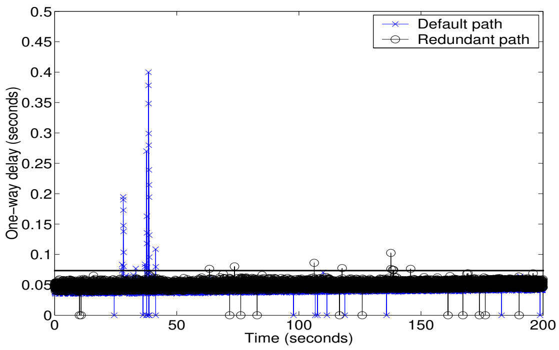
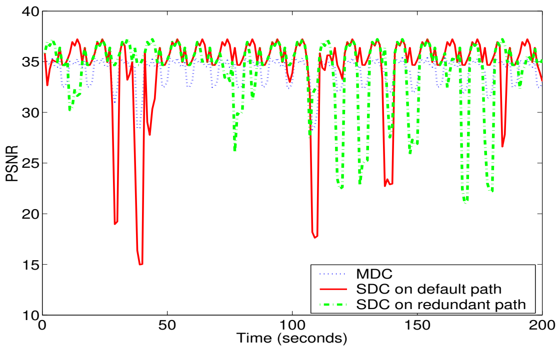


Figure 4.17: *Internet configuration for experiment two.*



(a)



(b)

Figure 4.18: *Foreman sequence (a) Delay as a function of time, with zero delay denoting actual packet loss; (b) PSNR as a function of time.*

(a)

(b)

Figure 4.19: *Typical images at a loss event for (a) SDC with uni-path streaming and (b) MDC with path diversity streaming.*

Chapter 5

Matching Pursuits Based Multiple Description Coding for Lossy Environments

5.1 Introduction

In Chapter 3, we propose path diversity media streaming over the Internet using Forward Error Correction. In many situations, e.g. short burst losses, using FEC together with path diversity is sufficient to achieve reasonable quality of the streamed media. However, FEC may not provide reasonable protection in other situations, e.g. long network outages. In these situations, Multiple description coding (MDC) is a better alternative for media streaming over the Internet. In this chapter, we propose a network adaptive matching pursuits based multiple description video coding for lossy environment. Multiple description coding is an error resilient source coding scheme that generates multiple encoded bitstreams of the source with the aim of providing an acceptable reconstruction quality of the source when only one description is received, and improved quality when multiple descriptions are available. This is different from layered coding [21] which requires the presence of the base layer for enhancement layer to be useful. As such, MDC is useful for delay limited applications in lossy environments such as wireless communication where outage can last several seconds or longer, rendering retransmission less useful. MDC can also be useful in emerging peer-to-peer (P2P) systems [67] on the Internet. In P2P applications such as Kazaa [67], each peer has the ability to limit the bandwidth for file transfer to others. As such, a single 400kbps connection from one receiving peer to another serving peer does not provide sufficient bandwidth for streaming a 700kbps video. This situation can be remedied by multiple serving peers stream the video simultaneously to the receiving peer. However, the frequent joins and leaves of peers can potentially disrupt the smooth video streaming on the order of several seconds to minutes. The solution to the dynamics of peers joining and leaving is to use multiple descriptions of video in which, each serving peer streams an independently decodable video description to the receiving peer.

Practical MDC schemes have been developed for video, and their performance over lossy channels have been characterized [22] [68][64]. Recently, an MDC technique based on matching pursuits (MP) signal decomposition, called MP-MDVC [49][64], has been shown to outperform Discrete Cosine Transform (DCT) based MDC [22]. In [68], the author applies the ROPE framework [69] to the DCT based MDC in [22] in order to optimize the video quality in lossy environments. In this chapter, we extend MP-MDVC [64] to optimize it for lossy environments. We choose MP-MDVC for the following reasons. First, MP-MDVC has been shown to outperform DCT based multiple description video coder[64]. Second, the rate distortion analysis for MP is simple and elegant as compared to orthogonal transforms where quantization is done after the transform is completed [70]. In particular, since the number of bits per atom[1] is more or less constant, rate control and rate prediction is straightforward in MP coding systems. The same is true for distortion prediction as adding an atom reduces distortion by the square of its magnitude. In contrast with MP, the rate and distortion in DCT based video coders are typically controlled by the quantization step size. As a result, for a given quantization step size, the rate and distortion are not known until the all DCT coefficients are coded for the entire frame. These factors result in a natural framework and ease of analysis for extending MP-MDVC to lossy environments, that is absent in DCT formulation.

The motivation for incorporating loss characteristics into multiple description coding technique is as follows. In high loss environments, most likely only one description at a time is received, hence it is desirable to achieve high reconstruction quality from

[1]an atom is a basis function together with the position information and magnitude

a single description. On the other hand, in low loss environments, two descriptions are likely to be received at a time, hence it is desirable to achieve as high of a reconstruction quality as possible from two descriptions. Since for a given total bit rate, there is an inherent trade-off between the reconstruction qualities of one and two descriptions, depending on the level of loss, one should strike a balance between the single and multiple description qualities. In this chapter, we formulate and solve this problem of how to achieve the above mentioned quality trade-off for matching pursuits based MDC. In particular, we propose a fast, steepest descent algorithm for computing the optimum number of atoms in each description based on the outage probabilities of the two channels so as to achieve minimum expected distortion, given bandwidth constraints, and maximum allowable distortion for each description. Our approach is based on a simple rate distortion analysis of MP, resulting in a fast practical implementation.

The rest of the chapter is organized as follows. In Section 5.2, we present overviews of MP and MP-MDVC. Next in Section 5.3, we describe our proposed optimization algorithm for lossy environments. In Section 5.4, we provide experimental results showing improved video quality over MP-MDVC[64]. Finally, we conclude in Section 5.5.

5.2 MP and MP-MDVC

5.2.1 Overview of MP

In many video coding standards, residual images are encoded using block-based DCT coding, thus introducing noticeable blocking artifacts at low bit rates. In matching pursuits video coding, motion compensated residual frames are decomposed into an overcomplete set of basis functions that is much larger than the complete basis set in DCT. Residual frame is coded using an iterative, greedy algorithm in which, at each iteration, the basis function with the largest inner product is subtracted from the residual frame. Since the magnitude of the inner product between the residue and the chosen function corresponds to distortion reduction at that iteration, this iterative, greedy method ensures that most important features are coded first. More details on matching pursuits video coding can be found in [63].

5.2.2 Overview of MP-MDVC

In this section, we briefly present the three loop structure originally proposed in [22] and subsequently used in MP-MDVC [64]. Figure 5.1 shows three loop MP-MDVC structure for generating two descriptions. In the central prediction loop, a new frame is first motion compensated from its prediction based on both descriptions, while in the side loops, the new frame is motion compensated based on only one description. This approach is employed in order for the decoder to track the encoder states when both descriptions are received, or when one of the descriptions is lost. Residue encoder applies MP decomposition to the motion compensated residue as described in Section 5.2.1, to result in a set of atoms. Let $F1(F2)$ denote the set of atoms generated by central loop for channel 1(2) as shown in Figure 5.1. In [64], the first L atoms found during MP decomposition in the central loop are shared by both sets $F1$ and $F2$, and subsequent atoms are alternately assigned into the two sets. Since atoms are found in decreasing order of magnitude, this results in the central loop atoms in $F1$ and $F2$ to be of approximately equal importance. Similarly, the sets $G1$ and $G2$ containing atoms from the side loops for channels 1 and 2 respectively, are found using MP decomposition. For convenience, we denote $F = F1 \cup F2$, $G = G1 \cup G2$, and the atom type by the name of its set, e.g. $F1$ atom. $F1$ and $G1$ atoms are sent on channel 1 while $F2$ and $G2$ atoms are sent on channel 2. Motion vectors, frame headers, and intra-coded (I) frames are duplicated and sent through both channels. Energy of residue $R1(R2)$ from motion compensation based on one description is first reduced by exploiting its correlation with coded residue from the central loop $F1(F2)$ through pixel-wise subtraction. Also, to save bits for residual coding, the same motion vectors are used in both side loops and central loop. More details on MP-MDVC can be found in [64].

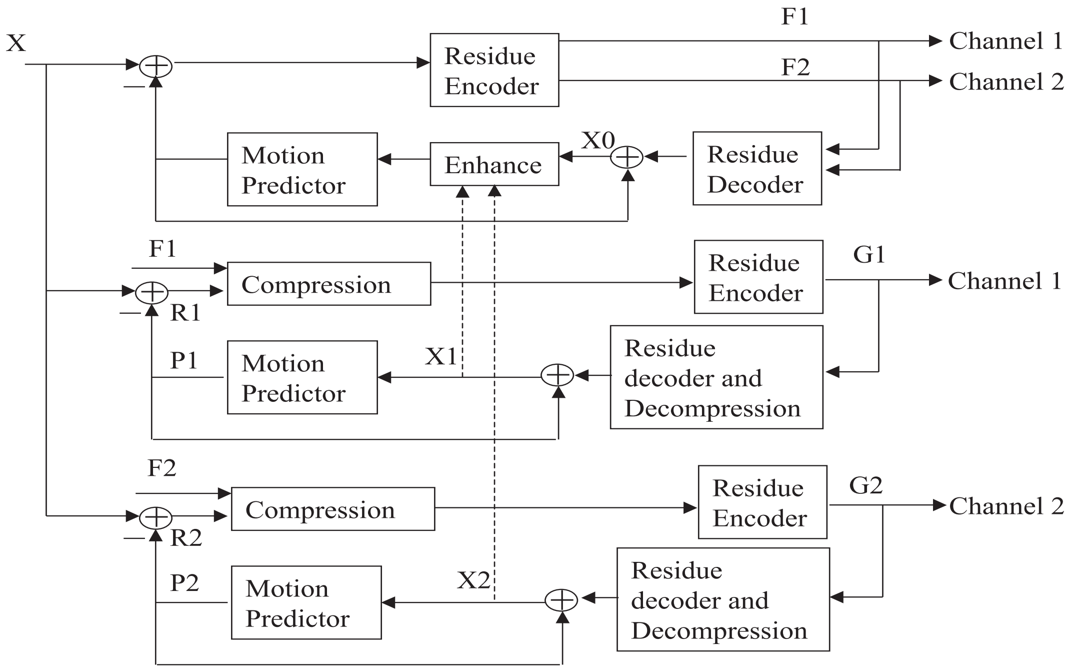


Figure 5.1: *Three loop structure of MP-MDVC.*

5.3 Fast algorithm MP-MDVC

In this section, we extend the MP-MDVC approach in [64] to optimally select the number of atoms in the central and side loops so as to match the outage probability of the channels.

5.3.1 Problem Formulation

Using the notation in Table 5.1, and assuming independent loss of descriptions, we wish to find a fast algorithm to compute f_1, g_1, f_2, and g_2 in order to minimize

$p_{1(2)}$	Probability of losing description 1(2)
D_0	Distortion induced by receiving both descriptions
$D_{1(2)}$	Distortion induced by receiving single description 1(2)
D'	Distortion induced by losing both descriptions
D^*	Maximum allowable distortion of any description
$R_{1(2)}$	Bits per frame of description 1(2)
$R^*_{1(2)}$	Maximum allowable bits per frame of description 1(2)
$f_1(f_2)$	Number of F atoms for description 1(2) per frame
$g_1(g_2)$	Number of G atoms in description 1(2) per frame

Table 5.1: *Optimization notation*

expected distortion, J, given by:

$$\begin{aligned} J &= (1-p_1)p_2 D_1(f_1, g_1) + (1-p_2)p_1 D_2(f_2, g_2) \\ &+ (1-p_1)(1-p_2)D_0(f_1, f_2, g_1, g_2) + p_1 p_2 D' \end{aligned} \tag{5.1}$$

subject to following constraints:

$$\max(D_1(f_1, g_1), D_2(f_2, g_2)) \leq D^* \tag{5.2}$$

$$R_1(f_1, g_1) \leq R_1^*$$

$$R_2(f_2, g_2) \leq R_2^*$$

In solving this problem, we are inherently assuming that D^*, R_1^*, and R_2^* are chosen in such a way that the set of feasible solutions is non-empty. The last term in Equation (5.1) can be ignored since it does not contain free variables. This is a general non-linear convex optimization with non-linear constraints. However, we can simplify the problem to arrive at a practical solution by simplifying the constraints. First, we remove the distortion constraint in (5.2) and add a penalty term P to J and call this sum J' to get

$$J' = P + J \tag{5.3}$$

where

$$P = (|D^* - \max(D_1, D_2)| + \max(D_1, D_2) - D^*)^2 \tag{5.4}$$

The rationale for this is that if the distortion constraint in (5.2) is not satisfied, namely, $\max(D_1, D_2) > D^*$ then $P = 4(\max(D_1, D_2) - D^*)^2$ is four times the square of the distortion difference. This is a large value resulting in large J', thus making the corresponding tuple (f_1, g_1, f_2, g_2) an unlikely solution. On the other hand, if the distortion constraint is satisfied, then $P = 0$, resulting in the minimum values of J' and J to be identical.

Second, in matching pursuits implementation [63], the number of bits used to code an atom is approximately constant, making the number of bits per frame directly proportional to number of atoms per frame. In addition, since higher bit rate results in lower distortion, any algorithm that minimizes the distortion should use all the available bandwidth. Based on these observations, the rate constraints for description i in (5.2), can be rewritten as $f_i + g_i = \frac{R_i^*}{B}$ where B is number of bits per atom.

As a result, we simplify our optimization problem into the problem of finding f_1, g_1, f_2, and g_2 in order to minimize

$$\begin{aligned} J' &= (1-p_1)p_2 D_1(f_1, g_1) + (1-p_2)p_1 D_2(f_2, g_2) \\ &+ (1-p_1)(1-p_2) D_0(f_1, f_2, g_1, g_2) + P \end{aligned} \tag{5.5}$$

subject to following constraints:

$$f_1 + g_1 = \frac{R_1^*}{B} \quad \text{and} \quad f_2 + g_2 = \frac{R_2^*}{B} \tag{5.6}$$

5.3.2 Solution to Optimization Problem

This problem can be solved fast using steepest descent method. Before describing the algorithm in details, we first provide several observations on which our algorithm is based.

Observation one:

MP decomposition codes atoms in decreasing order of magnitude, and the distortion reduction due to an atom is roughly square of its magnitude.

Observation two:

Using the three-loop structure in Figure 5.1, there is an inherent PSNR trade-off between the frames in the central and side loops for a given total bit rate. As the number of F atoms in the central loop increases, the number of G atoms in the side loop has to decrease in order to satisfy the constraints in Equation (5.6). Arguably, the side loop could potentially benefit from central loop atoms, since central loop atoms are used to predict side loop residues. However, these central loop atoms, especially the small magnitude ones, do not describe the side loop frame as well as the side loop atoms do. The reason is that the central loop atoms result from coding the frame predicted from the central loop which can be different from the frames predicted by the side loop. Hence, it is more efficient to use the same number of bits for side loop atoms to represent the side frame than central loop ones.

Observation three:

Suppose a frame I_0 is obtained from MP coding the original frame I_{orig} using N atoms, frame $I_{partial}$ is the resulting frame from randomly removing $M \leq N$ atoms from I_0, and frame I_{diff} is obtained by pixel-wise subtraction of $I_{partial}$ from I_{orig}. When I_{diff} is MP coded using $K \leq M$ atoms, the distortion reduction by these K atoms is roughly the sum of squares of magnitudes of K largest atoms belonging to the set of M removed atoms. Intuitively, the reason is as follows: I_{diff} contains the high energy regions which are used to be represented by the removed atoms. Since MP decomposition finds the regions with largest energies to code first, it is likely to find and code the largest atoms that have been previously removed in frame I_0.

We now propose a fast algorithm to solve the optimization problem in Equation (5.5) based on the above observations.

Step 1:

Assign the number of central loop atoms to the maximum allowable by bit rate constraint, namely, $|F| \leftarrow \lfloor \frac{R_1^*}{B} \rfloor + \lfloor \frac{R_2^*}{B} \rfloor$.

Step 2:

Code the frame in the central loop using F atoms, and record their corresponding magnitudes. Based on observation one, these magnitudes are used to compute the distortion in later steps.

Step 3:

Assign the central loop atoms alternately in the order of decreasing magnitude into two sets of atoms $F1$ and $F2$ until either $f_1 > \lfloor \frac{R_1^*}{B} \rfloor$ or $f_2 > \lfloor \frac{R_2^*}{B} \rfloor$; then assign the remaining atoms into the other unfilled set. We set L, the number of shared atoms between the two descriptions in central loop in [64] to 0. The reasoning is based on observation three: When coding the side frame, the resulting G atoms from the side loops are likely to more or less correspond to the atoms assigned to the other description; hence the two descriptions are likely to share similar energy atoms resulting from side loops.

Step 4: *Steepest Descent steps:*

From observation two on PSNR trade-off, for a fixed bit rate, increasing number of G atoms requires decreasing number of F atoms, and therefore, increases(decreases) the PSNR of frames in the side (central) loop. Thus, given a total number of atoms, the problem is optimum allocation of atoms between F and G sets, i.e. central and side loops. Furthermore, since the total bits per description remains fixed at $\lfloor \frac{R_i^*}{B} \rfloor$, we only allow allocation of atoms within a description, e.g. adding a side loop atom from a given description requires removing a central loop atom from the same description. Since our algorithm begins with the maximum number of central loop atoms, there are three options to descent to the minimum value: (a) decrease f_1 and increase g_1, (b) decrease f_2 and increase g_2, and (c) stop, the allocation is optimal. Starting from the smallest energy central loop atoms found in step 1, we choose one of the above options so as to lower J'. We repeat the process for the next central loop atoms in the order of increasing energy until removing a central loop atom and adding a side loop atom no longer results in a smaller value of J'.

5.3.3 Practical Implementation

Even though, in each step of the steepest descent algorithm, we need to compute J', we have not shown how to compute it efficiently. A simple method would be to use the three-loop structure in Figure 5.1 to directly code three separate frames, one for the central loop and two for the side loops, and compute the resulting distortions and J' in each step of the steepest descent algorithm. However, this simple method is computationally expensive since the two side loop frames have to be coded again every time a side loop atom is added in step 4 of the algorithm described in Section 5.3.2.

To reduce computational complexity, we propose a fast method to approximate the distortion of the side loop frames. Rather than coding the entire side loop frame each time a side loop atom is added in order to compute the resulting distortion, we approximate the distortion reduction by using the atom's magnitude. The key to this approximation is based on observation three. Recall that the central loop atoms are alternately assigned to each description; hence, each description is missing the central loop atoms from the other description. When coding the side loop frame for each description, the MP decomposition is likely to choose the K side loop atoms to be similar to the largest K missing central loop atoms for that description. Since the magnitude of central loop atoms are already computed and recorded in step 2, there is no need to re-code the side loop frame to compute the resulting distortion. Instead, the magnitude of these missing central loop atoms are used to approximate the distortion for each side loop frame. As an example, suppose the central loop initially codes with 8 central loop atoms, a_i, $i = 1, ...8$ with corresponding magnitudes $|a_i|$: 256, 255, 128, 127, 64, 63, 32, 31, 16, and 15. $F1$ contains a_1, a_3, a_5, a_7 while and $F2$ contains a_2, a_4, a_6, a_8. The energy in each frame is sum of squares of all the central loop atoms in that frame. By removing the least energy central atom a_8 in the central loop frame and adding one more side loop atom to the side loop 2, the distortion in the central loop frame increases by 15*15, i.e. squared magnitude of a_8, however, the distortion in the side loop frame 2 is predicted to be reduced by 256*256, squared magnitude of a_1. This is because the energy of the new side loop atom is predicted to equal to the largest missing central loop atom in side loop frame 2, namely, a_1. Similarly, by removing the next least energy atom, a_7, the distortion in the central loop increases by 16*16, however, the distortion in the side loop frame 1 is predicted to be reduced by 255*255. Depending on the loss probabilities p_1, p_2, and the penalty term P in Equation (5.5), the algorithm decrements the number of central loop atoms for each description and incrementing the number of side loop atoms for the same description until J' stops decreasing. In doing so, there is no need to code the side loop frame again every time an atom is added. Rather, after the algorithm terminates, the returned number of central and side loop atoms for each description is used to code each of the side loop frame once. Also, since J' is non-increasing and there are at most $|F|$ atoms, the steepest descent converges in at most $|F|$ steps.

This approximation assumes that the reference frames in the three loops are identical. In practice, these reference frames are not identical except for the first I-frame in a group of pictures (GOP). However, they are sufficiently similar to make this approximation works well in practice as shown in Section 5.4.

5.4 Experimental Results

In this section, we use standard MPEG CIF sequences Foreman and Hall coded at 10fps with GOP size of 30 for our experiments. The total number of atoms for each description is 100, making the approximate bit rate per description for Foreman and Hall to be 140 and 52kbps, respectively. The outage probability for both descriptions is set to 0.1. We assume that during an outage, the entire description is lost. This is reasonable since outages in wireless and P2P environments are on the order of several seconds to minutes. The minimum PSNR constraints for Foreman and Hall are chosen typically by applications or users. In this experiment, we set reasonable minimum constraints for Hall and Foreman to 31 and 29dB, respectively.

We first show the PSNR results of proposed approximations in Section 5.3.3. Figure 5.2 shows the predicted PSNR using approximation in Section 5.3.3 and the actual PSNR by coding the side loop frame for Foreman and Hall sequences as a function of frame number. As seen, the predicted and actual PSNR for both sequences are very close. The algorithm predicts better for Hall than Foreman sequences. In general, the algorithm predicts better for low motion sequences. Also, prediction error seems to be slightly larger as the frame number increases. This error is possibly due to the larger difference of the reference frames in central and side loops for later frames in a GOP. Another observation is that for most frames, the actual PSNR is slightly larger than the predicted PSNR. This is intuitively plausible since side loop atoms are likely to represent the side loop frame better than central atoms, as discussed in observation two. Figure 5.3 shows the PSNRs of two description (PSNR0) and one description (PSNR1) for Foreman and Hall sequences. As seen, the PSNRs of either description is larger than the specified minimum constraints of 29dB and 31dB for Foreman and Hall, respectively, showing that the algorithm satisfies the distortion constraints.

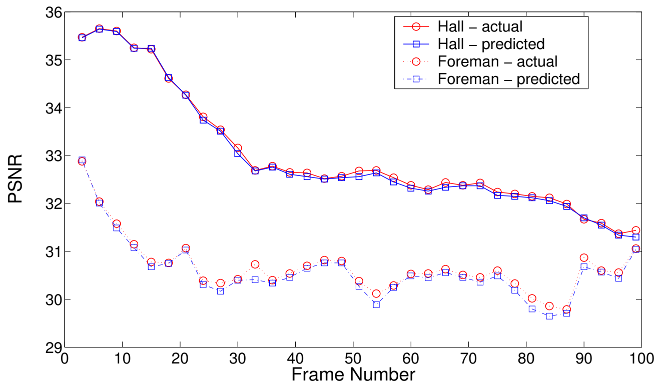


Figure 5.2: *Actual and predicted PSNR for Foreman and Hall.*

We now consider the how PSNR0 and PSNR1 vary as a function of outage prob-

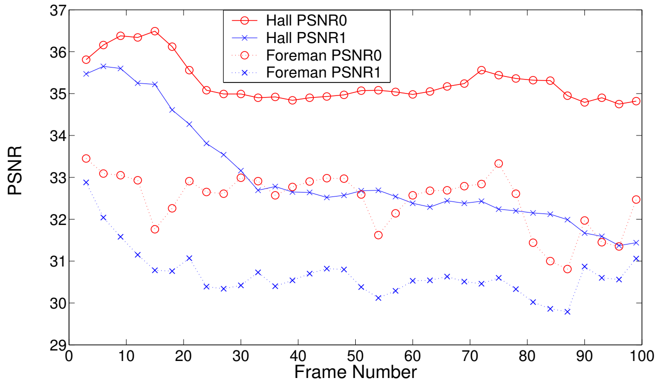


Figure 5.3: *PSNR0 and PSNR1 for Foreman and Hall.*

ability. For simplicity, we assume that outage probability of the two channels are identical. Intuitively, for small outage probability, one would expect PSNR0 to be large, and PSNR1 to be small since most of the time, the receiver receives two descriptions. On the other hand, when the outage probability is large, PSNR1 should be large since most of the time only one description is received. Figure 5.4 confirms this, showing PSNR0 decreases and PSNR1 increases with the outage probability. Figure 5.5 shows the corresponding average number of G atoms for each description as a function of outage probability. As expected, as outage probability increases, more G atoms are used, resulting in higher PSNR1. To compare the performance of our proposed approach MP-MDVC [64], Figures 5.6(a) and (b) show the expected PSNR in Equation (5.1) for standard sequences Hall, Foreman, News, and Mom as a function of outage probability. In MP-MDVC[64], $f_1 = g_1 = f_2 = g_2 = 50$ independent of network characteristics. As seen, at low outage probability, our method results

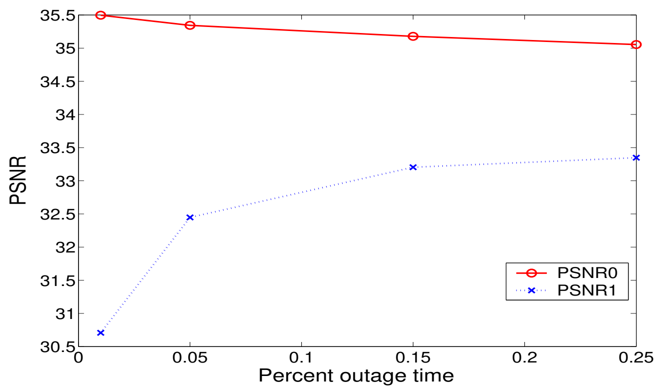


Figure 5.4: *PSNR0 and PSNR1 for Hall as a function of outage probability.*

in over 1dB improvement over MP-MDVC. As the outage probability increases, the expected PSNR gap decreases. This is due to the fact that the chosen number of G atoms in MP-MDVC method is large enough, leading to higher PSNR1, and hence resulting in higher expected PSNR for high loss environments.

5.5 Conclusions

In this chapter, we have proposed a fast, steepest descent algorithm for computing the optimum number of atoms in each description based on the outage probabilities of the two channels so as to achieve minimum expected distortion, given bandwidth constraints, and maximum allowable distortion for each description. Our approach is based on a simple rate distortion analysis of MP, resulting in a fast practical implementation. Analytical and experimental results show that by taking network loss characteristics into account, our algorithm results in improved performance over

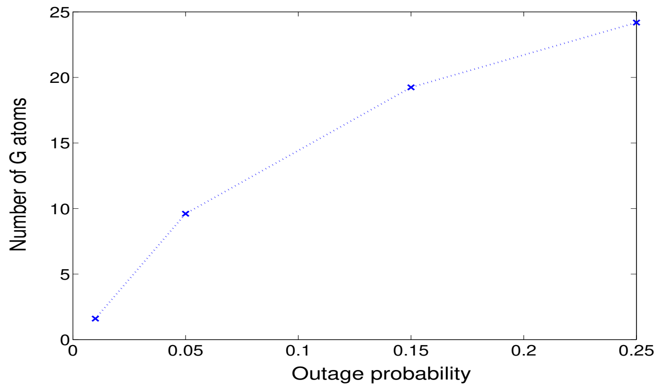
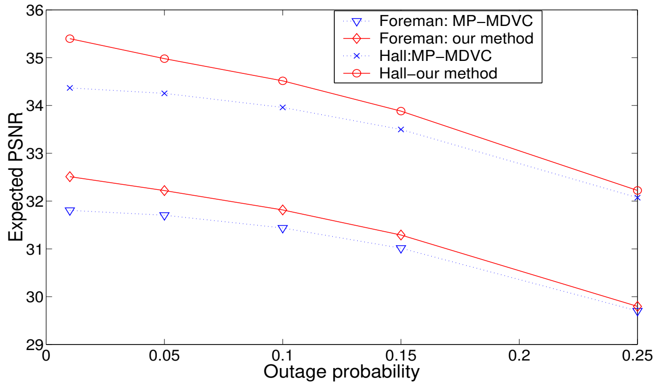
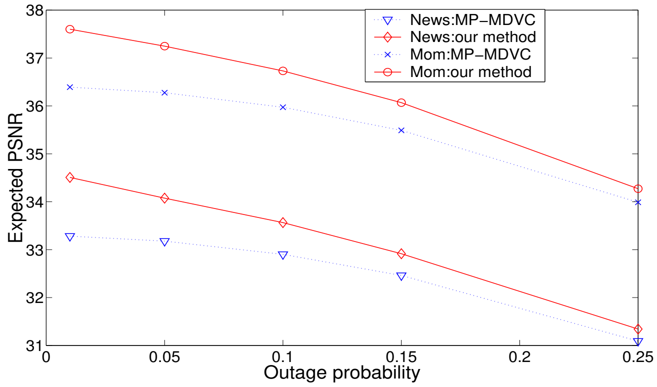


Figure 5.5: *Average number of G atoms for Hall as a function of outage probability.*

previously proposed MP-MDVC [64].



(a)



(b)

Figure 5.6: *Expected PSNR as a function of outage probability; (a) Foreman and Hall; (b) Mom and News.*

Chapter 6

Summary and Future Work

6.1 Summary

In this dissertation, we have developed a *path diversity* framework for concurrent media streaming to a receiver using multiple routes. Without requiring QoS, our framework attempts to improve the quality of the streamed media via multiple routes created using either multiple senders or relay nodes, in order to increase available bandwidth, reduce packet loss and delay. Our *path diversity* framework combines many approaches from the network architecture and protocols,to source and channel coding to improve the overall quality of the streamed media.

In Chapter 2, we present *path diversity* framework for multimedia streaming using multiple senders in order to achieve higher throughput, and to increase tolerance to loss and delay due to network congestion. In our framework, multiple senders simultaneously stream the media, e.g. video to a receiver via relatively disjoint paths. The advantage of this framework is that it combats unpredictability of congestion in the Internet. If the route between a particular sender and the receiver experiences congestion during streaming, the receiver can redistribute streaming rates among other senders, thus resulting in smooth video delivery. Another advantage is that, using multiple routes provides higher bandwidth than single route, hence, higher quality media can be streamed. Within this framework, we propose a receiver-driven protocol in which, the receiver coordinates simultaneous transmissions from multiple senders through the control packets sent to all senders from the receiver. The proposed protocol employs two main algorithms: the rate allocation and packet partition algorithms. The rate algorithm determines the sending rate on each route in order to minimize the packet loss and share bandwidth fairly with other traffic, while the packet partition algorithm ensures that no sender sends the same packets, and at the same times, minimizes startup delay. We have shown the feasibility of such a framework and the performance improvements of the proposed protocol for media streaming.

In Chapter 3, we have developed a novel rate allocation algorithm to incorporate FEC and bursty packet loss pattern on the Internet. We have shown theoretically and experimentally that FEC is more effective by streaming media simultaneously over multiple routes at appropriate sending rates. In addition, we have provided a robustness analysis for such a scheme, and suggested the optimal strategy for using FEC under various network conditions. Both simulations and Internet experiments indicates significant improvement using our rate allocation algorithm.

The path diversity system using multiple senders proposed in Chapter 2 cannot be used for live streaming or interactive applications. In Chapter 4, we have proposed a path diversity system that allows single sender to send packets simultaneously on both default Internet and redundant paths to the receiver. To create redundant path, the sender sends packets to the appropriate relay node, which then forwards the packets to receiver. The relay node selection algorithm is designed to ensure that packets traveling through the relay node take a different underlying physical path than that of the default Internet path between the sender and receiver. Our system shows a significant improvement in quality of the streamed media over the traditional uni-path streaming techniques.

In Chapter 5, we design a network adaptive matching pursuits based multiple description video coding scheme for our path diversity system. Multiple description coding is an error resilient source coding scheme that generates multiple encoded bit-streams of the source with the aim of providing an acceptable reconstruction quality of the source when only one description is received, and improved quality when multiple descriptions are available. Our network adaptive multiple description matching pursuits scheme is designed to produce unequal descriptions of the source based on network characteristics of each route, hence providing superior visual quality than traditional single description, uni-path approach.

6.2 Recommendations for Future Work

In chapter 3, we have not addressed the issue of changing FEC level during the session. This may be necessary when channel condition becomes worse. One research question is *what is the amount of FEC required to achieve certain level of quality of service, e.g. packet loss rate?* Also, in chapters 2 and 3, we only consider the possibility of changing the sending rates among a fixed set of senders to achieve the required bandwidth for the video. If the required video bitrate is still larger than the total bandwidth provided by all the current senders, then there are two promising approaches to alleviate the problems. The first approach is to use scalable video codec in order to reduce the video bitrate to the available bandwidth at the expense of graceful degradation of video quality. In the second approach, the receiver can dynamically request new senders in order to provide additional bandwidth as required by the video. There are research issues associated with these two approaches.

6.2.1 Scalable Video

In the traditional uni-path streaming using scalable video bitstream, the most important packets, e.g. packets contain bits from the based layer, are sent while the least important packets, e.g. packets contain bits from the enhanced layers are dropped under insufficient bandwidth. In the multi-path streaming scenario, in addition to deciding which packets should be sent, one may also want to consider on which path a particular packet should be sent. It may be advantageous to send the "important" packets on the "good" path while the "less important" packets on "not so good" path. It may be also advantageous to send duplicate important packets such as the base layer packets, on both paths to achieve error resiliency. Based on these, the research question is *how to partition packets among the routes or senders in order to achieve the highest visual quality under certain network conditions and bandwidth constraints?*

6.2.2 Requesting Additional Servers

When a user needs to request a new server for additional bandwidth, possible criteria for assigning a new server to the receiver may include the loss, delay, and bandwidth characteristics of the path between the new server and the user. Using our passive approach, we do not know these metrics until data are actually sent on this path to receiver, i.e. we rely on the data packet themselves to estimate the packet loss rate and delay. In order to quickly assign the new sender to the receiver for additional bandwidth, one may use *active probing* to estimate the packet loss rates and delays of all the servers to the receiver a priori. *Active probing* uses a small fraction of bandwidth to monitor the network conditions constantly, hence, any point in time, the best server can be assigned to the receiver quickly. This simple active probing approach requires the server to monitor the paths to all receivers. Clearly, this approach is not scalable. A research question is *how to make "active probing" method scalable?* Given an overlay network of N nodes, there are a total of $N(N-1)$ possible paths between two nodes. However, not all these paths are completely independent since they may share the same underlying physical links. Based on these, one promising approach is to probe network characteristics of a small appropriate chosen fraction of paths, and using these to reliably infer the network characteristics of other paths.

6.2.3 Peer to Peer Streaming

A natural extension of the path diversity media streaming framework is the Peer-to-Peer (P2P) streaming framework. Similar to our proposed path diversity framework, in a P2P streaming scenario, multiple senders simultaneously stream the video to a receiver. In contrast to our scenario in which, senders are reliable servers, in P2P streaming scenarios, senders are often other participants, and therefore, are usually unreliable due to their frequent joins and leaves the P2P network. These frequent leaves of peers in the P2P network result in long outages which may render FEC techniques less useful. The MDC technique proposed in Chapter 5 can alleviate this problem. Another promising approach from network perspective is to build an efficient hybrid overlay network consisting of unreliable peers and reliable servers in order to reduce the number of complete outages. Within this framework, the reliable servers can stream the base layer of the scalable video bitstream to the receiver, while the peers are responsible for the enhanced layers. This approach ensures that the receiver can receive reasonable quality video stream most of the times, and higher quality video stream when more peers are available. Some of the research questions are: *How many and where to place the servers for a given P2P networks? What is the protocol for information exchange among peers and servers? How to reduce number of "free riders", i.e. peers that do not contribute resources?*

Bibliography

[1] P. Baran, *On Distributed Communications, RM-3420-PR*, RAND Corporation, http://www.rand.org/publications/RM/baran.list.html, 1964.

[2] J. Walrand, *Communication Networks*, WCB/McGraw-Hill, 1998.

[3] J. Postel C. Sunshine D. Cohen, "The arpa internet protocol," *Computer Networks*, vol. 5, pp. 267–271, July 1981.

[4] B. Leiner, V. Cerf, D. Clark, R. Kahan, L. Kleinrock, D. Lynch, J. Postel, L. Roberts, S. Wolf, C. Labovitz, G. Malan, and F. Jahanian, "A brief history of the internet," *Internet Society*, 2000.

[5] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Transactions on Networking*, August 1999.

[6] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM*, August 1988, pp. 314–329.

[7] J. Saltzer, D. Reed, and D. Clark, "End-to-end arguments in system design," *ACM Transaction on Computer System*, vol. 2, no. 4, pp. 277–288, November 1984.

[8] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast application," in *Architectures and Protocols for Computer Communication*, October 2000, pp. 43–56.

[9] R. Rejaie, M. Handley, and D. Estrin, "Rap: And end-to-end rate-based congestion control mechanism for real time streams in the internet," in *Proceedings of IEEE INFOCOM*, March 1999.

[10] D. Sisalem and H. Schulzrinne, "The loss-delay based adjustment algorithm: A tco-friendly adaptation scheme," in *NOSSDAV*, May 1998.

[11] D. Clark and D. Tennehouse, "Architecture considerations for new generation of protocols," in *ACM Sigcomm*, September 1990, pp. 200–208.

[12] C. Weinstein and J. Forgie, "Experience with speech communication in packet networks," *IEEE Journal, Selected area Communication*, vol. 1, no. 6, pp. 963–980, December 1983.

[13] P. White, "Rsvp and integrated services in the internet: A tutorial," *IEEE Communication Magazine*, pp. 100–106, May 1997.

[14] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "Rsvp: A new resource reservation protocol," *IEEE Network Magazine*, vol. 7, no. 5, pp. 8–18, September 1993.

[15] S. Blake, D. Black, M. Carson, E. Davis, Z. Wang, and W. Weiss, "An architecture for differentiated services," in *RFC2475*, December 1998.

[16] D. Hoffman, G. Fernando, and V. Goyal, "Rtp payload format for mpeg1/mpeg2 video," in *RFC2250*, January 1998.

[17] J. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive fec-based error control for internet telephony," in *Proceedings of IEEE INFOCOM*, 1999, vol. 3, pp. 1453–60.

[18] U. Horn, K. Stuhlmuller, M. Link, and B. Girod, "Robust internet video transmission based on scalable coding and unequal error protection," *Signal Processing: Image communication*, vol. 15, pp. 77–94, 1999.

[19] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, "The pim architecture for wide-area multicast routing," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 153–162, April 1996.

[20] W. Tan and A. Zakhor, "Real-time internet video using error resilient scalable compression and tcp-friendly transport protocol," *IEEE Transactions on Multimedia*, vol. 1, pp. 172–186, june 1999.

[21] F. Wu, S. Li, and Y. Zhang, "A framework for efficient fine granularity scalable video coding," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 332–344, March 2001.

[22] A. Reibman, H. Jafarkhani, Y. Wang, M. Orchard, and R. Puri, "Multiple description coding for video using motion compensated prediction," in *Proceedings of International Conference on Image Processing (ICIP)*, October 1999, vol. 3, pp. 837–841.

[23] J. Apostolopoulos, "Reliable video communication over lossy packet networks using multiple state encoding and path diversity," in *Proceeding of The International Society for Optical Engineering (SPIE)*, January 2001, vol. 4310, pp. 392–409.

[24] R. Puri, K. Ramchandran, K. Lee, and V. Bharghavan, "Forward error correction (fec) codes based multiple description coding for internet video streaming and multicast," *Signal Processing: Image Communication*, vol. 6, no. 8, pp. 745–762, May 2001.

[25] K. Goyal and J. Kovacevic, "Generalized multiple description coding with correlating transforms," *IEEE Transactions on Information Theory*, vol. 47, pp. 2199–2224, April 2001.

[26] Y. Wang, M. Orchard, V. Vaishampayan, and A. Reibman, "Multiple description coding using pairwise correlating transforms," *IEEE Transactions on Image Processing*, vol. 10, no. 3, pp. 351–366, March 2001.

[27] C. Labovitz, G. Malan, and F. Jahanian, "Internet routing instability," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 515–528, October 1998.

[28] V. Paxson, "End-to-end routing behavior in the internet," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 601–615, October 1997.

[29] H.V. Poor and G.W. Wornell, *Wireless Communications: Signal Processing Perspectives*, Prentice Hall, 1998.

[30] N.F. Maxemchuk, *Dispersity Routing in Store and Forward Networks*, Ph.D. thesis, University of Pennsylvania, 1975.

[31] M.O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *Journal of the Association for Computing Machinery*, vol. 36, no. 2, pp. 335–348, April 1989.

[32] J. Byers, M. Luby, and M. Mitzenmacher, "Accessing multiple mirror sites in parallel: using tornado codes to speed up downloads," in *Proceedings of Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, March 1999, vol. 1, pp. 275–283.

[33] E.G. Steinbach, Y.J. Liang, and B. Girod, "A simulation study of packet path diversity for tcp file transfer and media transport on the internet," in *Tyrrhenian International Workshop on Digital Communication (IWDC)*, September 2002.

[34] V.N. Padmanabhan, H.J. Wang, P.A. Chou, and K. Sripanidkulchai, "Distributed streaming media content using cooperative networking," in *ACM NOSSDAV*, Miami, FL, May 2002.

[35] R. Rejaie and A. Ortega, "Pals:peer to peer adaptive layered streaming," in *NOSSDAV*, June 2003.

[36] J. Apostolopoulos, "On multiple description streaming with content delivery networks," in *InfoComm*, June 2002, vol. 4310.

[37] A. Majumda, Rohit Puri, and K. Ramchandran, "Distributed video streaming from multiple servers," in *International Conference on Image Processing (ICIP)*, Rochester, NY, September 2002.

[38] J.Chakareski and B. Girod, "Rate-distortion optimized packet scheduling and routing for media streaming with path diversity," in *Data Compression Conference*, April 2003.

[39] Y.J. Liang, E.G. Steinbach, and B. Girod, "Real-time voice communication over the internet using packet path diversity," in *Proceedings ACM Multimedia 2001*, Sept 2001, pp. 431–440.

[40] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting shared congestion of flows via end-to-end measurement," in *International Conference on Measurement and Modeling of Computer Systems (ICMMCS)*, June 2000, pp. 145–155.

[41] M. Kalman, E.G. Steinbach, and B. Girod, "R-d optimized media streaming enhanced with adaptive media playout," in *International Conference on Multimedia and Expo (ICME)*, August 2002.

[42] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *Computer Communication Review*, vol. 27, no. 2, pp. 24–36, April 1997.

[43] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modelling of the temporal dependence in packet loss," in *INFOCOM*, 1999, vol. 1, pp. 345–52.

[44] B. Wah and D.Lin, "Transformation-based reconstruction for real-time voice transmissions over the internet," *IEEE Transactions on Multimedia*, vol. 1, pp. 342–351, December 1999.

[45] D.G. Andersen, *Resilient overlay networks, Master Thesis*, Massachusetts Institute of Technology, 2001.

[46] Information Sciences Institute, http://www.isi.edu/nsnam/ns, *Network simulator*.

[47] M. Jordan and C. Bishop, *An Introduction to Graphical Models (In press)*.

[48] Y.Rekhter and T.Li, *A Border Gateway Protocol 4 (BGP-4)*, Internet Engineering Task Force, RFC 1771, 1995.

[49] X. Tang and A. Zakhor, "Matching pursuits multiple description coding for wireless video," in *Proceedings of 6th International Conference on Image Processing (ICIP)*, October 2001, vol. 1, pp. 926–929.

[50] T. Nguyen and A. Zakhor, "Path diversity with forward error corection (pdf) system for packet switched networks," in *INFOCOM*, San Francisco, CA, April 2003.

[51] S. Agarwal, C. Chuah, and H. Katz, "Opca: Robust interdomain policy routing and traffic control," in *IEEE Openarch*, April 2003.

[52] D.G. Andersen, H. Balakrishnan, M.F. Kaashoek, and R. Morris, "The case for resilient overlay networks," in *Proceeding of HotOS VIII*, May 2001.

[53] A.L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, pp. 509–512, October 1999.

[54] H. Eriksson, "The multicast backbone," *Communications of the ACM*, vol. 37, pp. 54–60, 1994.

[55] Stefan Savage, Tom Anderson, Amit Aggarwal, David Becker, Neal Cardwell, Andy Collins, Eric Hoffman, John Snell, Amin Vahdat, Geoff Voelker, and John Zahorjan, "Detour: a case for informed internet routing and transport," *IEEE Micro*, vol. 19, no. 1, pp. 50–59, january 1999.

[56] netVMG, http://www.netvmg.com/index2.html, *netVMG*.

[57] *Traceroute*, http://www.tracert.com.

[58] *CAIDA Tools*, http://www.caida.org/tools/, 2001.

[59] V. Paxon, G. Almes J. Mahdavi, and M. Mathis, *Framework for IP Performance Metrics, RFC 2330*, IETF, May 1998.

[60] V. Jacobson, *pathchar - A tool to Infer Characteristics of Internet Paths*, ftp://ee.lbl.gov/pathchar/, 1997.

[61] *Internet topology generator*, http://www.cs.bu.edu/brite.

[62] A. Khanna and J. Zinky, "The revised arpanet routing metric," in *ACM SIGCOMM*, 1989, pp. 45–46.

[63] R. Neff and A. Zakhor, "Very low bit rate video coding based on matching pursuits," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 158–171, February 1997.

[64] X. Tang and A. Zakhor, "Matching pursuits multiple description coding for wireless video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 1, pp. 566–575, June 2002.

[65] T. Nguyen and A. Zakhor, "Matching pursuits based multiple description video coding for lossy environments," in *International Conference on Image Processing (ICIP)*, Barcelona, Spain, September 2003.

[66] M. Borella, D. Swider, S. Uludag, and G. Brewster, "Internet packet loss: Measurement and implications for end-to-end qos," in *Proceedings of ICPP Workshop on Architectural and OS Support for Multimedia Applications Flexible Communication Systems*, Minneapolis, MN, 1998, pp. 3–12.

[67] http://www.kazaa.com.

[68] A. Reibman, "Optimizing multiple description video coders in a packet loss environment," in *Packet Video Workshop*, April 2002.

[69] R. Zhang, S. Regunathan, and K. Rose, "Video coding with optimal inter/intra-mode switching for packet loss resilience," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 977–995, June 2000.

[70] R. Neff and A. Zakhor, "Matching pursuits video coding-part2:operational models for rate and distortion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 1, pp. 27–39, January 2002.

[71] http://www.gnutella.com.

[72] http://www.isc.org.

[73] IMTC, International Multimedia Telecommunications Consortium, http://www.imtc.org/h323.htm, *H.323 Standard, Packet-based multimedia communications systems.*

[74] PlanetLab, http://www.planet-lab.org.

[75] P.A. Chou, A.E. Mohr, A. Wang, and S. Mehrotra, "Error control for receiver-driven layered multicast of audio and video," *IEEE Transactions on Multimedia*, vol. 3, pp. 108–22, March 2001.

[76] A. EL Gamal and T. Cover, "Achievable rates for multiple descriptions," *IEEE Transactions on Information Theory*, vol. 6, pp. 8151–8157, November 1982.

[77] A. Medina, I. Matta, and J. Byers, "On the origin of power-laws in internet topologies," *ACM Computer Communication Review*, pp. 160–163, April 2000.

[78] A. Mohr, E. Riskin, and R. Ladner, "Unequal loss protection: Graceful degradation over packet erasure channels through forward error correction," *IEEE Journal on Selected Areas in Communication*, vol. 18, pp. 819–828, April 2000.

[79] T. Nguyen and A. Zakhor, "Distributed video streaming," *Submitted to IEEE/ACM Transactions on Multimedia and Networking*.

[80] A. R. Reibman, H. Jafakhani, Y. Wang, M.T. Orchard, and R. Puri, "Multiple description video coding using motion-compensated temporal prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 3, pp. 193–204, March 2002.

[81] G. De Los Reyes, A. Reibman, S. Chang, and J. Chuang, "Error-resilient transcoding for video over wireless channels," *IEEE Transactions on Multimedia*, vol. 18, pp. 1063–1074, june 2000.

[82] J. Robinson and Y. Shu, "Zerotree pattern coding of motion picture residues for error-resilient transmission of video sequences," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 1099–1110, June 2000.

[83] V. Vaishampayan, "Design of multiple description scalar quantizers," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 821–834, May 1993.

[84] W. Wei, B. Wang, and D. Towsley, "Continuous-time hidden markov models for network performance evaluation," *Performance Evaluation*, vol. 49, no. 1-4, pp. 129–146, 2002.

[85] M. Zorzi and R. Rao, "On the statistics of block errors in bursty channels," *IEEE Transations on Communication*, vol. 45, no. 6, June 1997.

[86] D.G. Andersen, "Resilient overlay networks," in *Master Thesis, MIT*, May 2001.

[87] P.A. Chou and A. Sehgal, "Rate-distortion optimized for receiver-driven streaming over best effort networks," in *Packet Video Workshop*, April 2002.

[88] K. Lai and M. Baker, "Measuring link bandwidths using a deterministic model of packet delay," in *ACM SIGCOMM*, 2000, pp. 283–294.

[89] Y.J. Liang, E. Setton, and B. Girod, "Channel adaptive video streaming using packet path diversity and rate-distortion optimized reference picture selection," in *IEEE Fifth Workshop on Multimedia Signal Processing*, December 2002.

[90] Y.J. Liang, J.G. Apostolopoulos, and B. Girod, "Analysis of packet loss for compressed video: Does burst length matter?," in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, April 2003.

[91] M. Luby, M. Mitzenmacher, M. Shokrollahi, Spielman D, and V. Stemann, "Practical loss-resilient codes," in *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, May 1997, pp. 150–159.

[92] H. Ma and M. El Zarki, "Broadcast/multicast mpeg-2 video over wireless channels using header redundancy fec strategies," in *Proceedings of The International Society for Optical Engineering (SPIE)*, November 1998, vol. 3528, pp. 69–80.

[93] T. Nguyen and A. Zakhor, "Distributed video streaming," in *Proceedings of the SPIE - The International Society for Optical Engineering, Multimedia Computing and Networking (MMCN)*, San Jose, CA, January 2002, vol. 4673, pp. 186–95.

[94] T. Nguyen and A. Zakhor, "Protocols for distributed video streaming," in *International Conference on Image Processing (ICIP)*, Rochester, NY, September 2002.

[95] T. Nguyen and A. Zakhor, "Distributed video streaming with forward error correction," in *Packet Video Workshop*, Pittsburg, PA, April 2002.

[96] M.T. Orchard, Y. Wang, and V. Vaishampayan, "Redundancy rate-distortion analysis of multiple description coding using pairwise correlating transforms," in *Proceedings of International Conference on Image Processing (ICIP)*, October 1997.

[97] L. Ozarow, "On source coding with two channels and three receivers," in *Bell System Technical Journal*, December 1980, vol. 59.

[98] V.N. Padmanabhan, H.J. Wang, P.A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *ACM NOSSDAV*, May 2002.

[99] S. Servetto, K. Ramchandran, K. Nahrstedt, and A. Ortega, "Optimal segmentation of a vbr source for its parallel transmission over multiple atm connections," in *Proceedings of International Conference on Image Processing (ICIP)*, October 1997, vol. 2.

[100] S. Seshan and M. Stemm, "Spand: Shared passive network performance discovery," in *Usenix Symposium on Internet Technologies and Systems*, December 1997, pp. 135–146.

[101] S. Saroui, P.K. Gummadi, and S.D. Gribble, "Measurement study of peer-to-peer file sharing systems," in *Multimedia Computing and Networking*, January 2002, pp. 156–70.

[102] W. Tan and A. Zakhor, "Error control for video multicast using hierarchical fec," in *Proceedings of 6th International Conference on Image Processing (ICIP)*, October 1999, vol. 1, pp. 401–405.

Appendices

A Chapter 2 Appendix

A.1 Appendix:Proof of optimality for rate allocation algorithm

Statement: Consider N senders with the loss rate during the interval $(t, t+\delta)$ for the i^{th} sender denoted by $L(i,t)$ and its available TCP-friendly bandwidth by $B(i,t)$. Suppose we want to choose a subset of the senders to stream a video with total bit rate of $S_{req}(t)$. Without loss of generality, assume the senders are ordered in such a way that for $i < j$, we have $L(i,t) < L(j,t)$. Then to minimize the total loss rate given by

$$F(t) = \sum_{i=1}^{N} L(i,t)S(i,t) \tag{A.1}$$

subject to

$$
\begin{aligned}
&0 \leq S(i,t) \leq B(i,t) \\
&\sum_{i=1}^{N} S(i,t) = S_{req}(t)
\end{aligned} \qquad \text{(A.2)}
$$

We must choose sending rate for sender i, $S(i,t) = B(i,t)$ for $i = 1...M-1$, $S(M,t) = \sum_{i=1}^{M} B(i,t) - S_{req}(t)$ where M is the smallest integer such that $\sum_{i=1}^{M} B(i,t) \geq S_{req}(t)$ and choose $S(i,t) = 0$ for $M < i \leq N$.

Proof: We prove the above statement by contradiction. Suppose there exists a rate allocation with senders in which $F(t)$ is at minimum, say $F'(t)$, for some $0 < i' < M'$, but $S(i',t) \neq B(i',t)$. Due to constraints (A.2), this implies that $S(i',t) < B(i',t)$. It is then possible to show that by allocating some of the bit rate from sender M' to i', we can achieve smaller loss rate than $F'(t)$. Specifically, if $\Omega = \min\left[S(M',t), B(i',t) - S(i',t)\right]$ is the reallocated bit rate from sender M' to i', then the resulting loss rate will be given by

$$
\begin{aligned}
F^*(t) &= \sum_{k=1}^{M'-1} L(k,t)S(k,t) + L(i',t)\Omega + L(M',t)[S(M',t) - \Omega]) \\
&= \sum_{k=1}^{M'} L(k,t)S(k,t) - [L(M',t) - L(i',t)]\Omega
\end{aligned} \qquad \text{(A.3)}
$$

Since the first term in the above summation is $F'(t)$, and since the second term is always a positive quantity, we have $F^*(t) < F'(t)$. Clearly, this contradicts the assumption that $F'(t)$ is the minimum. QED.

A.2 Protocol Responsiveness

As discussed in Section 2.3, the count threshold γ and the sampling interval ϕ determine the minimum interval during which the sending rates must remain constant, and the responsiveness of sending rate to network conditions. To examine the responsiveness of the proposed protocol as a function of γ, we keep other parameters at fixed values shown in Table A.1, and vary the γ from 20 to 200 which corresponds to the range of 2 to 20 seconds, during which the sending rates must remain constant. We perform Internet experiment in which, packets of 500 bytes are sent simultaneously at an aggregate rate of 880kbps or 220 packets per seconds from Sweden and Belgium to U.C. Berkeley. Initially, simulated packet loss rates of Sweden and Belgium are set to 3% and 6%, respectively. At $t = 50$ seconds, Belgium loss rate reduces to 0.3%. Figures A.1 (a)-(d) shows the responsive throughputs for different values of γ.

Initially, both senders send packets at equal rate, i.e., 110 packets per second. As shown in Figure A.1(a), for $\gamma = 20$, at $t = 0$ second, throughputs of the senders fluctuate for about 15 seconds before converging to stable values. This throughput fluctuation is due to using only a small number of sampled bandwidths, i.e., $\gamma = 20$, to decide whether the sending rates need to be changed. On the other hand, when larger number of sampled bandwidths are used, i.e., $\gamma = 40, 100, 200$, the sender's throughputs exhibit no fluctuation, and change the to stable values after 9, 11, and 23 seconds as shown in Figures A.1(b)-(d), respectively. At $t = 50$ seconds, the loss rate of Belgium reduces to 0.3%, Figures A.1(a)-(c) show transient throughputs right before the stable throughputs after 10, 14, and 16 seconds for $\gamma = 20$ 40, and 100, respectively. For $\gamma = 200$, there is no transient throughputs, the protocol results in stable throughputs after 20 seconds, respectively. The general observation is that, small values of γ allows quick response to the network, however, it may result in unstable sending rates. Larger values of γ result in less fluctuation in throughputs at the expense of slow responsiveness to network conditions. We observe that, $\gamma = 40$ provides a reasonable tradeoff between throughput stability and responsiveness in most situations.

w(width of the hysteresis window)	$0.1S(i)$
ϕ(sampling interval for the hysteresis window)	100 milliseconds
γ(count threshold)	20-200
$PWIN$(time window for estimating loss)	128RTT
P(packet size)	500 bytes

Table A.1: *Simulation parameters for various* γ*.*

A.3 Throughput Reduction due to Delay Differences between Senders

As discussed previously in section 2.4.2, when using the "min" strategy, a throughput reduction may result due to the delay differences between the senders. Although this throughput reduction is small, for completeness, in this Appendix, we derive the amount of throughput reduction as a function of the sending rates and delays between the senders and the receiver. Consider three senders 1, 2, and 3 with respective respective propagation delays to the receiver D_1, D_2, and D_3. Without loss of generalities, we assume $D_1 \geq D_2 \geq D_3$. Due to these delay differences, the senders receive their control packet at different times as shown in Figure A.3. As described previously, upon receiving the control packet, each sender immediately changes its sending rate to the value specified in the control packet. For the period starting from the time sender 1 receives the control packets until sender 2 receives the control packet, sender 1 sends packets at the new rate S_1^{new} while senders 2 and 3 still send packets at the old rates S_2^{old} and S_3^{old}. Since the packets sent by senders 2 and 3 are not synchronized with sender 1 during this period, we ignore these packets and assume that their effective sending rates equal to zero. Hence, the amount of "synchronized" data, i.e. useful data sent during the interval between the arrival times of the control packets for the sender 1 and sender 3 equals to

$$S_1^{new}(D_2 - D_1) + S_2^{new}(D_3 - D_2)$$

More generally, if there are N senders, the amount of "synchronized" data sent during the interval between the arrival times of the control packets for the sender 1 and sender N equals to

$$S_1^{new}(D_2 - D_1) + S_2^{new}(D_3 - D_2) + ... + S_{N-1}^{new}(D_N - D_{N-1})$$

If S is the total sending rate, then the reduction in the amount of data R sent during this period is

$$
\begin{aligned}
R &= S(D_N - D_1) - (S_1^{new}(D_2 - D_1) + S_2^{new}(D_3 - D_2) + ... + S_{N-1}^{new}(D_N - D_{N-1})) \\
&= D_N(S - (S_1^{new} + S_2^{new} + ... + S_{N-1}^{new})) + S_1^{new} D_1 + S_2^{new} D_2 + ... + S_{N-1}^{new} D_{N-1} - SD_1 \\
&= S_1^{new} D_1 + S_2^{new} D_2 + ... + S_N^{new} D_N - SD_1 \\
&= (\sum_{i=1}^{N} S_i^{new} D_i) - SD_1
\end{aligned}
$$

Where we make use $S_N^{new} = S - (S_1^{new} + S_2^{new} + ... + S_{N-1}^{new})$.

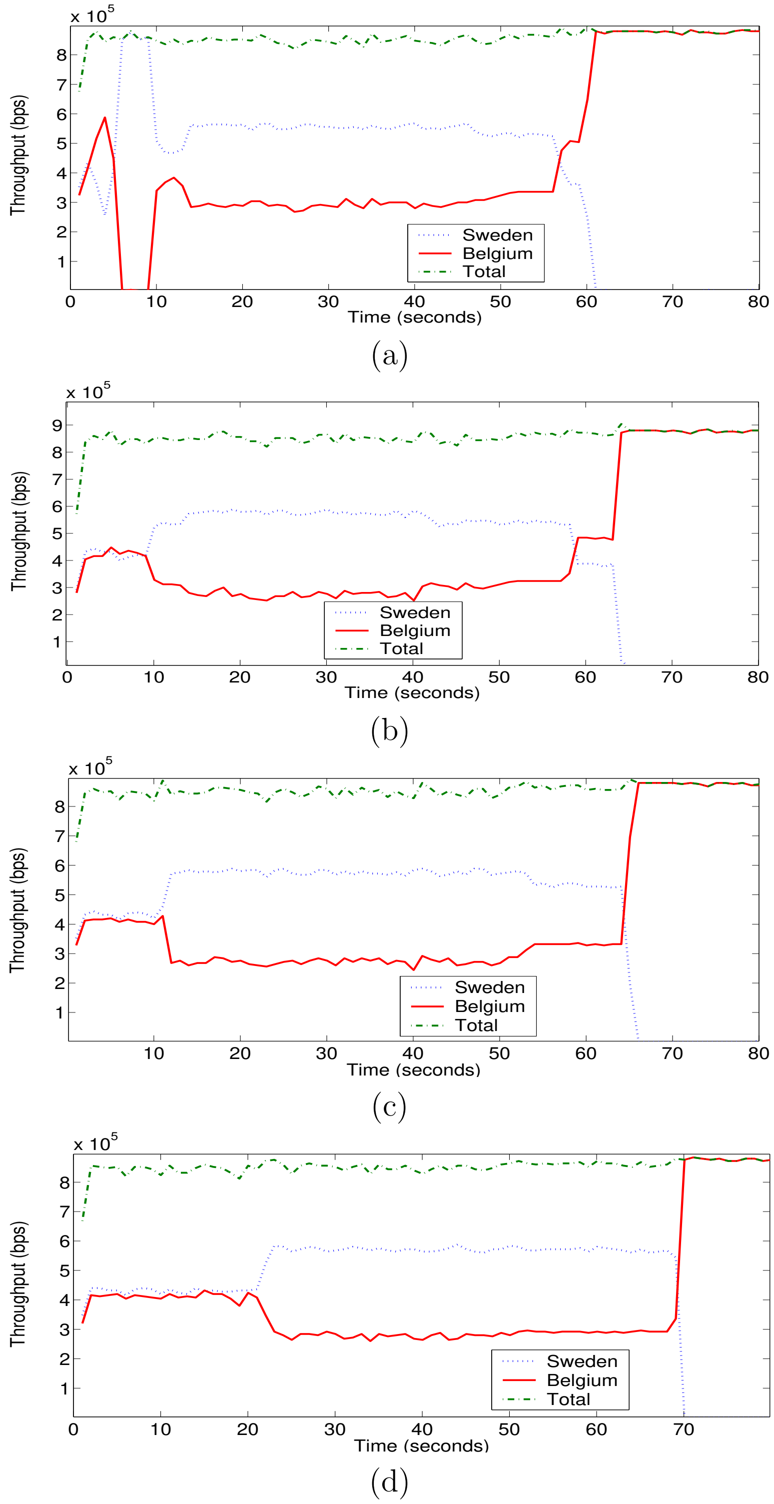


Figure A.1: *Protocol responsiveness for various values of* γ*; (a)* $\gamma = 20$*; (b)* $\gamma = 40$*; (c)* $\gamma = 100$*; (d)* $\gamma = 200$*.*

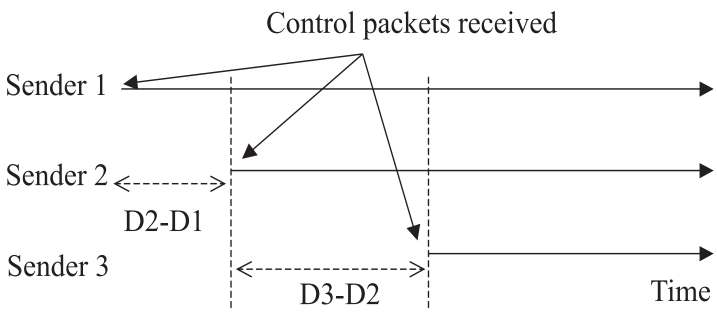


Figure A.2: *Illustration of throughput reduction*

B Chapter 3 Appendix

B.1 Procedure for computing $P(m, k, N_m)$

To compute $C(K, N_0, N_1)$ in Section 3.5, we first compute $P(m, i, N_m)$ based on the given network parameters (μ_g^m, μ_b^m) of sender m as follows. We use notations in Table B.1:

$S_m(n) \in \{g, b\}$	State of sender m after it sends n packets
$L_m(n)$	Number of lost packets out of n packets sent by sender m
$P_m^{loss}(i)$	Packet loss probability when sender m is in state i
p_{ij}^m	Transition probability from state i to state j for sender m

Table B.1: *Notations for computing* $P(m, kN_m)$.

Note that p_{ij}^m depends not only on the parameters μ_g^m and μ_b^m, the rates at which the state of sender m changes from "good" to "bad" and vice versa, but also on the rate that sender m sends the packets according to Equations 3.1 through 3.4. Then,

$$\phi_{ij}^m(k, n) \triangleq Prob(L_m(n) = k, S_m(n) = j | S_m(0) = i)$$

denotes the probability that sender m is in state j, and there are k lost packets after it sends n packets, given that it is initially in state i. We can compute $\phi_{ij}^m(k, n)$ recursively by conditioning on the previous state l, and by using the total probability theorem to obtain

$$\phi_{ij}^m(k,n) = \sum_{l\in g,b}[\phi_{il}^m(k-1,n-1)p_{lj}^m P_m^{loss}(j) + \phi_{il}^m(k,n-1)p_{lj}^m(1-P_m^{loss}(j))]$$

for all $k \geq 0$ and $n \geq 0$, with the boundary conditions:

$$\phi_{ij}^m(0,0) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

$$\phi_{ij}^m(k,n) = 0 \text{ for } n < k$$

The above boundary conditions hold because of following arguments. If sender m does not send packet and hence does not change its state, there will certainly be no lost packets. Therefore $\phi_{ij}^m(0,0) = 1$ for $i = j$. On the other hand, by definition, it is impossible to have sender m change its state without sending a packet, hence $\phi_{ij}^m(0,0) = 0$ for $i \neq j$. Finally, $\phi_{ij}^m(k,n) = 0$ for $n < k$ since number of lost packets cannot exceed the number of sent packets. Now, since $P(m,k,N_m)$ is the probability of k lost packets out of N_m packets sent by sender m, regardless of the initial and final states, we marginalize $\phi_{i,j}^m(k,N_m)$ to obtain

$$P(m,k,N_m) = \sum_{i\in\{g,b\}}\sum_{j\in\{g,b\}} \pi_i^m \phi_{ij}^m(k,N_m)$$

where $\pi_g^m = \mu_b^m/(\mu_g^m + \mu_b^m)$ and $\pi_b^m = \mu_g^m/(\mu_g^m + \mu_b^m)$ are the steady-state probabilities of sender m being in "good" and "bad" states, respectively. The irrecoverable probability is then computed as

$$C(K,N_0,N_1) = \sum_{j=N-K+1}^{N_A+N_B}\sum_{i=0}^{j} P(A,i,N_A)P(B,j-i,N_B)$$

B.2 Online Estimation of Network Parameters

Our rate allocation algorithm uses the network parameters, particularly, the average good and bad times $1/\mu_g$, $1/\mu_b$ to derive the optimal sending rates for the senders. When the network characteristics are stationary for sufficient duration of time, one can use sophisticated off-line algorithms such as the Hidden Markov Model inference algorithms [47][84] to accurately estimate the network parameters. On the other hand, when the network characteristics are highly dynamic, an on-line algorithm is preferred for fast response to changing network conditions. In this Appendix, we characterize the responsiveness of our rate allocation algorithm to network conditions using a simple on-line algorithm for estimating the network parameters.

For each route, our online algorithm estimates the network parameters as follows. Let C_g and C_b denote the length of good and bad runs, i.e. the number of consecutively received and lost packets, respectively, as shown in Figure B.2. The exponential weighting estimates of μ_g and μ_b are computed recursively using the following equations

$$1/\mu_b = \lambda(1/\mu_b) + (1-\lambda)(C_b^{curr}/S)$$

$$1/\mu_g = \lambda(1/\mu_g) + (1-\lambda)(C_g^{curr}/S)$$

where C_g^{curr} and C_b^{curr} denote the length of the most recent good and bad runs, respectively, $\lambda \in (0,1)$ is the weighting factor, and S is the sending rate. Note that we estimate the current bad (good) times as the product between the number of consecutive lost (received) packets and the sending interval. Smaller λ allows faster response to the changes in network conditions. However, smaller λ may result in large fluctuation in the estimates of $1/\mu_g$ and $1/\mu_b$, and thus, triggers changes in sending rates unnecessarily. On the other hand, larger λ produces more stable $1/\mu_b$ and $1/\mu_g$ at the expense of protocol's slow response to the changes in network conditions.

Once $1/\mu_g$ and $1/\mu_b$ are estimated, they are used in the rate allocation algorithm to compute the new rate. However, since $1/\mu_g$ and $1/\mu_b$ are continuous real values, ideally we would want a slight change in these μ's not to trigger new sending rates, and hence not sending a large number of control packets unnecessarily. Because of this, we compute new sending rates only when the current estimates of $1/\mu$'s is significantly different from the old $1/\mu$'s. Let $1/\mu_g^{cur}$, $1/\mu_b^{cur}$ be the current estimates of average good and bad times, and μ_g^{old}, μ_b^{old} be the old estimates of average good and bad times. Then the new sending rates are computed only when

$$1/\mu_b^{old} - \alpha_b > 1/\mu_b^{cur} > 1/\mu_b^{old} + \alpha_b$$

$$1/\mu_g^{old} - \alpha_g > 1/\mu_g^{cur} > 1/\mu_g^{old} + \alpha_g$$

where α_g and α_b are the guarding thresholds.

To examine the responsiveness of our rate allocation to different parameters, we perform experiments in which, two senders from Belgium and Sweden stream packets simultaneously at the total rate of 800kbps to the receiver at U.C. Berkeley. Packets are protected using RS(60,46). We set the network and protocol parameters for two routes as shown in Table B.2. The values of guarding thresholds α_g and α_b are chosen experimentally to be 200 and 15 milliseconds respectively for reasonable performance.

$1/\mu_g^{1(2)} = 1$ second	average good time of Sweden(Belgium) route
$1/\mu^1(2)_b = 20(40)$ milliseconds	average bad time of Sweden(Belgium) route
$\alpha_g^{1(2)} = 200$ milliseconds	guarding threshold for good time of Sweden(Belgium)
$\alpha_g^{1(2)} = 15$ milliseconds	guarding threshold for bad times of route Sweden(Belgium)
$\lambda = 0.85, 0.9, 0.96, 0.98$	exponential weighting factor

Table B.2: *Network and protocol parameters.*

Initially, the total rate is divided equally between two senders at 400kbps each. Since the average bad time for Belgium route is greater than that of Sweden route, after several seconds, the protocol tries to allocate more packets to Sweden route. Figures B.2(a) and B.2(b) show the sending rates and the corresponding throughputs of the two senders for $\lambda = 0.85$, respectively. Using $\lambda = 0.85$, our protocol adapts to the optimal rates after approximately 12 seconds. However, the sending rates still fluctuate considerably after 12 seconds. This means that control packets are sent unnecessarily since the network characteristics of the two routes remain the same. As seen in Figures B.3 through B.5, increasing λ results in more stable asymptotic rates with less fluctuations, at the expense of slower adaptation.

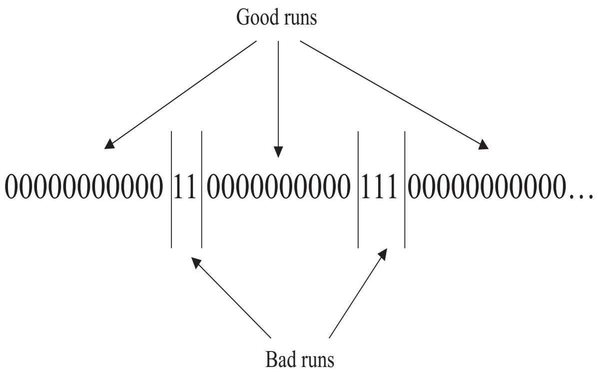


Figure B.1: *Illustration of network parameters estimation; 0 and 1 indicate received and lost packets, respectively.*

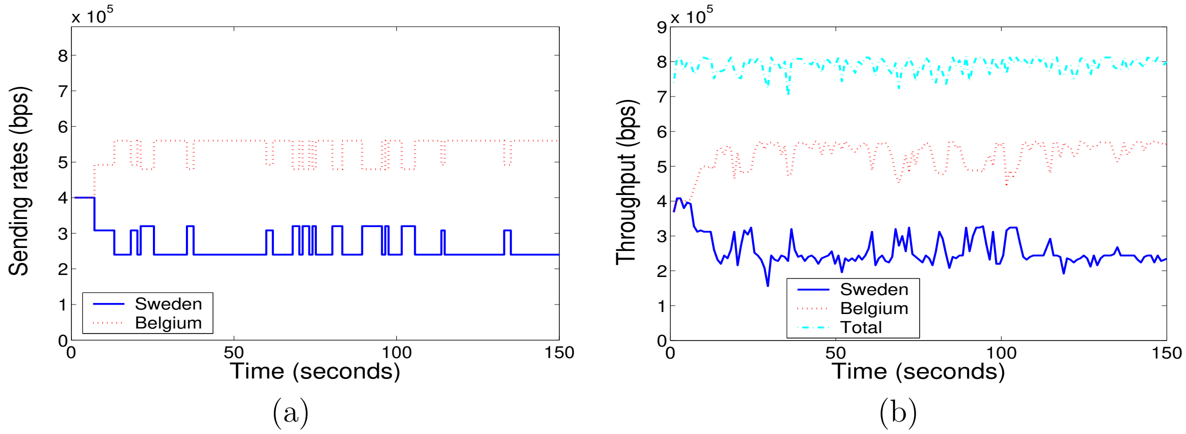
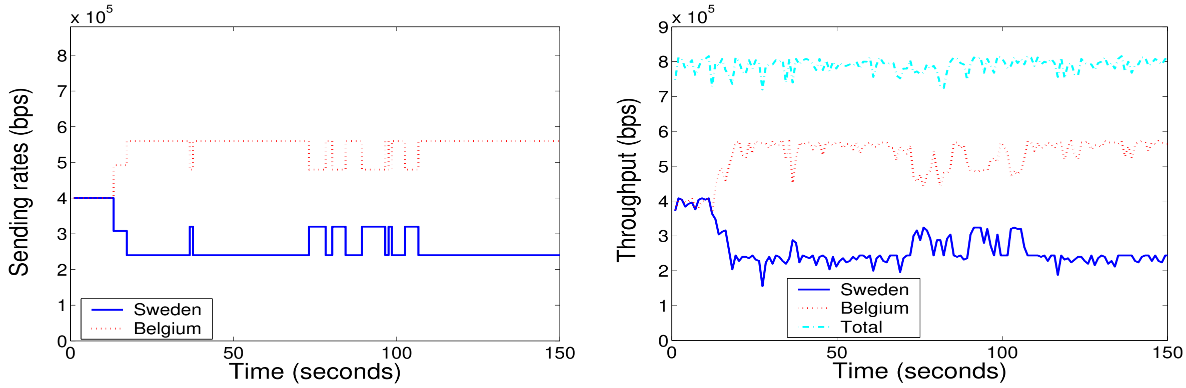
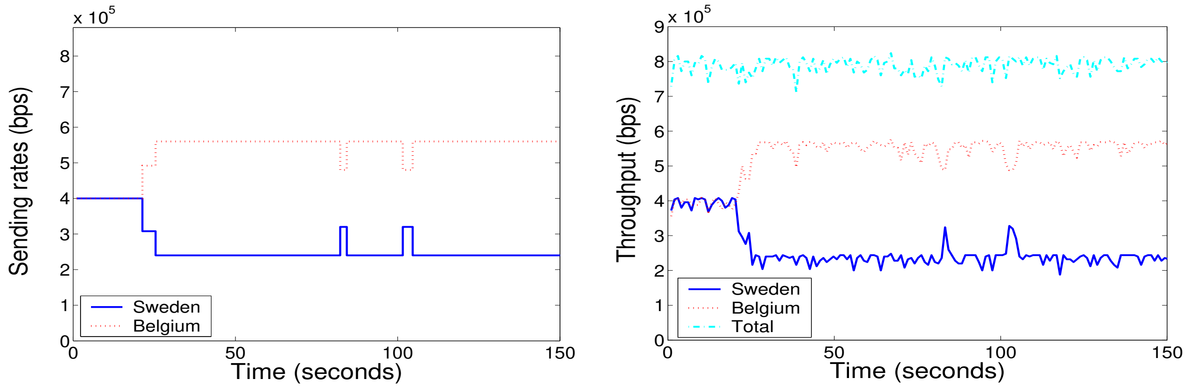


Figure B.2: *Protocol responsiveness for* $\lambda = 0.85$*; (a) sending rate triggered at the receiver; (b) corresponding throughputs at the receiver.*



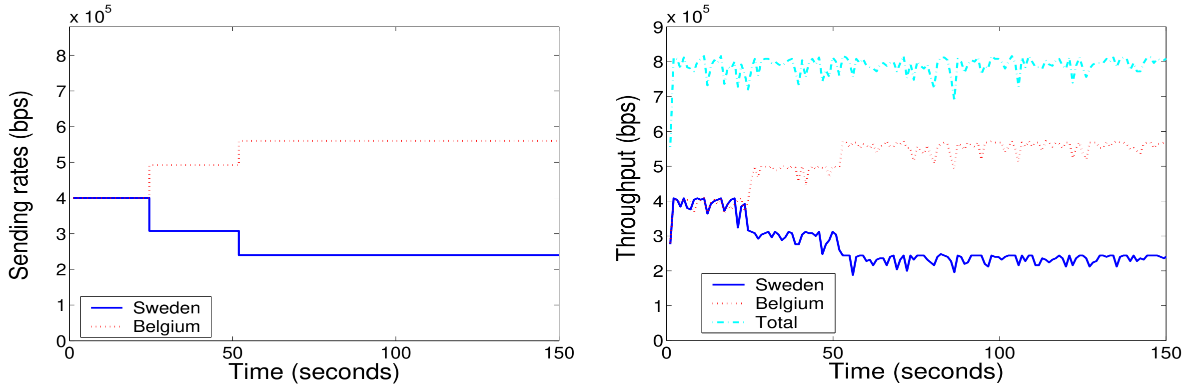
(a) (b)

Figure B.3: *Protocol responsiveness for* $\lambda = 0.9$*; (a) sending rate triggered at the receiver; (b) corresponding throughputs at the receiver.*



(a) (b)

Figure B.4: *Protocol responsiveness for* $\lambda = 0.96$*; (a) sending rate triggered at the receiver; (b) corresponding throughputs at the receiver.*



(a) (b)

Figure B.5: *Protocol responsiveness for* $\lambda = 0.98$*; (a) sending rate triggered at the receiver; (b) corresponding throughputs at the receiver.*