Rate-Distortion Optimized Packet Scheduling for Video Streaming

Jakov Čakareski (Jacob Chakareski)

Thesis: Doctor of Philosophy
Electrical and Computer Engineering
Rice University, Houston, Texas (November 2005).
INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI Microform 3216679
Copyright 2006 by ProQuest Information and Learning Company. All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

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

Rate-Distortion Optimized Packet Scheduling for Video Streaming

by

Jakov Čakareski (Jacob Chakareski)

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
Doctor of Philosophy

APPROVED, THESIS COMMITTEE:

\[\text{Signature}\]
Bernd Girod, Thesis Director
Professor
Electrical Engineering
Stanford University

\[\text{Signature}\]
Richard G. Baraniuk, Thesis Chair
Professor
Electrical and Computer Engineering

\[\text{Signature}\]
Edward W. Knightly
Professor
Electrical and Computer Engineering

\[\text{Signature}\]
Richard A. Tapia
Noah Harding Professor
Computational and Applied Mathematics

Houston, Texas

November, 2005
“Our greatest glory consists not in never falling, but in rising every time we fall.”

-- Confucious, BC 551-479
Rate-Distortion Optimized Packet Scheduling for Video Streaming

Jakov Čakareski (Jacob Chakareski)

Abstract

Internet video streaming places new demands on source coding and network transport algorithms. The challenge is to deliver compressed video packets before their play-out deadline, despite of varying throughput, packet delay, and loss. This problem has to be solved in a way that simultaneously maximizes the video quality at the streaming client, meets transmission rate limitations, and satisfies latency constraints.

The most significant recent advance in streaming technology is the emergence of rate-distortion optimized streaming techniques that take into account packet importance and knowledge about the channel in a Lagrange rate-distortion cost function $J = D + \lambda R$. The research in this thesis aims to overcome the current limitations in rate-distortion optimized packet scheduling, such that these techniques can be applied to a broader range of streaming problems, and a better end-to-end performance can be achieved.

In particular, first we develop a new advanced framework for rate-distortion optimized video streaming which incorporates for the first time a general description scheme for the rate-distortion characteristics of packetized compressed video in the event of lost or omitted packets, and an iterative algorithm for computing optimal
packet schedules in the presence of error concealment.

Then, we derive instances of our framework for several advanced video streaming architectures, applications, and coding techniques, some of which could not be previously combined with rate-distortion optimized streaming due to its initial limitations. These include streaming over multiple network paths or from multiple media servers, streaming from an intermediate network proxy, and streaming with rich acknowledgements.

Finally, we assess the performance of our framework through series of systematic experiments in all of these scenarios. The experimental results are accurately predicted through analysis. In addition, we evaluate the performance of rate-distortion optimized packet scheduling for streaming over network traces of packet losses and packet delays collected in the Internet.
Acknowledgments

The origins of the research out of which this thesis was born can be traced back to a meeting held at the Data Compression Conference (DCC) in Snowbird Hill, Utah, in March 2002. Less than a year ago, during the summer of 2001, at Microsoft Research in Redmond, Washington, I was involved in a research project on scheduling packet transmissions over computer networks for media streaming applications. Highly motivated about continuing along the lines of this research, I was looking forward to the meeting with Prof. Bernd Girod from Stanford University at the DCC. Its purpose was to discuss prospective collaboration and apparently we shared the same enthusiasm for the subject in question. Therefore it took us less than five minutes to agree on a summer project that I was supposed to work on at Stanford under Prof. Girod’s guidance.

Since then everything started moving at light speed for me and the summer project turned into a continued stay at Stanford University for the rest of my Ph.D. studies. In August 2002, I took a leave of absence from Rice University and joined Stanford as a visiting Ph.D. student. For the next 2.5 years I worked on my doctoral research in the Information Systems Laboratory in the Department of Electrical Engineering. There, I was a member of the Image, Video and Multimedia Systems (IVMS) group headed by Prof. Girod.

Therefore, it is to Prof. Girod that the foremost acknowledgement regarding the present thesis should go. He provided the necessary environment for performing the
research that the thesis describes. In addition, it is under his guidance that this thesis has been brought to its present form. Every encounter with my advisor has been a new experience for me and I learned a great deal from them. Therefore, I will be always grateful to him for shaping my career as a doctoral student in such a profound way. Moreover, the influence of the IVMS group should also be noted in this regards. Being affiliated with its past and present members, highly motivated and successful individuals, have provided an additional impetus to my doctoral studies. Their names deserve to be recognized here: Anne Aaron, Chuo-Ling Chang, Sang-eun Han, Mark Kalman, Prashant Ramanathan, Shantanu Rane, David Rebolledo-Monedero, Eric Setton, Xiaoqing Zhu, Yi Liang, Markus Flierl and Rui Zhang. There is one more person at Stanford that I would like to acknowledge here. That is our administrative assistant Kelly Yilmaz. Her unlimited administrative skills and great understanding for the students’ needs certainly made my stay at Stanford more relaxed and enjoyable.

The person that introduced me to the subject that the thesis deals with and that has collaborated with me ever since is Dr. Philip A. Chou. At Microsoft Research in Redmond he was my mentor and I learned enormously from him. My respect for him can be succinctly described by the statement that his work will always be inspiring for me. Our collaboration has resulted into several important contributions to the present thesis. Finally, his continuous friendship and support have helped me persevere throughout the work related to this thesis and see it to completion. Therefore, because of all these, a special acknowledgement goes to him at this point.

Next, I would like to acknowledge Professors Richard Baraniuk, Edward Knightly and Richard Tapia from Rice University for agreeing to serve on my thesis committee. They have provided valuable input for my work, and I acknowledge the time
they have spent with me on this thesis. At Rice, several other people deserve an
acknowledgement. My dear friends, Ahmad (Farbod) Khoshnevis, Bahareh Sadeghi,
and Lavu Sridhar, for everything that they have done for me during the course of
my Ph.D. studies. I will always have a great appreciation for their unconditional
friendship and support. Of course, my gratitude goes also to Tarik, Sinan, Jaber,
Violeta, Nadeem, Kiran, Vikram, and other members of the Rice community for
making my early Ph.D. days there memorable.

Furthermore, throughout the work on this thesis I have made friends among
colleagues in the field, such as John Apostolopoulos, Mihaela van der Schaar, and
Pascal Frossard. Their words of advice and support have been a continuous source
of encouragement for me to carry on the work related to this thesis despite all the
challenges that I needed to face along the way. It is for this that I would like to thank
them here. In addition, and for the same reason, the name of Professor Alexander
Emanuel from the Worcester Polytechnic Institute needs to be recognized here.

Last, but certainly not least, I would like to acknowledge my friends and family
in Macedonia, whose love and companionship have proved critical for making this
thesis become true. It is to them, especially to my parents, that I dedicate the thesis.

Lausanne, Switzerland, November 2005
Contents

Abstract iii
Acknowledgments v
List of Illustrations x
List of Tables xiv

1 Introduction 1

2 Background 5
  2.1 Rate-Scalable Video ...................... 6
  2.2 Error Control ................................ 10
  2.3 Rate-Distortion Optimized Packet Scheduling .. 14
  2.4 Current Limitations of Rate-Distortion Optimized Packet Scheduling 16
    2.4.1 Diversity Techniques .................. 18
    2.4.2 Proxy Caching and Monitoring Agents .... 21

3 Rate-Distortion Optimization (RaDiO) Framework 23
  3.1 Source Characterization .................... 23
  3.2 Channel Characterization .................. 27
  3.3 Rate-Distortion Optimization using Iterative Sensitivity Adjustments 30
  3.4 Window Control ................................ 41
  3.5 Computational Complexity .................. 44
4 Computing the Optimal Policy $\pi^*$

4.1 Streaming with Diversity ........................................ 47
  4.1.1 Sender-driven transmission .................................. 48
  4.1.2 Receiver-driven transmission ................................ 51
4.2 Proxy-Driven Streaming from the Network Edge ................. 53
  4.2.1 Proxy-Driven Transmission .................................. 53
  4.2.2 No previous DATs ............................................. 56
  4.2.3 With previous DATs .......................................... 60
4.3 Streaming with Rich Acknowledgements .......................... 60

5 Experimental Results .............................................. 68

5.1 Experiments with Statistical Channel Models .................... 70
  5.1.1 Streaming with Diversity .................................... 71
  5.1.2 Proxy-Driven Streaming from the Network Edge ............ 78
  5.1.3 Streaming with Rich Acknowledgements ..................... 84
5.2 Experiments with Internet Packet Traces ........................ 89
  5.2.1 Channel Estimation .......................................... 90
  5.2.2 Traces without Packet Losses ................................ 92
  5.2.3 Traces with Uniformly Distributed Packet Losses .......... 97

6 Conclusions .......................................................... 100

Glossary ............................................................. 105

Bibliography .......................................................... 106
Illustrations

2.1 Internet Video Streaming: conditions and requirements. ................ 5
2.2 Scalable audio-video presentation. ....................................... 6
2.3 Switching between Stream 2 and Stream 1 with SP frames. .......... 10
2.4 Implementation of the PET scheme with Reed-Solomon codes.
   (Original figure provided via PowerPoint slides in April 2003,
   courtesy of P.A. Chou [1].) .................................................. 12
2.5 Delay of Internet packets sent in 30 ms intervals between Santa
   Clara, CA and Cambridge, MA. The trace shows that delays of
   successive Internet packets are often highly correlated. ............... 17

3.1 Excess transmission rate $\Delta R$ (%) vs. Average packet loss rate $\epsilon_A$
   (%) for streaming (top left) Foreman, (top middle) Mother &
   Daughter, (top right) Carphone, (bottom left) Container, (bottom
   middle) News, and (bottom right) Salesman. ........................... 38
3.2 Excess transmission rate $\Delta R$ (%) vs. Ratio play-out delay / mean
   RTT for streaming (top left) Foreman, (top middle) Mother &
   Daughter, (top right) Carphone, (bottom left) Container, (bottom
   middle) News, and (bottom right) Salesman. ........................... 40
3.3 Implementation of window control for media streaming. ............... 43
4.1 Diversity in packet networks. (a) Path diversity. (b) Server diversity. 48
4.2 Markov decision tree for a data unit with packet path diversity. . . . 49
4.3 Streaming on demand to a client through a proxy server. . . . . . . . 54
4.4 The decision tree of a transmission policy for a data unit when no
previous DATs have arrived at the proxy. . . . . . . . . . . . . . . . . . . . . . 56
4.5 Streaming on demand using rich acknowledgments. . . . . . . . . . . 61
4.6 The state space of a Markov decision process for streaming a data
unit with rich acknowledgements. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64

5.1 Coding dependencies between frames in an encoding. . . . . . . . . . 69
5.2 R-D performance for streaming Foreman over $M = 1, 2$ and 3 paths. 73
5.3 R-D performance for streaming Mother & Daughter over
$M = 1, 2$ and 3 paths. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 74
5.4 R-D performance for streaming (top left) Carphone, (top right)
Container, (bottom left) News, and (bottom right) Salesman, over
$M = 1, 2$ and 3 paths. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
5.5 R-D performances for streaming Foreman over $M = 1, 2$ paths and
different state transition models. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
5.6 R-D performance for streaming (left) Foreman and (right) Mother &
Daughter from $M = 1, 2$ and 3 servers. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
5.7 R-D performance for streaming (top left) Carphone, (top right)
Container, (bottom left) News, and (bottom right) Salesman, from
$M = 1, 2$ and 3 servers. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78
5.8 R-D performances for streaming Foreman from $M = 1, 2$ servers and
different state transition models. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
5.9 Proxy-driven vs. sender-driven performance for Foreman: Y-PSNR (dB) vs. Transmission rate (Kbps) in the forward directions on Path 1 and Path 2, for \(\epsilon_2 = 0, 5, 10, 15\%\) and \(\epsilon_1 = 10\%.\)

5.10 Proxy-driven vs. sender-driven performance for Foreman: Y-PSNR (dB) vs. Transmission rate (Kbps) in the forward directions on Path 1 and Path 2, for \(\epsilon_1 = 0, 5, 10, 15\%\) and \(\epsilon_2 = 10\%.\)

5.11 Y-PSNR (dB) of the Foreman sequence vs. Normalized rate (Kbps) on Paths 1 and 2 for proxy-driven and sender-driven systems.

5.12 Y-PSNR (dB) of the (top left) Carphone, (top middle) Container, (top right) Mother & Daughter, (bottom left) News, and (bottom right) Salesman sequences vs. Normalized rate (Kbps) on Paths 1 and 2 for proxy-driven and sender-driven systems.

5.13 Y-PSNR (dB) vs. Transmission rate (Kbps) for streaming Foreman with rich and conventional acknowledgements.

5.14 Y-PSNR (dB) vs. Normalized transmission rate (Kbps) for streaming (top left) Foreman, (top middle) Mother & Daughter, (top right) Carphone, (bottom left) Container, (bottom middle) News, and (bottom right) Salesman with rich and conventional acknowledgements.

5.15 Rate of acknowledgements (Kbps) vs. Transmission rate (Kbps) on the forward channel for streaming Foreman.

5.16 CDF of the packet delay on the (a) Downlink and (b) Uplink.

5.17 R-D performance for streaming Foreman over a network trace with no packet losses.
5.18 R-D performance for streaming *Foreman* over a network trace without packet losses and $\Delta = 200$ ms and 400 ms. .............. 97

5.19 CDF of the packet delay on the (a) Downlink and (b) Uplink. .... 98

5.20 R-D performance for streaming *Foreman* over a network trace with uniformly distributed packet losses. ................. 99
Tables

5.1 Encoding rate and quality for the sequences used in the experiments. 70
5.2 Network path characterization. ............................................. 71
5.3 Number of packets (per data unit) transmitted in forward direction
over Paths 1 and 2 for sender-driven and proxy-driven systems. .... 83
Chapter 1

Introduction

The first commercial products for Internet video streaming have been introduced around 1995. Since then the field has experienced tremendous growth [2]. There were a total of 5.6 billion streams served during the first half of 2004 from hundreds of thousands of streaming media servers. Second only to the Web browser, the two leading streaming media players have the largest active user reach of non-browser based Internet applications* at 54 percent. The trend is only expected to continue in the future and this is happening despite the fact that Internet transmission at present is characterized with variability in packet delay, loss and throughput, which makes data delivery with deadlines difficult.

Therefore, it comes as no surprise that these challenges in Internet video streaming, in conjunction with the commercial promise of the technology, have inspired a considerable body of research work directed towards efficient and low-latency video transmission and coding, as witnessed by a series of recent reviews and special issues, such as [4–17]. In particular, a variety of techniques have been proposed over the last few years to address the problem of error-resilient video streaming, including intra/inter-mode switching [18, 19], error tracking [20], feedback-based reference picture selection [21–23], S-frames [24] and SP-frames [25], forward error correction

*Nielsen//NetRatings, the global standard for Internet audience measurement and analysis, reports that 76 percent of active Web surfers access the Internet using a non-browser based Internet application, such as media players, instant messengers and file sharing applications [3].
[26–31], and multiple description coding [32–37].

The most significant recent advance in streaming technology is the emergence of rate-distortion optimized packet scheduling techniques [38–42] that take into account the different importance of individual packets and knowledge about the channel in a Lagrangian rate-distortion cost function \( J = D + \lambda R \). This is in contrast to the early streaming systems which simply transmitted media packets without regard for the importance of individual packets or the prevalent channel conditions, except, maybe, the available maximum transmission rate. As we will discuss in Chapter 2, these techniques are still in their infancy and there are several initial limitations. Nevertheless, the basic formulation is elegant and the performance improvements reported to date relative to non-Lagrangian heuristics are very encouraging.

**Major Contributions**

The research in this thesis aims to overcome the current limitations in rate-distortion optimized packet scheduling, such that these techniques can be applied to a broader range of streaming problems, and a better end-to-end performance can be achieved. The specific contributions of the thesis can be summarized as follows:

- A distortion model that takes into account error-concealment, which is widely used with video streaming as it allows uninterrupted decoding of dependent packets, even if some of their predecessor packets are lost. All previous works on rate-distortion optimized packet scheduling did not have a distortion model that accounts for error-concealment. Therefore, they could not model realistically the video streaming systems employed in practice today.

- An iterative algorithm for computing rate-distortion optimized schedules for
media packets in the presence of error-concealment. As explained above, error concealment is not present in the distortion models employed by previous works. Therefore, these approaches did not need to account for error concealment in their algorithms for computing optimal packet schedules. On the other hand, since our distortion model takes error-concealment into consideration, we needed to reformulate a previous iterative algorithm so that it considers error-concealment when computing rate-distortion optimized packet schedules. Through our generalized algorithm, the effects of error-concealment can be clearly distinguished in the optimization framework.

- Formulation of different advanced streaming scenarios within our optimization framework such as streaming with diversity, where diversity can be achieved through streaming either over multiple network paths or from multiple media servers, and proxy-driven streaming from the network edge. In addition, we also derive instances of our framework for certain unconventional streaming scenarios, such as streaming with rich acknowledgements.

- Systematic study of the end-to-end performance of rate-distortion optimized packet scheduling in the advanced and unconventional streaming scenarios considered in this thesis. Analysis that accurately predicts the performance of rate-distortion optimized streaming based on the packet loss rates of the underlying communication channels. Evaluation of the performance of rate-distortion optimized packet scheduling for streaming over Internet traces of collected packet losses and packet delays.
Organization of the Thesis

The thesis is organized as follows. In Chapter 2 we provide background on rate-distortion optimized streaming over the Internet and discuss prior work. Related areas such as rate scalability and error control are also discussed in this chapter along with the most relevant select publications. Chapter 3 introduces our framework for advanced rate-distortion optimized video streaming that comprises a distortion model and an iterative algorithm for computing optimal packet schedules with both taking into account error concealment. Next, in Chapter 4, we formulate instances of our framework respectively for the scenarios of streaming with diversity, proxy-driven streaming from the network edge, and streaming with rich acknowledgements. Then, in Chapter 5 we study through simulation experiments the performance of the proposed packet scheduling framework in each of the streaming scenarios under consideration. At the end of the chapter, we study the performance of the optimization framework for streaming over network traces of packet losses and packet delays measured in the Internet. Finally, concluding remarks are provided in Chapter 6.
Chapter 2

Background

Video streaming over the Internet is plagued by low bit-rates, widely varying throughput, widely varying packet loss rates, and widely varying delay. All systems, therefore, require efficient compression, some form of rate scalability, and error-resiliency techniques. (See Figure 2.1.) If low latency is a requirement, special scheduling techniques are required. Conversational services, e.g., require latencies of less than 150 ms. On the other hand, latency requirements for video-on-demand are more relaxed. Commercial systems typically allow a 5-to-10 second latency today, but as higher bit-rate Internet video increasingly competes with television broadcasting, in the future systems will have to cut down video-on-demand latencies to less than one second. In the following, we review the state-of-the-art in each of the above areas, along with select publications most relevant for the research presented in this thesis.

![Diagram](image)

Figure 2.1: Internet Video Streaming: conditions and requirements.
2.1 Rate-Scalable Video

Rate scalability can be elegantly achieved by scalable video codecs that provide layered embedded bit-streams that are decodable at different bitrates, with gracefully degrading quality. The scalability of a representation can change the resolution in time [43], in space [44-46], or in amplitude (“SNR scalability”) [47-49], or in any combination of these. Layered scalable representations have been widely studied for video streaming over best-effort networks, including IP and wireless networks, e.g., in [26, 50-57]. Scalable representations have become part of established video coding standards, such as MPEG and H.263+ [58-60]. (See Figure 2.2 for an illustration of a scalable audio-video encoding.)

![Diagram of scalable audio-video presentation](image)

Figure 2.2 : Scalable audio-video presentation.

Fine-granular scalability (FGS) [61, 62] is one special case of scalable video coding, where there are two bit-stream layers, base and enhancement. In particular, video frames belonging to the base layer are encoded using standard predictive encoding techniques, while the corresponding frames of the enhancement layer are encoded non-predictively, without a temporal reference to other frames. This partially-embedded scheme is supported by the MPEG-4 video coding standard [63-65]. Bit-
streams of compressed enhancement layer frames can be truncated at any position, which allows for granular variation in reconstructed video quality in proportion to the number of decoded bits. However, due to the lack of temporal dependency in the enhancement layer the coding efficiency of the FGS scheme is significantly reduced (especially in low bit-rate regimes) relative to those of standard non-scalable (single layer) coding techniques, e.g., MPEG and H.26x.

In order to improve the coding efficiency of the "classical" FGS scheme, a further motion compensation (MC) based FGS technique (MC-FGS) was proposed [66-68]. In MC-FGS, high-quality reference frames (base + enhancement) are used for removing temporal redundancy in both base and enhancement layer frames. This brings the compression efficiency close to that of non-scalable coding. However, MC-FGS is sensitive to loss or corruption of data in the enhancement layer that can potentially occur during transmission. This in turn will introduce the well known error-propagation (or drift) phenomenon, due to the predictive nature of the encoding process, which can significantly degrade the reconstruction quality of the decoded bitstreams. Independently and around the same time, the authors in [69] have proposed another alternative framework, denoted progressive FGS (PFGS). Similarly to MC-FGS, a prediction loop is employed for both base and enhancement layer frames, however only partial temporal dependency is employed for encoding the enhancement layer frames. This improves the coding efficiency of PFGS relative to the original FGS framework. In addition, PFGS maintains a separate prediction dependency for the base layer frames that extends over several previous high quality (base + enhancement layer) frames, which alleviates the error-drift problem to a certain degree by making the encoded bit streams more robust to data losses or corruption.
In general, however, the rate-distortion performance of scalable representations is still inferior to that of non-scalable representations, particularly at low bit-rates. To compensate for this, some works have recently developed scalable video coding schemes with leaky prediction in the enhancement layers that can close much of the efficiency gap [70, 71].

Scalable video representations aid in TCP-friendly streaming, as they provide a convenient way for performing the rate control required to mitigate network congestion, see, e.g., [72–74]. In receiver-driven layered multicasting, video layers are sent in different multicast groups, and rate control is performed individually by each receiver by subscribing to the appropriate groups [75–77]. Layered video representations have further been proposed in combination with differentiated quality of service (DiffServ) [78] in the Internet, e.g., [79–84]. The idea is to transmit the more important layers with better, but more expensive, quality of service (QoS), and the less important layers with fewer or no QoS guarantees. Unfortunately, widespread deployment of DiffServ-capable routers is not expected anytime soon, but, as will be discussed in Sections 2.2 and 2.3, priority mechanisms can also be built without network support by employing suitable algorithms in the terminals.

**Rate Switching**

As explained in Section 2.1 transmission rate scalability can be elegantly achieved by streaming layered embedded representations that are decodable at different bitrates, with gracefully degrading quality. In practice, however, commercial streaming solutions perform transmission rate control without layered embedded representations, by employing multiple independent representations of the same video sequence encoded at different bit-rates, as implemented, e.g., in the widely used SureStream [85].
There, the major issue then is the dynamic assembly at the receiver of compressed bitstreams belonging to different independent representations without a mismatch error (denoted earlier as the error-drift). This approach is commonly known as bitstream switching [86] in the literature. Error-drift free switching between two bitstreams can be most easily performed at locations in the streams that correspond to compressed video frames that are encoded independently of other frames, i.e., non predictively, as done for example in [87]. These are the so called intra-coded (I) frames. In addition, specially designed switching frames, denoted S-frames [24,56] and SP-frames [25,88–92] have been proposed for the same purpose, however at the cost of lower compression efficiency.

In SP-frames, primary switching pictures (SP) are embedded into the encoded bitstream. They serve as anchor frames as only at these locations can switching from one bitstream to another be made. Then, when switching is actually performed at a particular location of a primary SP-frame, a corresponding secondary switching picture (SP1), that is not a regular part of either bitstream, is sent to the receiver. The SP1 frame replaces the primary SP-frame in the bitstream to which we switch and serves as a reference picture for the following frames in this bitstream. At the same time, the SP1 frame uses as a reference a picture in the bitstream from which we switch. In Figure 2.3, we show an illustration of stream switching via SP-frames for rate scalability. In particular, while streaming Stream 2 a secondary switching frame (SP1) is sent to switch from Stream 2 to Stream 1, as illustrated in Figure 2.3.

A related concept is the principle of spare pictures [93]. In this approach, whenever a reference picture (or part of it) is missing at the decoder, another picture (or part of it) is used as a replacement: hence the name spare picture. The selection
of an alternative reference among multiple possible choices is done based on resemblance with the original reference. Spare pictures are more efficient in terms of data rate than S- and SP-frames. However, they do not eliminate completely the drift problem, because the resemblance between a spare picture and the original reference picture is not perfect.

2.2 Error Control

Forward error correction (FEC) across packets can be employed to overcome packet losses [26, 29–31, 54, 94–96] in the case of transmission over packet erasure channels. As long as a sufficient number of packets is received, missing video packets can be recovered at the receiver with the help of received redundancy packets. Reed-Solomon codes [97–99] are particularly suitable for this application. With the Priority Encoding Transmission (PET) scheme proposed in [100], the redundancy packets can be
organized to provide unequal error protection (UEP) for different layers of a scalable video representation. For high packet loss rates, the important, strongly protected layers can still be recovered, while the less important layers might not be decodable. Thus, an effect similar to DiffServ is achieved, but without network support. (See Figure 2.4 for an illustration of an implementation of the PET scheme with Reed-Solomon codes.) Several publications have addressed the problem of how much redundancy should be sent and how it should be distributed across packets, e.g., [101–109]. The PET scheme has been combined with receiver-driven multicasting by sending parity packets in separate multicast groups [27, 28, 76, 77, 96, 110–112]. In the case of transmission over wireless channels, FEC across media bitstreams can be applied to account for potential corruption of transmitted bits. Rate Compatible Punctured Convolutional (RCPC) codes [113] are typically used as an alternative to Reed-Solomon codes for this scenario. As in the case of FEC across packets, the main issue here is how much redundancy should be sent and how it should be distributed across the transmitted bits [114–118]

Multiple description coding (MDC) has been proposed as an alternative to layered scalable coding (LC) for image and video transmission over unreliable channels [35, 36, 119–123]. Prior to its introduction for multimedia communication, all related work on MDC dealt mainly with signal processing and information-theoretic aspects of this coding technique, e.g., [124–133]. In MDC, each description alone can guarantee a basic level of reconstruction quality of the source, and every additional description can further improve that quality. There have been a few studies reported in the literature to date that examine the performance of LC and MDC for video streaming [134–137]. In our own work [138], we have shown that contrary to conventional wisdom layered scalable video representations are superior to multiple
Figure 2.4: Implementation of the PET scheme with Reed-Solomon codes. (Original figure provided via PowerPoint slides in April 2003, courtesy of P.A. Chou [1].)

description coding, when packet schedules are optimally coordinated. Finally, combining LC and MDC has been considered for video streaming, e.g., in [1, 139–141], in order to exploit the individual benefits and at the same time to avoid the individual shortcomings of these two source coding techniques.

Other techniques that can be employed for protection and resilience against loss of video packets include data interleaving [19, 142–147], which provides certain robustness to transmission introduced errors at the price of longer decoding delay, and dynamic switching of the encoding mode (Intra or Inter) for macroblocks of video frames, where certain macroblocks are intra-coded depending on the current network condition and in order to alleviate the effects of error propagation. An equivalent
technique to the latter one for dealing with error propagation is dynamic control of prediction dependencies of video frames, such as in reference picture selection (RPS) [23, 148–154] and NEWPRED in MPEG-4 [155], where receiver feedback is used to modify the reference picture for a video frame on the fly. A related approach is video redundancy coding (VRC) [150, 156], where a video sequence is first sub-sampled into $K$ complementary subsequences (or threads) of video frames in a round-robin fashion, which are then encoded independently. The instance of VRC for $K = 2$ corresponds to the scheme called multiple state encoding proposed in [33]. Finally, long-term memory (LTM) prediction can be used in conjunction with dynamic control of prediction dependencies, as in [22, 157–161], in order to improve the compression efficiency of the latter technique.

In practical streaming systems, the most common error control technique is Automatic Repeat reQuest (ARQ) where lost packets are recovered through retransmission [98, 162–166]. ARQ systems use combinations of time-outs and positive and negative acknowledgments to determine which packets should be retransmitted. They can allow for prioritized retransmission in the case of layered video. Unlike FEC schemes, ARQ automatically adapts to varying channel loss rates and, hence, tends to be more efficient for the Internet. On the other hand, like all feedback-based error control algorithms, it cannot be used for multicasting. For Internet video-on-demand, however, each client is individually served ("unicasting") and hence feedback can be used. Moreover, even for live streaming to large audiences, splitter servers are typically employed to provide a unicast to each client, rather than relying on the multicast mechanisms which might or might not be supported in the routers. The main problem then, for ARQ schemes, is to determine what packets to transmit, and when, in order to maximize the video quality at the receiver.
2.3 Rate-Distortion Optimized Packet Scheduling

One of the earliest publications that addresses in a rigorous manner the issue of scheduling transmissions and retransmissions of layered media representations is the 1998 technical report on Soft ARQ by Podolsky, McCanne, and Vetterli [38]. The authors use a Markov chain analysis to find the optimal policy for transmitting layered media at a fixed rate and minimum end-to-end distortion. The complexity of their algorithm grows exponentially with the number of packets considered for transmission. In another related paper, Miao and Ortega develop a low-complexity heuristic for sender-driven scheduling of media packets over a best-effort network [39]. In a follow-up work [40], the authors propose a fast adaptive algorithm for determining the schedule with significantly lower complexity than the original framework. A particularly intriguing framework for computing rate-distortion optimized packet scheduling policies has been developed, at around the same time, by Chou et al. [41, 42]. Chou’s framework overcomes the complexity problem encountered in [38] with iterative optimization and makes several additional seminal contributions. In the following, we discuss Chou’s packet scheduling framework in greater detail since we used it as a starting point for the research presented in this thesis.

In Chou’s framework, packets may be sent at regularly-spaced transmission opportunities. At each opportunity, the goal is to select the transmission actions that minimize the expected distortion of the decoded media stream while not exceeding a maximum transmission rate. Each packet $n$ is characterized by its size in bytes $B_n$, its decoding deadline $t_n$, and the amount of distortion $\Delta d_n$ that will be removed from the decoded stream if the packet is decoded. The dependencies among media packets are expressed in a Directed Acyclic Graph (DAG), and it is assumed that a packet can only be decoded if all of its ancestor packets in the DAG have been
successfully decoded.

Packet transmission delays, for both forward and backward channels, are modeled as random variables with a certain distribution. Packet loss is included in the delay distribution as a mass at infinite delay. With these delay characteristics and the DAG source characterization, the packet scheduler can calculate the expected distortion and transmission rate resulting from a sequence of transmission actions. In a sequence of actions, each packet may be sent once, several times, or not at all and it is acknowledged by the receiver upon receipt. Because the expected distortion resulting from a set of transmissions at a given opportunity will be affected by the transmissions made at subsequent opportunities, the packet scheduler must optimize the set of packets to be sent at the current transmission opportunity jointly with a plan of prospective future transmissions. At each time step, therefore, the framework optimizes entire policies governing the transmissions that will occur within a time horizon, contingent on acknowledgements. Policies are selected that minimize $J = D + \lambda R$, where $D$ is the expected distortion and $R$ is the expected transmission rate that is induced. The transmission policies are recomputed at each time step, taking into consideration a new time horizon, new packets eligible for transmission, and new acknowledgements received in the mean time.

In the original paper [41, 42], Chou and Miao report impressive performance gains for sender-driven audio streaming, due to the rate-distortion optimized packet scheduling. The flexibility of the framework has allowed its application to a number of streaming scenarios. In [167], Chou and Sehgal extend it to receiver-driven streaming. In [168–171], Chakareski and Chou apply it to streaming over hybrid wireline-wireless channels. In [172], Sehgal and Chou employ the framework for cost-distortion optimized caching of streaming media. In a related work [173], Seh-
gal and Chou explore cost-distortion optimized streaming over DiffServ networks and show that DiffServ yields no advantage over a network without differentiated services when delay constraints allow time for retransmissions. In [174–176], Kalman, Steinbach, and Girod extend the framework to Adaptive Media Playout (AMP), which enables a client to extend packet delivery deadlines and thus to improve the appearance of a reconstructed video. In [177, 178], Cheung and Yoshimura employ a simplified version of Chou’s framework to enhance the performance of streaming systems based on monitoring agents. Kalman, Ramanathan, and Girod extend Chou’s framework to streaming with multiple delivery deadlines in [179, 180], while Setton and Girod propose an approximation of the framework to study congestion-distortion optimized streaming over network bottleneck links [181]. Finally, in a related work [182], Chakareski and Frossard present an extension of the framework that incorporates multiple rate constraints for streaming applications with adaptive rate control.

2.4 Current Limitations of Rate-Distortion Optimized Packet Scheduling

Despite the significant progress that rate-distortion optimized packet scheduling has made recently, there remain important limitations that we address with the research presented in this thesis. For example, in the original framework [41, 42] packets that depend directly or indirectly on lost packets are discarded. Thus, there is no provision to account for error concealment techniques [4, 183] that are widely used with video streaming and allow continued decoding of dependent packets, even if some of their predecessors are lost. Zhang et al. [184] recently showed that Chou’s framework, in
fact, severely overestimates the effects of packet losses, if concealment is employed. Another limitation is the assumption that the distortion reductions associated with decodable packets are additive. After concealment, however, the distortion due to a lost packet typically depends on which other packets have been received. Therefore, more sophisticated distortion models are required. First attempts along these lines have been reported recently in the context of pixel-based error concealment [184–186]. In Chapter 3, we develop a distortion model that accounts for previous frame error-concealment, as it is widely used today in practical video streaming systems.

![Graph](image)

**Figure 2.5**: Delay of Internet packets sent in 30 ms intervals between Santa Clara, CA and Cambridge, MA. The trace shows that delays of successive Internet packets are often highly correlated.

Another limitation of the previous work is that the delays and losses of successive packets are modelled as statistically independent. However, in the Internet, packets usually arrive in the same order as they were transmitted and end-to-end delays of successive packets are strongly correlated (Figure 2.5). Losses often appear in bursts
that are correlated to increases in delay [187]. Therefore, in Chapter 3 we design a more accurate model that takes into consideration the statistical dependence of delay and loss among consecutive packets. In addition, rate-distortion optimized packet schedulers to date have assumed accurate knowledge of the statistical parameters of the packet loss and delay experienced on a network path. To address this issue, in Section 5.2 we design a simple channel estimation scheme based on active and passive feedback that can be incorporated within our advanced framework and that therefore relieves the framework from the channel knowledge assumption.

Finally, more complicated streaming media architectures, such as systems with packet diversity and systems with an intermediate proxy server, could especially benefit from rate-distortion optimized streaming, as a wider range of policies is available to enhance the end-to-end system performance. Exploring these various scenarios has only begun very recently. In the following, we elaborate on these scenarios in greater detail.

2.4.1 Diversity Techniques

Diversity techniques have been studied extensively in the context of wireless communication. They were introduced in order to exploit the large variability in terms of channel quality when multiple channels are considered for simultaneous transmission. The achieved diversity can be in time, frequency or space, or in any combination of these three [188].

A number of studies have shown that there is an analogous situation in Internet communication: in 30-80% of the cases there is an alternate path that performs significantly better than the default path between two hosts [189]. Performance is measured in terms of round-trip-time, loss rate and bandwidth. These studies have
motivated the introduction of packet path diversity for video streaming in [33], where the author proposes to send complementary descriptions of a multiple description (MD) coder through two different Internet paths. The presented experimental results showed the potential benefits of the proposed system.

Since then a number of studies have appeared that exploit the concept of packet diversity in media communication. In [34] the authors employ path diversity in the context of video communication using unbalanced MD coding to accommodate the fact that different paths might have different bandwidth constraints. The unbalanced descriptions are created by adjusting the frame rate of a description sent over a particular path. In [190, 191] a framework for real-time voice communication over the Internet is proposed, based on path diversity and MD coding. Significant improvement in performance is reported over existing schemes using a single network path transmission. In [192] the authors study image and video transmission in a multihop mobile radio network. It is shown that combining MD coding and multiple path transport in such a setting provides higher bandwidth and robustness to end-to-end connections. In a related work [193], the authors study multipath transmission of video encoded into multiple independent streams for applications in ad hoc networks. Additional examples of works where the authors have exploited path diversity to provide improved performance in sender-driven streaming can be found in [194, 195].

In [196] a receiver-driven framework for distributed streaming of video from multiple senders to a single receiver is proposed. The authors introduce packet partition and rate allocation algorithms to minimize the effects of packet erasures and late packet loss. The algorithms are incorporated within a TCP-friendly protocol that allows for smooth sending rates at the senders in order to reduce network delay jitter.
It is shown that the proposed framework provides lower distortion relative to conventional non-adaptive distributed video streaming. In a follow up work [197, 198], the authors incorporate Forward Error Correction (FEC) codes into the framework to account for lost packets as retransmissions are not allowed. Similarly, the works in [195, 199] consider receiver-driven control protocols that synchronize the senders’ transmissions in a rate-distortion optimized way. For improved error-resilience, MD coding is employed at each sender to pre-encode (prior to transmission) a progressively encoded media content that is streamed afterwards to the client.

In [153] a framework for video transmission over the Internet is presented, based on path diversity and rate-distortion optimized reference picture selection. Here, based on feedback the packet dependency is adapted to channel conditions in order to minimize the distortion at the receiving end, while taking advantage of path diversity. In a related work [200], the authors employ feedback and channel probing within an MD coding framework, to determine which network path should be used for transmission and to adapt the source encoding of the video in order to mitigate error propagation effects. In [37, 201] the performance of path diversity and multiple description coding in Content Delivery Networks (CDN) is studied. 20-40% reduction in distortion is reported over conventional CDNs for the network conditions and topologies under consideration. In [136] the authors examine the performances of MD coding and layered coding as two prospective techniques for video coding in the context of path diversity in multihop wireless networks. Finally, the work in [202] exploits path diversity to provide low latency wireless video over 802.11b networks.

Despite the substantial amount of previous work, as elaborated above, streaming with diversity has not yet been addressed in a rate-distortion optimized way. In other words, in all cases, heuristics are used for choosing the best path/server for
streaming each packet, rather than rate-distortion optimized scheduling/routing. In Section 4.1, we formulate a framework that addresses the scenario of streaming with diversity in a rate-distortion optimized way.

2.4.2 Proxy Caching and Monitoring Agents

Proxy caching and monitoring agents are related concepts in media streaming, where the ability to store and to locate information at the edge of the backbone network is exploited to provide improved performance. In proxy caching [172, 203–209], proxy servers temporarily store the most frequently accessed content from a central media server. The cached content is served afterwards from the proxy servers to newly connecting clients. Proxy caching is in fact an instance of the so called edge architecture approach, where several servers are strategically placed around the Internet so that each client can choose the server that results in shortest round-trip time and least amount of congestion. The edge servers are typically placed at junction points of two or more heterogeneous networks. Many content delivery network (CDN) companies, such as Akamai [210] and Speedera Networks [211], use this technology to achieve better load balancing and higher throughput, while reducing access latency and bandwidth requirements. Proxy servers can also be successfully used as video transcoders to provide the necessary rate scaling between two or more heterogenous networks [212–217]. In such scenarios, instead of signaling back to the communication source, bandwidth bottleneck problems are dynamically resolved by the proxy during media transmission. In addition to rate scaling, some video transcoding proxies also provide error resilience support as a more rapid and dynamic way of error handling at the edge of different networks [218, 219]. Furthermore, in [220, 221] a scheme is proposed that combines stream scheduling at an origin server and pre-
fix/partial caching at an intermediate proxy to minimize the aggregate transmission rate over the backbone network under a cache capacity constraint. It is shown through simulations that the proposed scheme provides significant improvement in performance relative to a full caching technique. A related work [222] examines the trade-offs between prefix/partial caching and full caching in the presence of constraints on the backbone network bandwidth and the available cache space. The authors propose streaming strategies that maximize the revenue rate of the service provider given these constraints.

Monitoring agents are network proxies located at the edge of the backbone network that send information back to the media server. Using this knowledge the server can better estimate the network state and adapt the media content sent to the client. Statistical feedback from monitoring agents was used in [223–225] while both statistical and timely feedback was used in [177, 178]. Statistical feedback, e.g., conventional RTCP reports [226], contains information such as the mean packet loss rate and the mean and variance of the round-trip time (RTT) over a window of packets. Timely feedback, such as positive and/or negative acknowledgements, indicates individual packets that have or have not arrived at the monitoring agent correctly and on time. In a related body of works [227–230], a network proxy in conjunction with timely feedback is employed to improve TCP performance over wireless networks. The presented simulation results show improved performance in terms of average throughput and link utilization. In addition, the proposed techniques allow for a significant reduction of the initial buffering delay and the client buffer size.

Motivated by the benefits provided by proxy caching and monitoring agents, as discussed above, we formulate in Section 4.2 a framework for rate-distortion optimized proxy-driven streaming from the network edge.
Chapter 3

Rate-Distortion Optimization (RaDiO) Framework

The advanced framework for Rate-Distortion Optimized (RaDiO) streaming optimizes the transmission process between a sender and a receiver using a rate-distortion criterion end-to-end. That is, the framework delivers the best possible quality to the receiver (end user), regardless of what the quality of the communication channels on the network path(s) is and regardless of how many bytes we are allowed to transmit over them. In order to solve this problem, our framework employs abstractions, or models, of three things: the source of data units, the channel or the network path, and the transmission protocol or algorithm. In the following, we describe in detail each of these models. It should be mentioned that in the exposition in the present chapter and in the next chapter, we try to be consistent with the terminology and notation used in prior works on rate-distortion optimized streaming, so that hopefully it creates less confusion for the reader and makes the comparison across different works easier.

3.1 Source Characterization

In a streaming media system, the encoded data are packetized into data units and are stored in a file on a media server. The (typically) predictive nature of the

---

Publications related to this chapter are [231,232].
encoding process imposes decoding dependencies between the various data units in
the presentation, which can be expressed by a directed acyclic graph. In particular,
if each node of the graph corresponds to a data unit, then an edge of the graph
directed from data unit \( l' \) to data unit \( l \) implies that data unit \( l' \) can be decoded
only if data unit \( l \) is first decoded. In other words, the graph simply imposes a
decoding order on the data units in the presentation*.

Associated with each data unit \( l \) is a size \( B_l \), and a decoding time \( t_{DTS,l} \). Specifically, \( B_l \) is the size of the data unit in bytes. \( t_{DTS,l} \) represents the time at which
the decoder is scheduled to extract the data unit from its input buffer and decode it.
(This is the decoder time-stamp in MPEG terminology.) Therefore, in the context
of streaming \( t_{DTS,l} \) is the delivery deadline by which data unit \( l \) must arrive at the
client, or be too late to be usefully decoded. Packets containing data units that arrive
after the data units' delivery deadlines are discarded†. For clarity purposes, in our
analysis we assume that every media unit, whether a video frame or an audio frame,
from the encoded media presentation is packetized into one data unit. Nonetheless,
our analysis also applies to the more general case, when an encoded media unit may
be packetized into multiple data units.

**Distortion Modeling**

Any rate-distortion optimized packet scheduler depends critically on an accurate
prediction of the video quality that results at the client, if some of the transmitted
packets carrying data units arrive late and others are lost or omitted (i.e., these data

*In the case of non-predictive encoding, each of the data units can be decoded independently.
†Recently, some works [179,180] have considered keeping late packets since they can be still
useful for accelerated retroactive decoding of future descendant packets.
units are not transmitted). If the importance of some data units is overestimated, while the importance of others is underestimated, the overall video quality suffers, since the scheduler optimizes the transmission policy based on flawed assumptions. In the following, we describe in detail the proposed distortion model.

To account for the fact that lost or late data units are concealed by the decoder at the receiver\(^4\) with other decoded data units that have arrived on time, we assign to each data unit \(l\) a concealment set \(\mathcal{N}_c\) as follows. \(\mathcal{N}_c^{(l)} = \{\emptyset, 1, \ldots, l\}\) is the set of data units that the receiver considers for error concealment in case data unit \(l\) is not decodable\(^5\) by the receiver on time. The decoder always prefers the most recent decodable data unit from the concealment set, i.e., for \(l_1, l_2 \in \mathcal{N}_c^{(l)}: l_1 > l_2\) and both \(l_1, l_2\) decodable on time the decoder always chooses \(l_1\) to conceal the missing data unit \(l\). This concealment strategy is denoted in the video coding community as previous frame error concealment, since a missing frame is replaced with the most recent decoded frame from the past.

Note that the concealment case \(\mathcal{N}_c^{(l)}(l + 1) = l\) simply describes the outcome when data unit \(l\) itself is decodable on time, i.e., no error concealment is performed. Furthermore, the first entry in the concealment set, \(\mathcal{N}_c^{(l)}(1) = \emptyset\), accounts for the fact that sometimes there might not be prior data units that are available at the decoder to conceal the loss of data unit \(l\). In that case, the decoder conceals the missing data with gray level pixel values. It should be noted here that error concealment is only performed for missing video data.

For each concealment event in \(\mathcal{N}_c^{(l)}\) there is a corresponding reduction in recon-

---

\(^4\)After decoding the data units that have arrived on time.

\(^5\)A data unit \(l\) is decodable if all of its ancestor data units are received on time, i.e., prior to their respective delivery deadlines.
struction error (distortion) $\Delta d_{l}^{(l_1)}$ for the media presentation if data unit $l$ is not decodable and is concealed with data unit $l_1 \in \mathcal{N}_c^{(l)}$ that is received and decoded on time. Thus, in our model we have multiple distortion reductions $\Delta d_{l}^{(l_1)}$ associated with a single data unit $l$ in order to capture the effect that the distortion reduction associated with a missing data unit is dependent on what other data units have been received and decoded on time. This is in contrast to all prior works on rate-distortion optimized streaming, including Chou’s framework [41, 42], where there was a single distortion reduction value $\Delta d_{l}$ associated with each data unit $l$. Specifically, in these works if a data unit $l$ is decodable on time, then it contributes with a $\Delta d_{l}$ to the reduction in reconstruction error (distortion) for the media presentation. Otherwise, the distortion reduction due to data unit $l$ is simply zero.

Our model can be easily generalized to the case when both future and past data units, relative to data unit $l$, that are received and decoded on time are considered for concealment of the missing data unit $l$. That can be done by enlarging the concealment set $\mathcal{N}_c^{(l)}$ appropriately such that it includes future data units. Certainly, in that case distortion reduction values $\Delta d_{l}^{(l_1)}$ must also be associated for the events when data unit $l$ is concealed with another data unit $l_1 \in \mathcal{N}_c^{(l)}$ for $l_1 > l$. Furthermore, even more sophisticated techniques for error concealment proposed in the past, such as bidirectional interpolation [233–235], where a missing frame is synthesized using past and future decoded frames, relative to the missing frame can be incorporated into our model by appropriately expanding the concealment set and by generating the corresponding distortion reduction values. Nonetheless, we de-

---

It should be mentioned that later on in the same year when our framework was initially published [231], Kalman, Ramanathan, and Girod [179,180] independently proposed an equivalent approach that also accounts for previous frame copy concealment.
cided not to consider any of these additional techniques in our model in order to limit its complexity. Additional argument for our decision was the fact that previous frame error concealment is the most widely employed and probably the only error concealment technique used in practical video communication systems today.

Lastly, note that the proposed distortion model can be obtained as auxiliary information when the video sequence is compressed. Along with packet sizes and decoding deadlines, this information forms the rate-distortion preamble, a compact description of packet contents. To compute an optimized transmission schedule, a scheduling algorithm only considers the rate-distortion preamble, rather than the actual compressed video sequence.

3.2 Channel Characterization

Another critical component of a rate-distortion optimized packet scheduler is an accurate model of packet loss and delay. The scheduler must determine the probability of each packet loss pattern. Losses can be due, e.g., to packet dropping in routers, to adverse wireless access link conditions, or to arrivals after the deadline.

We model the forward and backward channels on a network path between a sender and a receiver as burst-loss channels using a $K$-state discrete-time Hidden Markov model. The forward and the backward channel make state transitions independently of each other every $T$ seconds, where the transitions are described by probability matrices $\mathcal{P}_{(F)}$ and $\mathcal{P}_{(B)}$, respectively. In the following, we derive the packet loss and delay probabilities that are experienced on these channels.
Packet Loss and Delay Probabilities

In each state the forward and the backward channel are characterized as independent time-invariant packet erasure channels with random delay. Hence, they are completely specified by the probabilities of packet loss \( \epsilon_F^k \) and \( \epsilon_B^k \), respectively, and the probability densities of the transmission delay \( p_F^k \) and \( p_B^k \), respectively, for \( k = 1, \ldots, K \). This means that if the sender sends a packet on the forward channel at time \( t \), given that the forward channel is in state \( k \) at \( t \), then the packet is lost with probability \( \epsilon_F^k \). However, if the packet is not lost, then it arrives at the receiver at time \( t' \), where the forward trip time \( FTT^k = t' - t \) is randomly drawn according to the probability density \( p_F^k \). Therefore, we let \( P\{FTT^k > \tau\} = \epsilon_F^k + (1 - \epsilon_F^k) \int_\tau^\infty p_F^k(t)dt \) denote the probability that a packet transmitted by the sender at time \( t \), given that the forward channel is in state \( k \) at \( t \), does not arrive at the receiver's application by time \( t + \tau \), whether it is lost in the network or simply delayed by more than \( \tau \). Then similarly, \( P\{BTT^k > \tau\} = \epsilon_B^k + (1 - \epsilon_B^k) \int_\tau^\infty p_B^k(t)dt \) denotes the probability that a packet transmitted by the receiver at time \( t \), given that the backward channel is in state \( k \) at \( t \), does not arrive at the sender by time \( t + \tau \), whether it is lost in the network or simply delayed by more than \( \tau \). Finally, we are interested in \( P\{RTT^{kj} > \tau\} \), which is the probability that the sender does not receive an acknowledgement by time \( t + \tau \) for a packet transmitted at time \( t \), given that the forward and the backward channel are respectively in states \( k \) and \( j \), at \( t \).

To derive \( P\{RTT^{kj} > \tau\} \) assume first that the transmission on the forward channel occurred immediately after the channel made a state transition. If \( FTT^k \leq T \), the packet is received by the receiver before the backward channel makes the next state transition. Then \( P\{RTT^{kj} > \tau|FTT^k \leq T\} = P\{FTT^k + BTT^j > \tau|FTT^k \leq T\} \) as the receiver sends an acknowledgement while the backward channel is still in
the current state \( j \). The probability of this event is \( P\{FTT^k \leq T\} \). However, if \( lT < FTT^k \leq (l + 1)T \), for \( l \geq 1 \), then the state of the backward channel makes \( l \) transitions before the packet actually arrives at the receiver. The probability of this event is \( P\{lT < FTT^k \leq (l + 1)T\} \). Here the state on the backward channel when the acknowledgement is sent can be any of the \( K \) possible values. Hence we compute the desired quantity as the expected value over all of them, i.e., \( P\{RTT^{kj} > \tau | lT < FTT^k \leq (l + 1)T\} = \sum_{p=1}^{K} P_{j,p}^{(l)} \sum_{p=1}^{K} P\{FTT^k + BTT^p > \tau | lT < FTT^k \leq (l + 1)T\} \). Note that \( P_{j,p}^{(l)} \) is the probability of making a transition from state \( j \) to state \( p \) in \( l \) transition intervals. These probabilities are obtained using matrix powers, i.e., \( P_{(B)}^{(l)} = P_{(B)}^{l} \). Finally, by averaging over all possible outcomes for \( FTT^k \) we write

\[
P\{RTT^{kj} > \tau \} = \sum_{l=0}^{\infty} \sum_{p=1}^{K} P_{j,p}^{(l)} \sum_{p=1}^{K} P\{lT < FTT^k \leq (l + 1)T, FTT^k + BTT^p > \tau \}
\]

\[
= \sum_{l=0}^{M-1} \sum_{p=1}^{K} P_{j,p}^{(l)} \sum_{p=1}^{K} P\{lT < FTT^k \leq (l + 1)T, FTT^k + BTT^p > \tau \}
\]

\[
+ \sum_{p=1}^{K} P_{j,p}^{(M)} \sum_{p=1}^{K} P\{MT < FTT^k \leq \tau, FTT^k + BTT^p > \tau \}
\]

\[
+ P\{FTT^k > \tau \}, \tag{3.1}
\]

where the first equality follows from Bayes' rule [236], while the second one holds since \( P\{lT < FTT^k \leq (l + 1)T, FTT^k + BTT^p > \tau \} = P\{lT < FTT^k \leq (l + 1)T\} \), for \( lT \geq \tau \). Finally, \( M = \lceil \tau/T \rceil \) and \( P_{(B)}^{(0)} = I \), the identity matrix.

In a receiver-driven transmission the receiver sends request packets to the sender, requesting transmission of particular media packets. The sender transmits a media packet only when it receives a request for it. Therefore, for this scenario \( P\{RTT^{kj} > \tau \} \) is the probability of the event that the receiver sends a request on the backward channel at time \( t \), but the requested media packet does not arrive on the forward channel by time \( t + \tau \), given that the forward and the backward channel are re-
respectively in states $j$ and $k$, at $t$. Building on (3.1), it is easy to see that we can write

$$P\{RTT^{kj} > \tau\} = \sum_{l=0}^{M-1} \sum_{p=1}^{K} P^{(l)}_{jp(f)} P\{lT < BTT^k \leq (l+1)T, FFT^p + BTT^k > \tau\}$$

$$+ \sum_{p=1}^{K} P^{(M)}_{jp(f)} P\{MT < BTT^k \leq \tau, FFT^p + BTT^k > \tau\}$$

$$+ P\{BTT^k > \tau\} \quad (3.2)$$

### 3.3 Rate-Distortion Optimization using Iterative Sensitivity Adjustments

Suppose there are $L$ data units in the media presentation. Let $\pi_l \in \Pi$ be the transmission policy for data unit $l \in \{1, \ldots, L\}$ and let $\pi = (\pi_1, \ldots, \pi_L)$ be the vector of transmission policies for all $L$ data units. $\Pi$ is a family of policies that is defined precisely depending on the transmission scenario under investigation. For now, it suffices $\Pi$ to remain abstract. Finally, let $M$ be the number of network paths over which transmission of data units is anticipated. For example, in the scenario of streaming with diversity, $M$ is simply the number of different network paths over which data units are transmitted.

Any given policy vector $\pi$ induces, for the media presentation, an expected distortion $D(\pi)$ and expected transmission rates, $R_m(\pi)$, for $m = 1, \ldots, M$, on the $M$ network paths, respectively, in the forward directions. We seek the policy vector $\pi$ that minimizes $D(\pi)$ subject to a constraint on a cost-weighted sum of the expected transmission rates, $R(\pi) = \sum_{m=1}^{M} \gamma_m R_m(\pi)$, where $\gamma_m$ represents the relative\(^\dagger\) cost of transmitting over network path $m$. The optimal policy vector $\pi$ can be found by

\(^\dagger\)Number of cost units per transmitted source byte.
minimizing the Lagrangian $D(\pi) + \lambda R(\pi)$ for some Lagrange multiplier $\lambda > 0$, thus achieving a point on the lower convex hull of the set of all achievable $(D, R)$ pairs.

We now compute expressions for $R(\pi)$ and $D(\pi)$. The expected transmission rate $R(\pi)$ is the sum of the expected number of bytes transmitted for each data unit $l \in \{1, \ldots, L\}$, $R(\pi) = \sum_l B_l \rho(\pi_l)$, where $B_l$ is the number of bytes of data unit $l$ and $\rho(\pi_l) = \sum_{m=1}^{M} \gamma_m \rho_m(\pi_l)$ is the expected cost of bytes transmitted over all network paths per source byte (under policy $\pi_l$), simply called the expected cost. The expected distortion $D(\pi)$ is somewhat more complicated to express, but still it can be expressed in terms of the probability $\epsilon(\pi_l)$ that data unit $l$ does not arrive at the receiver on time (under policy $\pi_l$), called the expected error.

Specifically, let data unit $l$ be not decodable on time at the receiver and let the decoder employ data unit $l_1 \in \mathcal{N}^l_c$ to conceal the loss of $l$, as described in Section 3.1. The probability of the event data unit $l_1$ is decodable on time is $\prod_{j \in A(l_1)} (1 - \epsilon(\pi_j))$, where $A(l_1)$ is the set of ancestors of $l_1$, including $l_1$. Note that to obtain this probability we have used the assumption that packet transmission processes are independent across data units. However, we do not assume this for the transmission processes of different packets associated with a single data unit, as it will become evident in the next chapter when we present the expressions for computing the expected error-cost for a policy $\pi_l$. Now given that $l_1$ is decodable, the probability of the event none of the data units $j \in \mathcal{N}^l_c : j > l_1$ is decodable on time is

$$\prod_{l_2 \in C(l,l_1)} \left( 1 - \prod_{l_3 \in A(l_2) \setminus A(l_1)} (1 - \epsilon(\pi_{l_3})) \right)$$

where $C(l, l_1)$ is the set of data units $j \in \mathcal{N}^l_c : j > l_1$ that are not mutual descendants

**Since $\lambda \gamma_m$ may be regarded as separate Lagrange multipliers, this also achieves a point on the lower convex hull of the set of all achievable $(D, R_1, R_2, \ldots, R_M)$ $(M+1)$-tuples.**
of each other, i.e., for any two data units \( j \) and \( k \) in \( C(l, l_1) \) it holds \( j \notin D(k) \) and \( k \notin D(j) \), where \( D(j) \) is the set of descendants of data unit \( j \)\(^{11}\).

The product term within the brackets is the probability that all of the ancestors of \( l_2 \in C(l, l_1) \), excluding those common with \( l_1 \), are received on time, where "\( \setminus \)" denotes the operator "set difference". Note that we only need to account for the event that data units from \( C(l, l_1) \) are not decodable, as all the other data units from the concealment set are descendants of at least one data unit from \( C(l, l_1) \) and hence will definitely not be decodable on time. Finally, remember that the case when data unit \( l \) is decodable is included using this notation as \( l \) is the last data unit in \( N_c^l \).

We use these results first to take an expectation over all possible cases of concealment for data unit \( l \) and then to sum over all data units in order to obtain

\[
D(\pi) = D_0 - \sum_l \Delta d_i^{(l_1)} \prod_{j \in A(l_1)} (1 - \epsilon(\pi_j)) \\
\times \prod_{l_2 \in C(l, l_1)} \left(1 - \prod_{l_3 \in A(l_2) \setminus A(l_1)} (1 - \epsilon(\pi_{l_3}))\right),
\]

(3.3)

where \( D_0 \) is the expected reconstruction error for the presentation if no data units are received.

Finding a policy vector \( \pi \) that minimizes the expected Lagrangian \( J(\pi) = D(\pi) + \lambda R(\pi) \), for \( \lambda > 0 \), is difficult since the terms involving the individual policies \( \pi_l \) in \( J(\pi) \) are not independent. Therefore, we employ an iterative descent algorithm, called Iterative Sensitivity Adjustment (ISA), in which we minimize the objective function \( J(\pi_1, \ldots, \pi_L) \) one variable at a time while keeping the other variables constant, until convergence [41]. Specifically, let \( \pi^{(0)} \) be any initial policy vector and let \( \pi^{(n)} = (\pi_1^{(n)}, \ldots, \pi_L^{(n)}) \) be determined for \( n = 1, 2, \ldots \), as follows. We select

\(^{11}\)Frequently, the set \( C(l, l_1) \) may only consist of a single data unit.
one component $l_n \in \{1, \ldots, L\}$ to optimize at step $n$ in a round-robin fashion, i.e., $l_n = (n \mod L)$. Then for $l \neq l_n$, we keep $\pi_i^{(n)} = \pi_i^{(n-1)}$, while for $l = l_n$, we compute

$$
\pi_i^{(n)} = \arg\min_{\pi_i} J(\pi_i^{(n)}, \ldots, \pi_{i-1}^{(n)}, \pi_i, \pi_{i+1}^{(n)}, \ldots, \pi_L^{(n)}) = \arg\min_{\pi_i} S_i^{(n)} \epsilon(\pi_i) + \lambda B_i \rho(\pi_i),
$$

where the second equality follows by grouping terms that do not depend on $\pi_i$ and where

$$
S_i^{(n)} = \sum_{l_1, l_2 \in \mathcal{C}_i^l} S_{l_1, l_2}^{(n)} = S_{l_1, l_2}^{+} - S_{l_1, l_2}^{-}
$$

(3.5)

can be regarded as the sensitivity to losing data unit $l$, i.e., the amount by which the expected distortion will increase if data unit $l$ cannot be recovered at the client, given the current transmission policies for the other data units. Convergence of the ISA algorithm is guaranteed because $J(\pi^{(n)})$ is non-increasing with $n$ and is bounded from below with zero, since it is non-negative.

Note that in contrast to [41, 42] where there is only one sensitivity factor, the sensitivity here consists of two nonnegative terms $S_i^{+}$ and $S_i^{-}$. The first term increases the sensitivity associated with data unit $l$ in case $l$ is in the ancestor set of data unit $l_2$ used for concealment of another data unit $l_1$. On the other hand, the second term reduces the sensitivity associated with $l$ in case $l$ is not in the ancestor set of $l_2$. This result is intuitive and allows us to model appropriately the situations where data unit $l$ is irrelevant for concealment of another data unit. Expressions for
$S_{l;1}^{+(n)}$ and $S_{l;1}^{-(n)}$ are easily obtained from (3.3) by grouping terms, i.e.,

\[
S_{l;1}^{+(n)} = \sum_{l_2 \in \mathcal{N}_{1,l}^2} \Delta d_{l_1}^{(l_2)} \prod_{j \in A(l_2) \setminus \{j \neq l\}} (1 - \epsilon(\pi_j^{(n)})) \\
\times \prod_{l_3 \in C(l_1,l_2)} \left( 1 - \prod_{l_4 \in A(l_3) \setminus A(l_2)} (1 - \epsilon(\pi_{l_4}^{(n)})) \right),
\]

(3.6)

\[
S_{l;1}^{-(n)} = \sum_{l \in A(C(l_1,l_2)) \setminus A(l_2)} \Delta d_{l_1}^{(l_2)} \prod_{j \in A(l_2)} (1 - \epsilon(\pi_j^{(n)})) \prod_{l_3 \in C(l_1,l_2) \setminus A(l_2), l_3 \neq l} (1 - \epsilon(\pi_{l_3}^{(n)})) \\
\times \prod_{l_3 \in C(l_1,l_2) \setminus A(l_2)} \left( 1 - \prod_{l_4 \in A(l_3) \setminus A(l_2)} (1 - \epsilon(\pi_{l_4}^{(n)})) \right).
\]

(3.7)

The minimization (3.4) is now simple, since each data unit $l$ can be considered in isolation. Indeed the optimal transmission policy $\pi_l \in \Pi$ for data unit $l$ minimizes the “per data unit” Lagrangian $\epsilon(\pi_l) + \lambda' \rho(\pi_l)$, where $\lambda' = \lambda B_l / S_{l;1}^{(n)}$. Thus to minimize (3.4) for any $l$ and $\lambda'$, it suffices to know the lower convex hull $\epsilon(\rho) = \min_{\pi \in \Pi} \{\epsilon(\pi) : \rho(\pi) \leq \rho\}$ of the function, which we call the expected error-cost function. The error-cost function can be considered as a normalized distortion-rate function pertaining to the transmission of a single dimensionless data unit, and it depends only on the transmission scenario and the channel characteristics.

In the following chapter, we show how to find the optimal policy $\pi^* = \arg \min_{\pi \in \Pi} \epsilon(\pi) + \lambda' \rho(\pi)$ for the families of transmission policies $\Pi$ corresponding to three advanced streaming scenarios. Specifically, in Section 4.1, we show how the optimal policy can be computed for the scenario of streaming with diversity, while in Section 4.2 we do the same for the scenario of proxy-driven streaming from the network edge. Finally, in Section 4.3, we compute the optimal policy $\pi^*$ for the unconventional scenario of streaming with rich acknowledgements.

Recall that during the development of the expression for the expected distortion
$D(\pi)$ we mentioned that we used the assumption that packet transmission processes are independent for different data units. That was done in order to factor the expectation across the products in (3.3) as otherwise the derivation of $D(\pi)$ becomes intractable even for a small number of data units. Naturally, this assumption leads to an approximation of the actual $D(\pi)$, and in the following we try to quantify the potential effects of the approximation.

Now, since evaluating the expected distortion without the independence assumption is computationally intractable, instead we compute the performance loss of the optimization framework from this chapter relative to an ideal streaming system that is described below. That will give us a bound on the potential loss in performance due to the approximation, as the rate-distortion performance of the optimization framework without the independence assumption will always be below that of the ideal system, and at least equal to (if not above) the one for the case when the independence assumption is in place.

In particular, we examine the performance of the optimization framework for sender-driven streaming of packetized video content over a network path where the communication channels exhibit memory, i.e., dependence between subsequent packet transmissions. The video content employed in these experiments are the standard test video sequences *Foreman, Mother & Daughter, Carphone, Container, News*, and *Salesman* that are used throughout the experiments in Chapter 5. Distortion is measured in terms of the average luminance PSNR of the decoded video frames at the receiver, as a function of the average transmission rate between the sender and the receiver. The forward and backward channels on the network path are characterized using the Hidden Markov Model from Section 3.2 for $K = 2$ states. One of the states is "good", meaning it exhibits short delays and low loss rate for transmitted
packets. On the other hand, the second state is "bad", as it features comparatively longer packet delays and higher packet loss rate. It is important to note that the play-out delay for the video presentation is selected such that it is many times larger than the mean round-trip delay experienced by packets during "bad" states on both channels.

We compare the performance of the optimization framework, denoted henceforth as $RadIo$, with that of an ideal rate-distortion optimal sender-driven system, denoted $Ideal\ R-D$. In particular, the performance of $Ideal\ R-D$ is obtained using the rate-distortion function of the source and the packet loss probabilities of the communication channels as follows. For $Ideal\ R-D$, we assume that the play-out deadline is infinitely large. This renders the effect of random packet delay irrelevant, as $Ideal\ R-D$ can always wait infinitely long before making a retransmission of an unacknowledged packet. Hence, the communications channels in the forward and backward directions act effectively as packet erasure channels with loss probabilities $\epsilon_F$ and $\epsilon_B$, respectively.

The loss probability $\epsilon_F$ is equal to the packet loss probability during a "good" state on the forward channel, while $\epsilon_B$ is equal to the mean value of the packet loss probabilities on the backward channel over the two states. Since the play-out deadline is infinite, $Ideal\ R-D$ can always wait long enough in order to transmit all of its packets exclusively during a "good" state, hence the selected value of $\epsilon_F$. Furthermore, since an arriving packet at the receiver is immediately acknowledged, $\epsilon_B$ can only be selected as the statistical average of the packet loss probabilities over all states on the backward channel.

Now, given the infinite play-out delay $Ideal\ R-D$ transmits close to channel capacity. As is well known [237], the capacity of an erasure channel with erasure proba-
bility $\epsilon$ is $1 - \epsilon$. Therefore, an optimal system transmits $1/(1 - \epsilon)$ channel packets for every data unit. In particular, since Ideal R-D continues to transmit packets until it receives an acknowledgement, which it receives with probability $(1 - \epsilon_F)(1 - \epsilon_B)$ for each transmitted packet, Ideal R-D transmits on average $1/[(1 - \epsilon_F)(1 - \epsilon_B)]$ packets per data unit. Hence, the performance of Ideal R-D is obtained by multiplying the rate values from the rate-distortion function of the Foreman sequence with the factor $1/[(1 - \epsilon_F)(1 - \epsilon_B)]$.

For a given set of channel parameters, we record the rate-distortion performances of the two systems under examination. Then, we measure the maximum increase in transmission rate of RaDiO relative to Ideal R-D over the whole range of distortion values achieved by the two systems. We denote this quantity $\Delta R$ and we express it in percents of the corresponding transmission rate of Ideal R-D. Finally, we repeat the experiment for different sets of channel parameters, where we vary the packet loss rate experienced on the communication channels during "bad" states, while keeping the rest of the parameters fixed.

In Figure 3.1, we show this excess rate of RaDiO as a function of the average packet loss rate experienced on the communication channels, denoted as $\epsilon_A$. Precisely, $\epsilon_A$ is defined as the statistical average of the packet loss rates associated with the two possible states on a channel. Due to the particular choice of the play-out delay for the video presentation described earlier, it is the packet loss rate that is the dominant factor affecting the performance of the optimization framework. Therefore, in this set of experiments we decided to measure the performance of RaDiO as a function of $\epsilon_A$ as this quantity provides us with a measure of the average packet loss rate experienced during a streaming session.

Firstly, it can be seen from the graphs shown in Figure 3.1 that over all sequences
Figure 3.1: Excess transmission rate $\Delta R$ (%) vs. Average packet loss rate $\epsilon_A$ (%) for streaming (top left) Foreman, (top middle) Mother & Daughter, (top right) Carphone, (bottom left) Container, (bottom middle) News, and (bottom right) Salesman.

The performance loss of RaDiO relative to Ideal R-D increases as the average packet loss rate increases. This is expected, since the optimization framework increasingly triggers retransmissions of data units due to unacknowledged previous transmissions which in turn are caused by lost packets or lost acknowledgements. However, the rate of increase and the absolute values of $\Delta R$ vary over the individual sequences, as also seen from Figure 3.1. In particular, the excess rate $\Delta R$ is more pronounced for Foreman (top left) and Carphone (top right) relative to the other four sequences. This is due to the fact that the later sequences exhibit less motion and scene complexity, which makes error concealment work more successfully in terms of accounting for the effect of missing data units at the receiver on the resulting video quality.

Secondly, as shown in Figure 3.1 the performance loss of RaDiO relative to the
ideal system does not exceed 4% over all six sequences for average packet loss rates below 10%. This means that the performance degradation of the optimization framework due to the independence assumption cannot be greater than this quantity. Similarly, for average packet loss rate in the range of 10%-20% the relative performance loss reaches up to 11%, as also shown in Figure 3.1 for the case of Foreman (top left). Therefore, by the same analogy this implies that the potential performance reduction of RaDiO that may be incurred by employing the independence assumption in (3.3) is bounded by 11% in this range. Finally, as $\epsilon_A$ is increased beyond 20%, the excess rate $\Delta R$ increases as well and reaches close to 20% and 8% in the case of Foreman and Container, respectively.

Next, we examine what the performance loss of RaDiO is relative to Ideal as a function of the play-out delay of the video application running at the receiver. In particular, in this set of experiments we vary the play-out delay relative to the mean mean round-trip delay experienced by packets during "bad" states on both channels, while keeping the packet loss rate fixed. Figure 3.2 answers the question how $\Delta R$ varies for the individual sequences, as a function of the ratio play-out delay / mean round-trip time (RTT). The average packet loss rate has been fixed to 9.4% in these experiments.

It can be seen that over all sequences the performance loss monotonically decreases as the play-out delay increases. This is expected since larger values of the play-out delay provide RaDiO with longer periods of time over which it can wait to receive acknowledgements for outstanding video packets. This in turn prevents unnecessary retransmissions of such packets, which increasingly occur as the play-out delay is reduced thereby causing a further degradation in performance of RaDiO relative to Ideal. For example, when the ratio play-out delay /mean RTT is two,
Figure 3.2: Excess transmission rate $\Delta R$ (%) vs. Ratio play-out delay / mean RTT for streaming (top left) Foreman, (top middle) Mother & Daughter, (top right) Carphone, (bottom left) Container, (bottom middle) News, and (bottom right) Salesman.

The excess rate $\Delta R$ is around 9.5% and 7%, in the case of Foreman (top left) and Container (bottom left), respectively. Then, when this ratio is increased to eight, $\Delta R$ reduces respectively to 4% and 3%, as shown in Figure 3.2. Note that the performances of $\Delta R$ for Foreman (top left) and Carphone (top right) exhibit comparatively larger values relative to those for the other four sequences. This is analogous to what was observed in the context of the results shown in Figure 3.1.

Finally, it should be mentioned that evaluating the performance of RaDiO relative to Ideal in this scenario may seem somewhat unfair as the performance of the second system is obtained assuming infinite play-out delay. Nonetheless, a general conclusion that follows from this second set of experiments is that there will be an additional performance loss of RaDiO relative to the previous set of results shown.
in Figure 3.1, when the play-out delay and the mean round-trip delay are not significantly different in size.

3.4 Window Control

In the previous section, we saw that distortion-rate performance is measured in an average sense. Therefore, it is possible for the optimization algorithm to decide to schedule the transmissions of many of the $L$ data units in a single instance, thereby causing a large spike in the instantaneous transmission rate, which is certainly unacceptable and should be avoided. Note that despite this undesirable effect, one could still have a low average transmission rate, if for the rest of the streaming session the number of scheduled transmissions is comparatively smaller.

We employ window control to deal with this issue. This is a standard technique that has been used in the past in both classical networking and media streaming for limiting the number of data units that can be simultaneously considered for transmission\footnote{The particular algorithm that is described in this section has been adopted from [41]. We employ this algorithm in Chapter 5 to perform simulation experiments. Its description is included here for completeness.}. In particular, instead of computing the schedules for all $L$ data units at the beginning of the session and then executing those schedules throughout the session, we compute packet schedules for data units at every transmission opportunity during the session as follows. Let $s$ be a given transmission instance (or time). Then, we compute packet schedules at $s$ for only a subset $\mathcal{W}(s)$ of data units, where the window $\mathcal{W}(s) \in \{1, 2, \ldots, L\}$ slides over the whole set of data units as $s$ progresses. Sometimes, this approach is also denoted as "sliding window" technique as the edges of the window $\mathcal{W}(s) = \{w_{\text{lag}}(s), w_{\text{lag}}(s) + 1, \ldots, w_{\text{lead}}(s)\}$ advance monotonically.
with $s$, i.e., they slide over $\{1, 2, \ldots, L\}$.

We chose $w_{\text{lag}}(s)$ and $w_{\text{lead}}(s)$ in the following manner. Let $\Delta$ denote the play-out delay, which is the time period between the instance when a client requests a media presentation (the client clicks "play" on its application), and the moment when the first data unit in the session is played at the client. Sometimes this quantity is also denoted as the pre-roll delay, as it refers to the period of time during which received data units are stored in the client’s buffer for forthcoming play. This delay helps the client to deal with jitter in arrival times of packets and to recover from packet losses by retransmissions. Next, let $\Delta_0$ denote the initial size of the sliding window in units of time (the meaning of which will become clear in what follows), and let $\nu$ denote the playback speed of the media presentation at the client. $\nu$ is a unit-less quantity and determines how fast media content is consumed relative to the original speed at which the content was captured and time-stamped. For example, $\nu = 1/3$ means that each second of media content will be consumed in 3 seconds of client time*. Typically, $\nu$ is kept constant and equal to one throughout the duration of the session†. Then, the lagging edge $w_{\text{lag}}(s)$ and the leading edge $w_{\text{lead}}(s)$ of the sliding window are computed as

\begin{align*}
  w_{\text{lag}}(s) &= \begin{cases}
    1 & : s < \Delta \\
    \arg \min_i t_{DTS,i} \geq (s - \Delta)\nu & : s \geq \Delta,
  \end{cases} \quad (3.8) \\
  w_{\text{lead}}(s) &= \arg \max_i t_{DTS,i} \leq s\nu + \Delta_0. \quad (3.9)
\end{align*}

Note that the window edges are determined based on the delivery deadlines of the data units. Computing $w_{\text{lag}}(s)$ as in (3.8) is intuitive as it does not make sense

---

*Referring to the clock running at the client and according to which content is played back.

†Some works have considered adapting $\nu$ during the presentation as a means of enhancing the client's buffer performance [175, 176, 238–242].
to compute any longer transmission schedules of data units that have expired. In particular, these are the data units $l \in \{1, 2, \ldots, L\}$ whose delivery deadlines at the client have been already passed, i.e., $\frac{t_{\text{DSI}}}{\nu} + \Delta < s$. On the other hand, the choice of $w_{\text{lead}}(s)$ is not intuitive and therefore is not unique across implementations. In (3.9), we have selected the leading edge to be a linear function of $s$ such that the initial size of the transmission window is smaller, which prevents the occurrence of a sudden burst in the instantaneous transmission rate at the onset of the streaming session. Then, over the period of time given by the pre-roll delay the window is gradually increased to a maximum size determined by the parameters $\Delta$ and $\Delta_0$, which allows for the transmission rate to be incrementally increased to a certain maximum value. In Figure 3.3, we show the progress over time ($s$) of the lagging and leading edges of the sliding transmission window.

![Diagram](image)

**Figure 3.3 :** Implementation of window control for media streaming.

Finally, it deserves to be mentioned that when deriving the first draft of the present section we have borrowed parts of the discussion on window control from the
corresponding section in [41], courtesy of the first author. The section, however, has undergone a complete revision since then.

3.5 Computational Complexity

In this section, we briefly describe the computational requirements of the optimization framework for steady-state operation (ignoring the transients at the beginning and at the end of a media session), and provide an upper bound on the number of operations per data unit. The discussion that follows has been presented in part in [243].

The complexity of RaDiO streaming is on the order of $N_i|\mathcal{W}|(C|\mathcal{N}_c| + 2^{NM})$, where $N_i$ is the number of iterations that the ISA algorithm from Section 3.3 performs until convergence (typically on the order of 2-3) and $|\mathcal{W}|$ is the size of the transmission window $\mathcal{W}$ during which a data unit is considered for transmission. The multiplicative factor $N_i|\mathcal{W}|$ is needed because the optimal policy $\pi^*$ for a data unit is computed at every iteration of the ISA algorithm and as long as the data unit is in the transmission window $\mathcal{W}$. Note, however, that once the reception of the data unit is acknowledged, there is no further computing cost associated with it even though the data unit may still be present in the transmission window $\mathcal{W}$. Therefore, the quantity provided above accounts for the computational complexity in the worst possible case, i.e., it represents an upper bound, as mentioned on the beginning. Furthermore, in the expression, $|\mathcal{N}_c|$ is the size of the concealment set for a data unit and it signifies the number of concealment events that are considered when the optimal policy $\pi^*$ for a data unit is computed. $C$ is a constant that can be very large and that depends on the sizes of the ancestor and descendant sets for the data units that are involved in the concealment events in $\mathcal{N}_c$. Finally, $M$ is the
number of network paths over which data units can be sent and $N$ is the number of transmission opportunities over which the optimal policy $\pi^*$ is computed for a data unit.
Chapter 4

Computing the Optimal Policy $\pi^*$

In this chapter, we address three advanced streaming scenarios using our rate-distortion optimization framework for packet scheduling from Chapter 3. In particular, here we formulate instances of our advanced framework for the scenarios of streaming with diversity and streaming from an intermediate proxy server located at the network edge. In addition, we also present a formulation of our framework for an unconventional streaming scenarios denoted streaming with rich acknowledgements. In this last scenario, we employ an unconventional method for acknowledging the reception of media packets in sender-driven streaming.

As shown at the end of Section 3.3, the core step of our optimization procedure is computing the optimal transmission policy $\pi^*$ for streaming a single data unit from the media presentation. The formulation of the algorithm for computing the optimal policy $\pi^*$ is dependent on the transmission scenario under consideration. In the following, we formulate this algorithm for each of the three advanced streaming scenarios considered in this chapter, starting with the scenario of streaming with diversity. But first, we introduce some necessary preliminaries that appear throughout the rest of the chapter.

We assume that there are $N$ discrete transmission opportunities $t_0, t_1, \ldots, t_{N-1}$ prior to a data unit’s delivery deadline $t_{DTS}$. At every $t_i$, for $i = 0, \ldots, N - 1$, an

---

Publications related to this chapter are [138, 231, 232, 244-249].
agent* in the streaming process, performs a transmission action $a_i$. In the case of $a_i = 1$, the agent sends a packet, either a data packet or a request packet, while in the case of $a_i = 0$ the agent does not send anything. The nature of data packets and request packets will become apparent as we examine the specific streaming scenarios considered in this chapter. The transmission actions $a_i, a_{i+1}, \ldots, a_{N-1}$ represent a transmission schedule or policy $\pi$ according to which the agent sends packets related to a single data unit at $t_i, t_{i+1}, \ldots, t_{N-1}$. It is the duty of the agent to recompute the optimal policy $\pi^*$ at every $t_i$ and then set $a_i$ to the first transmission action in $\pi^*$. Note that it would be sufficient to determine $\pi^*$ only once at time $t_0$, except for the fact that $\lambda'$ in (3.4) may be adjusted by the optimization procedure from Section 3.3 at each transmission opportunity $t_i$ in order to take into account feedback observed by the agent due to transmissions of packets related to other data units prior to $t_i$. This feedback can be in the form of acknowledgement packets or actual data packets or both, depending on the transmission scenario under consideration.

4.1 Streaming with Diversity

This section addresses the application of diversity to streaming of packetized media in a rate-distortion optimized way. In a packet switched network, diversity is achieved by using multiple transmission paths over the network, in a sender-driven transmission. Similarly, in a receiver-driven transmission diversity is achieved by simultaneously requesting packets from multiple servers located at different locations in the network. The two prospective diversity scenarios are shown in Figure 4.1.

*This can be a media server, or an intermediate proxy server, or a receiving client.
Figure 4.1: Diversity in packet networks. (a) Path diversity. (b) Server diversity.

4.1.1 Sender-driven transmission

Recall that there are $M$ different network paths over which the server can simultaneously send a data unit to the client. At each transmission opportunity $t_i$, for $i = 0, 1, \ldots, N - 1$, the server is allowed to transmit a packet for the data unit on the forward channels of any $m \leq M$ paths. The server need not transmit a packet at every transmission opportunity. The server does not transmit any further packets after an acknowledgement packet (ACK) is received on the backward channel of any of the paths.

At each transmission opportunity $t_i$, $i = 0, 1, \ldots, N - 1$, the server takes an action $a_i = [a_{i1}, \ldots, a_{iM}]$, where $a_{im} = 1$ means that a packet is sent on the forward channel of path $m$ and $a_{im} = 0$ means that no packet is sent on the forward channel of path $m$. Then, at the next transmission opportunity $t_{i+1}$, the server makes an observation $o_i$, where $o_i$ is the set of acknowledgements received by the server in the interval $(t_i, t_{i+1}]$. For example, $o_i = \{ACK_{j_1}^{m_1}, ACK_{j_2}^{m_2}\}$ means that during the interval $(t_i, t_{i+1}]$, ACKs arrived on the backward channels for the packets sent at time $t_{j_1}$ and $t_{j_2}$ on the forward channels of paths $m_1$ and $m_2$, respectively. The history,
or the sequence of action-observation pairs \((a_0, o_0) \circ (a_1, o_1) \circ \cdots \circ (a_i, o_i)\) leading up to time \(t_{i+1}\), determines the state \(q_{i+1}\) at time \(t_{i+1}\), as illustrated in Figure 4.2. If the final observation \(o_i\) includes an ACK, then \(q_{i+1}\) is a final state. In addition, any state at time \(t_N = t_{DTS}\) is a final state. Final states in Figure 4.2 are indicated by double circles.

![Markov decision tree for a data unit with packet path diversity.](image)

The action \(a_i\) taken at a non-final state \(q_i\) determines the transition probabilities \(P(q_{i+1}|q_i, a_i)\) to the next state \(q_{i+1}\). Formally, a policy \(\pi\) is a mapping \(q \mapsto a\) from non-final states to actions. Thus any policy \(\pi\) induces a Markov chain with transition probabilities \(P_\pi(q_{i+1}|q_i) \equiv P(q_{i+1}|q_i, \pi(q_i))\), and consequently also induces a probability distribution on final states. Let \(q_F\) be a final state with history \((a_0, o_0) \circ (a_1, o_1) \circ \cdots \circ (a_{F-1}, o_{F-1})\), and let \(q_{i+1} = q_i \circ (a_i, o_i)\), \(i = 1, \ldots, F-1\), be the sequence of states leading up to \(q_F\). Then \(q_F\) has probability \(P_\pi(q_F) = \prod_{i=0}^{F-1} P_\pi(q_{i+1}|q_i)\), transmission cost \(\rho_\pi(q_F) = \sum_{i=0}^{F-1} \sum_{m=1}^{M} a_{im}\), and error \(\epsilon_\pi(q_F) = 0\) if \(o_{F-1}\) contains an ACK and otherwise \(\epsilon_\pi(q_F)\) is equal to the probability that none of the transmitted
packets arrives at the client on time, given \( q_F \). Hence, we can express the expected cost and error for the Markov chain induced by policy \( \pi \):

\[
\rho(\pi) = E_\pi \rho_\pi(q_F) = \sum_{q_F} P_\pi(q_F) \rho_\pi(q_F),
\]

\[
\epsilon(\pi) = E_\pi \epsilon_\pi(q_F) = \sum_{q_F} P_\pi(q_F) \epsilon_\pi(q_F).
\]

We find the optimal policy \( \pi^* \) at every \( t_i \) by examining the expected error-cost performances \( \epsilon(\pi) + \lambda' \rho(\pi) \) over all possible policies \( \pi \), where the expected error-cost pairs \( \{(\rho(\pi), \epsilon(\pi)) : \pi \in \Pi\} \) are calculated conditioned on \( q_i \) and all the policies under consideration are consistent with the history \( (a_0, o_0) \circ (a_1, o_1) \circ \cdots \circ (a_{i-1}, o_{i-1}) \) leading up to state \( q_i \) at time \( t_i \).

In the following we explain how \( \epsilon(\pi) \) and \( \rho(\pi) \) are computed for a given policy \( \pi \). Let \( t_i \) be the current transmission opportunity and let \( C^F_{jm}, C^B_{jm} \in \{1, \ldots, K\} \) be respectively the states on the forward and the backward channel of path \( m = 1, \ldots, M \) at transmission opportunity \( t_j : j \leq i \). We assume that the sender has this information available. It is a reasonable assumption, as any congestion control mechanism employed by a streaming media system, such as those in [72–74, 250–254], will include some kind of channel estimation. This is because in the absence of explicit feedback from the network, the sender and/or the receiver must infer the state of the network by observing data units as they enter and leave the network. Furthermore, to examine how the performance of the optimization framework is sensitive to not having this information available, in Section 5.2 we perform streaming experiments based on network traces of packet delays and packet losses collected in the Internet. For this purpose, we propose in Section 5.2 an online channel estimation procedure that is used in conjunction with the scheduling framework.

The expected error for a policy \( \pi \) is simply the probability that all the transmitted packets from \( \pi \) as well as those from the transmission history do not arrive at the
client on time. Hence we write

$$\epsilon(\pi) = \prod_{j<i, \, m : \, a_{jm}=1} P\{\text{FTT}^C_{im} > t_{DTS} - t_j | \text{RTT}^C_{im} > t_i - t_j\} \prod_{j \geq i, \, m : \, a_{jm}=1} \sum_{k=1}^{K} \mathcal{P}_{C_{im}^{k(F)}}^{(j-i)} P\{\text{FTT}^k > t_{DTS} - t_j\} \quad (4.1)$$

Furthermore, upon receipt of an acknowledgement packet, the server truncates its transmission pattern and does not consider sending any packets afterwards. Therefore, the cost for each transmission $a_{jp} = 1 : j \in \{i, \ldots, N-1\}, p = 1, \ldots, M$ is equal to the probability that none of the previous transmissions results in an acknowledgement packet received by the server by $t_j$. Hence, the expected cost is simply the sum of the individual costs over all transmission opportunities and paths, i.e.,

$$\rho(\pi) = \sum_{j \geq i, \, p : \, a_{jp}=1} \prod_{l<i, \, m : \, a_{lm}=1} P\{\text{RTT}^C_{im} > t_j - t_l | \text{RTT}^C_{im} > t_i - t_l\} \prod_{i \leq j, \, m : \, a_{im}=1} \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \mathcal{P}_{C_{im}^{k_1(F)}}^{(l-i)} \mathcal{P}_{C_{im}^{k_2(B)}}^{(l-i)} P\{\text{RTT}^{k_1k_2} > t_j - t_l\} \quad (4.2)$$

where the first product term in both $\epsilon(\pi)$ and $\rho(\pi)$ accounts for previous transmissions (if any). Finally, note that the contribution to the error-cost of prospective transmissions in $\pi$ at opportunities $t_j : j > i$ can be accounted for only as an expected value over all possible channel states at $t_j$.

### 4.1.2 Receiver-driven transmission

In this scenario there are $M$ media servers at different locations in the network. The client communicates with the servers over $M$ individual network paths, respectively. At every $t_i$ the client is allowed to transmit a request packet for a data unit on the backward channels of any $m \leq M$ paths requesting transmission of that data unit.
from the corresponding media servers. The client need not transmit a request at every transmission opportunity. The client does not transmit any further requests after a packet with the data unit arrives on the forward channel of any of the paths.

At each transmission opportunity $t_i$, $i = 0, 1, \ldots, N-1$, the client takes an action $a_i = [a_{i1}, \ldots, a_{iM}]$, where $a_{im} = 1$ means that a request is sent on the backward channel of path $m$ and $a_{im} = 0$ means that no request is sent on the backward channel of path $m$. Then, at the next transmission opportunity $t_{i+1}$, the client makes an observation $o_i$, where $o_i$ is the set of packets received by the client in the interval $(t_i, t_{i+1}]$. For example, $o_i = \{DAT_{j_1}^{m_1}, DAT_{j_2}^{m_2}\}$ means that during the interval $(t_i, t_{i+1}]$, packets with the data unit arrived on the forward channels as a response to the requests sent at time $t_{j_1}$ and $t_{j_2}$ on the backward channels of paths $m_1$ and $m_2$, respectively.

As noted the operation of the client in this scenario is analogous to the one of the server from Section 4.1.1, with ACKs being replaced with DATs. Therefore, there is no need for describing it again here and we end this section with the corresponding expected error-cost expressions. We use the same notation and assumption as in Section 4.1.1. $\epsilon(\pi)$ is easily obtained from (4.1) by replacing the forward trip time ($FTT$) with the round trip time ($RTT$). Similarly, $\rho(\pi)$ is obtained from (4.2) by noting the following. A request packet incurs a cost of one, which is the cost of transmission of a single data unit packet, only if the request packet arrives eventually at the server. Therefore, the cost of sending a request packet is less than one and equals the probability that the request is not lost on the backward channel of a
network path. Hence, we write

\[ \epsilon(\pi) = \prod_{j<i, \ m : a_{jm}=1} P\{RTT^{C^m_j C^o_m} > t_{DTS} - t_j | RTT^{C^m_j C^o_m} > t_i - t_j \} \prod_{j \geq i, \ m : a_{jm}=1} \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} P_{C^m_{im} k_1(B)} P_{C^o_{im} k_2(F)} P\{RTT^{k_1 k_2} > t_i - t_j \} \]

\[ \rho(\pi) = \sum_{j \geq i, \ p : a_{jp}=1} \prod_{l<i, \ m : a_{lm}=1} P\{RTT^{C^m_l C^o_m} > t_j - t_l | RTT^{C^m_l C^o_m} > t_i - t_l \} \prod_{i \leq l<j, \ m : a_{lm}=1} \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} P_{C^m_{im} k_1(B)} P_{C^o_{im} k_2(F)} P\{RTT^{k_1 k_2} > t_l - t_i \} \times \sum_{k=1}^{K} P^{(j-i)}_{C^o_{ip} k(B)} P\{BTT^k < \infty \} \]

where the last sum in \( \rho(\pi) \) equals the probability that the request packet \( a_{jp} = 1 \) is not lost on the backward channel of path \( p \).

### 4.2 Proxy-Driven Streaming from the Network Edge

This section addresses the problem of streaming packetized media over a lossy packet network through an intermediate proxy server to a client, in a rate-distortion optimized way. The proxy is located at the junction of the backbone network and the last hop to the client. The proxy employs a hybrid receiver/sender-driven transmission scheme to communicate with both the media server and the client. Packets may be lost in the backbone network due to congestion, or on the last hop due to erasures. The scenario under consideration is illustrated in Figure 4.3.

#### 4.2.1 Proxy-Driven Transmission

We consider a system in which the media server communicates to the client indirectly, through a proxy. We refer to the server-proxy path as Path 1 and to the
proxy–client path as Path 2. Thus, following the notation from Section 3.3, there are $M = 2$ network paths in total. For the rest of the chapter, we model the forward and backward directions of a network path using the model from Section 3.2 with $K=1$ state. In other words, forward and backward channels are modelled as independent packet erasure channels with random delays.\textsuperscript{1}

A multimedia session starts when a client requests a presentation from the media server. The request packet is received by the proxy server and is not forwarded further. The proxy then sends a request to the media server for the rate-distortion preamble\textsuperscript{2} for the desired presentation.

After it has the preamble, the proxy starts communicating simultaneously with the media server and the client using a hybrid receiver/sender-driven transmission scheme. The communication with the media server is receiver-driven, where the proxy sends request packets to the media server, requesting a particular data unit

\textsuperscript{1}Our assumption for independence is for the purposes of tractability of our analysis. This is justified provided that the retransmissions are infrequent (e.g., the retransmission intervals are larger than the one way delay), as noted for example by Bolot [255].

\textsuperscript{2}The concept of rate-distortion preamble was described in Section 3.1.
to be transmitted. The media server responds to a request by sending the requested data unit in a data packet to the proxy. The proxy may send requests for a particular data unit periodically if necessary, until the data unit finally arrives at the proxy, or until the proxy gives up. Upon seeing an arriving data packet, the proxy stores the corresponding data unit in its buffer for later (re)transmissions to the client and stops requesting that data unit from the media server. The communication with the client is sender-driven, where the proxy now sends to the client the buffered data unit periodically if necessary, until it receives an acknowledgement packet from the client for that data unit, or until the proxy gives up transmitting the data unit. We call this hybrid receiver/sender-driven transmission scheme proxy-driven.

Specifically, at each transmission opportunity $t_i$, $i = 0, 1, \ldots, N - 1$, the proxy takes an action $a_i$, where $a_i = 1$ if the proxy sends a packet and $a_i = 0$ otherwise. Then, at the next transmission opportunity $t_{i+1}$, the proxy makes an observation $o_i$, where $o_i$ summarizes the packets that arrived at the proxy during the interval $(t_i, t_{i+1}]$. In particular, $o_i = \emptyset$ means that no packets arrived, $o_i = DAT$ means that at least one data packet arrived from the sender (but no acknowledgement packets arrived from the client), and $o_i = ACK$ means that at least one acknowledgement packet arrived from the client, during the interval $(t_i, t_{i+1}]$. As in Section 4.1, the history, or the sequence of action-observation pairs $(a_0, o_0) \circ (a_1, o_1) \circ \cdots \circ (a_{i-1}, o_{i-1})$ leading up to time $t_i$, determines the state $q_i$ at time $t_i$.

Two cases of transmission history can be distinguished: (1) no previous $DAT$s, i.e., $o_0 = o_1 = \cdots = o_{i-1} = \emptyset$, and (2) with previous $DAT$s. In the following we show how $\pi^*$ can be determined for each of them.
Figure 4.4: The decision tree of a transmission policy for a data unit when no previous DATs have arrived at the proxy.

4.2.2 No previous DATs

Figure 4.4 shows the tree of action-observation pairs associated with a policy \( \pi \) for this case of transmission history. Final states in Figure 4.4 are indicated by double circles. Starting from the initial state at time \( t_0 \), the proxy follows the upper path (the backbone of the tree), taking actions \( a^0_0, a^0_1, a^0_2, \ldots \), i.e., sending request packets (or not) to the server, until it receives a data packet. Once the first data packet is received, say during the interval \( (t_i, t_{i+1}] \), i.e., \( \alpha_i = DAT \), the proxy begins to follow the corresponding horizontal branch from time \( t_{i+1} \), taking actions \( a^{i+1}_{i+1}, a^{i+1}_{i+2}, a^{i+1}_{i+3}, \ldots \), i.e., sending data packets (or not) to the client, until it receives an acknowledgement packet. The decision tree of any particular policy \( \pi \) can also be represented as an
upper triangular matrix of actions:

$$
\pi = \begin{bmatrix}
    a^0_0 & a^0_1 & a^0_2 & \cdots & a^0_{N-1} \\
    a^1_0 & a^1_1 & a^1_2 & \cdots & a^1_{N-1} \\
    a^2_0 & a^2_1 & a^2_2 & \cdots & a^2_{N-1} \\
    \ddots & & & \ddots & \\
    a^N_{N-1} & & & & a^N_{N-1}
\end{bmatrix}
$$

Here, the first row \(a^0 = [a^0_0, a^0_1, a^0_2, \ldots, a^0_{N-1}]\) is the sequence of actions to take until the first DAT is received and each subsequent row \(a^{i+1} = [a^{i+1}_0, a^{i+1}_{i+2}, \ldots, a^{i+1}_{N-1}]\) is the sequence of actions to take until the first ACK is received.

Each such policy \(\pi\) induces a stochastic finite state automaton. Label the state preceding action \(a^k_i\) by \(q^k_i\). Then the transition probabilities along the backbone are given by \(P_\pi(q^0_{i+1}|q^0_i) = A(i)\) and \(P_\pi(q^{i+1}_0|q^0_i) = 1 - A(i)\), where

$$
A(i) = \prod_{j \leq i : a^G_j = 1} P\{RTT_1 > t_{i+1} - t_j | RTT_1 > t_i - t_j\}.
$$

Note that these transition probabilities depend on \(a^0\) only.

Using the transition probabilities, the expected error \(\epsilon(\pi|q^0_i)\) of policy \(\pi\) starting from the backbone state \(q^0_i\) can be computed recursively for \(i = N-1, N-2, \ldots, 0\) as

$$
\begin{align*}
\epsilon(\pi|q^0_N) &= 1 \\
\epsilon(\pi|q^0_i) &= A(i)\epsilon(\pi|q^0_{i+1}) + (1 - A(i))\epsilon(\pi|q^{i+1}_{i+1}),
\end{align*}
$$

where

$$
\epsilon(\pi|q^{i+1}_{i+1}) = \prod_{j > i : a^{i+1}_j = 1} P\{FTT_2 > t_{DTS} - t_j\}
$$

is the expected error of policy \(\pi\) starting from branch state \(q^{i+1}_{i+1}\). Note that (4.4) depends only on \(a^{i+1}\).
Similarly, the expected cost \( \rho(\pi|q^0_i) \) of policy \( \pi \) starting from the backbone state \( q^0_i \) can be computed recursively. As introduced in Section 3.3, let \( \gamma_2 \) be the cost per source byte of transmitting a data packet from the proxy to the client downstream on Path 2, and let \( \gamma_1 \) be the cost per source byte of transmitting a data packet from the server to the proxy downstream on Path 1, so that \( \gamma_1(1 - \epsilon_{B_1}) \) is the expected cost per source byte of transmitting a request packet from the proxy to the server upstream on Path 1. Then the expected cost of policy \( \pi \) starting from backbone state \( q^0_i \) can be computed recursively for \( i = N - 1, N - 2, \ldots, 0 \) as

\[
\rho(\pi|q^0_N) = 0
\]

\[
\rho(\pi|q^0_i) = \gamma_1(1 - \epsilon_{B_1})a^0_i + A(i)\rho(\pi|q^0_{i+1}) + (1 - A(i))\rho(\pi|q^{i+1}_{i+1}),
\]

where

\[
\rho(\pi|q^{i+1}_{i+1}) = \gamma_2 \sum_{j > i: a^i_{j+1} = 1} \prod_{k < j: a^i_{k+1} = 1} P\{RTT_2 > t_j - t_k\} \tag{4.5}
\]

is the expected cost of policy \( \pi \) starting from branch state \( q^{i+1}_{i+1} \). Note that (4.5) depends only on \( a^{i+1} \).

Finally, the expected Lagrangian of policy \( \pi \) starting from state \( q \) can be expressed

\[
J(\pi|q) = \epsilon(\pi|q) + \lambda\rho(\pi|q).
\]

By expanding the earlier recursive expressions, the expected error, cost, and
Lagrangian of policy $\pi$ can also be expressed non-recursively as

$$
\epsilon(\pi | q_0^0) = \sum_{i=0}^{N-1} \left( \prod_{j=0}^{i-1} A(j) \right) \left( 1 - A(i) \right) \epsilon(\pi | q_{i+1}^{i+1}) + \prod_{j=0}^{N-1} A(j)
$$

$$
\rho(\pi | q_0^0) = \sum_{i=0}^{N-1} \left( \prod_{j=0}^{i-1} A(j) \right) \left( 1 - A(i) \right) \rho(\pi | q_{i+1}^{i+1}) + \sum_{i=0}^{N-1} \left( \prod_{j=0}^{i-1} A(j) \right) \gamma_1 (1 - \epsilon_B) a_i^0
$$

$$
J(\pi | q_0^0) = \sum_{i=0}^{N-1} \left( \prod_{j=0}^{i-1} A(j) \right) \left( 1 - A(i) \right) J(\pi | q_{i+1}^{i+1}) + \prod_{j=0}^{N-1} A(j)
$$

$$
+ \lambda' \sum_{i=0}^{N-1} \left( \prod_{j=0}^{i-1} A(j) \right) \gamma_1 (1 - \epsilon_B) a_i^0
$$

Note that the expression above for the expected cost $\rho(\pi | q_0^0)$ can easily be factored into the form $\gamma_1 \rho_1(\pi) + \gamma_2 \rho_2(\pi)$ that was introduced in Section 3.3.

Thus, minimizing the Lagrangian over all policies $\pi$ can be expressed

$$
\min_{\pi} J(\pi | q_0^0) = \min_{a^0} \left\{ \sum_{i=0}^{N-1} \left( \prod_{j=0}^{i-1} A(j) \right) \left( 1 - A(i) \right) J^*(q_{i+1}^{i+1}) \right. \\
\left. + \prod_{j=0}^{N-1} A(j) + \lambda' \sum_{i=0}^{N-1} \left( \prod_{j=0}^{i-1} A(j) \right) \gamma_1 (1 - \epsilon_B) a_i^0 \right\},
$$

where

$$
J^*(q_{i+1}^{i+1}) = \min_{a^{i+1}} J(\pi | q_{i+1}^{i+1}).
$$

That is, minimizing the Lagrangian over $\pi$ can be accomplished by first independently minimizing the Lagrangian $J(\pi | q_{i+1}^{i+1})$ over the sequence of actions $a^{i+1}$, for $i = 0, 1, \ldots, N - 1$, and then minimizing $J(\pi | q_0^0)$ over $a^0$. In this way the optimal policy $\pi^*$ can be computed for each $\lambda'$. Note that at times $t_i$, for $i > 0$, the proxy needs to recompute only the rows $a^{i+1}$ of the matrix representation of $\pi^*$ and the entries $j \geq i$ of the first row $a^0$. 
4.2.3 With previous DATs

Here, the data unit has already arrived at the proxy. Assume that the first data packet is received at the proxy by time $t_i$, for $i = 1, 2, \ldots, N - 1$. Then, at every $t_j$, for $j \geq i$, the proxy needs to determine only the elements $a_j^i, a_{j+1}^i, \ldots, a_{N-1}^i$ of the row $\mathbf{a}^i$ that minimize the Lagrangian $J(\pi|q_j^i)$.

To account for previous transmissions, if any, of the data unit on the last hop due to actions $a_1^i, a_{i+1}^i, \ldots, a_{j-1}^i$, we adjust the expressions for the expected error and cost, given in (4.4) and (4.5), respectively, as follows

$$
\epsilon(\pi|q_j^i) = \prod_{i \geq 1} P\{FTT_2 > t_{DTS} - t_i|RTT_2 > t_j - t_i\}, \quad (4.6)
$$

$$
\rho(\pi|q_j^i) = \gamma_2 \sum_{i > j: a_i^j = 1} \prod_{k \leq i: a_k^j = 1} P\{RTT_2 > t_i - t_k|RTT_2 > t_j - t_k\}. \quad (4.7)
$$

Note that the conditional probabilities in (4.6) and (4.7) become unconditional for transmission actions $a_j^i, a_{j+1}^i, \ldots, a_{N-1}^i$. Finally, as stated earlier, we are interested in the sequence of actions $[a_j^i, a_{j+1}^i, \ldots, a_{N-1}^i]^*$ that minimizes the Lagrangian $J(\pi|q_j^i) = \epsilon(\pi|q_j^i) + \lambda'\rho(\pi|q_j^i)$, i.e.,

$$
[a_j^i, a_{j+1}^i, \ldots, a_{N-1}^i]^* = \arg\min_{a_j^i, a_{j+1}^i, \ldots, a_{N-1}^i} J(\pi|q_j^i).
$$

4.3 Streaming with Rich Acknowledgements

Here, we employ an unconventional procedure for communicating to the server the receipt of media packets for sender-driven transmission. Instead of separately acknowledging each media packet as it arrives, the receiver, i.e., the client, periodically sends to the server a single acknowledgment packet, denoted henceforth rich acknowledgment, that contains information about all media packets that have arrived at the
client by the time the rich acknowledgment is sent. This information in essence reflects the current state of the client’s buffer, i.e., which packets have been received by the client thus far. The proposed scenario of streaming with rich acknowledgments is illustrated in Figure 4.5.

![Diagram of streaming on demand using rich acknowledgments.]

Figure 4.5: Streaming on demand using rich acknowledgments.

It should be noted that the concept of rich acknowledgments is not new and has been introduced in the networking community under the name of vector acknowledgments. However, to the best of our knowledge rich acknowledgments have not been explored yet in media streaming. The purpose of vector acknowledgments is to allow a TCP sender to perform selective retransmission of lost data packets, which is not provided by the acknowledgment scheme employed by TCP, called cumulative ack (CACK) [256]. In essence, vector acks are binary maps that describe the correctly received or missing data in the receiver’s buffer and have been adopted into several proposed feedback schemes [257–259]. Another alternative for providing selective retransmission in TCP is the Selective ACK option (SACK) [260], which allows a receiver to communicate simultaneously the identities of several contiguous blocks.
of successfully received data.

**Media Communication using Rich ACKs**

There are $M = 1$ network paths between the media server and the client in this scenario. Once a media session between the server and the client is established, the media server starts sending packets with data units from the presentation on the forward channel at discrete transmission opportunities $t_i, t_{i+1}, \ldots$ according to its transmission policy. The client in turn periodically monitors the status of its buffer at every $t_i$ and returns this information to the server on the backward channel via a rich acknowledgment packet. The buffer information basically informs the server what data units have arrived at the client by the time ($t_i$) the rich acknowledgment is transmitted. At the same time this also informs the server what data units have not arrived at the client by $t_i$. Therefore, the information provided by a received rich acknowledgment is richer than that provided by a received conventional acknowledgment.

Note that in practice the server computes a sliding window of data units $\mathcal{W}_i$ at every $t_i$, as explained in Section 3.4. Only the data units from $\mathcal{W}_i$ are considered for transmission at $t_i$. Therefore, at every $t_i$ the client needs to send to the server information on the arrival status only for the data units in $\mathcal{W}_i$. In essence, this information comprises a binary vector $r^i$ of length $w_i$, where $r^i_j = 1(0)$ means that the $j^{th}$ data unit from $\mathcal{W}_i$ has (not) arrived at the client by $t_i$ and $w_i$ is the length of the window $\mathcal{W}_i$ in data units. Then, from the time stamp $t_i$ of a received rich acknowledgment the server can recompute the transmission window at that point ($t_i$) and thus can easily determine to which data units the information in $r^i$ refers to.

$^5$Thus the motivation for the name.
Finally, note that in order to be able to return the appropriate rich acknowledgement at every $t_i$, the client needs to know how the server computes its transmission window at that $t_i$. This knowledge can be provided to the client by the server at the beginning of the streaming session or it is simply fixed and known ahead of time.

Computing the Optimal Policy $\pi^*$

For streaming a single data unit from the presentation, at each transmission opportunity $t_i$, $i = 0, 1, \ldots, N - 1$, the server takes an action $a_i$, where $a_i = 1$ if the server sends a packet with the data unit and $a_i = 0$ otherwise. The observation $o_i$ that the server makes by the next transmission opportunity $t_{i+1}$ is the set of rich acknowledgments received by the server in the interval $(t_i, t_{i+1}]$. For example, $o_i = \{NAK(t_1), ACK(t_3)\}$ means that during the interval $(t_i, t_{i+1}]$, the rich acknowledgment sent at $t_1$ arrived at the server informing that the data unit has not arrived at the client by $t_1$ ($NAK(t_1)$) and that the rich acknowledgment sent at $t_3$ arrived at the server informing that the data unit has arrived at the client by $t_3$ ($ACK(t_3)$).

Note that for the purposes of our algorithm it is irrelevant for the server to distinguish the transmission times of the rich acknowledgments received within $(t_i, t_{i+1}]$ confirming that the data unit has arrived at the client by their respective transmission times. Once the server receives a confirmation that the data unit has arrived at the client, it stops sending any packets afterwards, regardless of the transmission times of the rich acknowledgments that brought that confirmation and regardless of any number of NAKs that might also arrive during $(t_i, t_{i+1}]$. Therefore, we drop the timing notation on these rich acknowledgments and simply use $ACK$ to denote the event that at least one rich acknowledgment has been received during $(t_i, t_{i+1}]$, confirming the receipt of the data unit due to previous transmissions. In addition,
we also denote the event \( o_i = \{ \text{NAK}(t_1), \text{ACK} \} \) simply as \( o_i = \text{ACK} \) due to the same argument. Finally, we denote the event \( o_i = \{ \text{NAK}(t_1), \text{NAK}(t_3) \} \) simply as \( o_i = \text{NAK}(t_3) \) since not receiving the data unit by \( t_3 \) implies that the data unit was certainly not received by \( t_1 \). In other words, we denote the events \( o_i \), when multiple NAKs are received as an observation, using only the most recently sent NAK.

The state space for streaming a single data unit with rich acknowledgements is shown in Figure 4.6. As before, final states \( q_F \), indicated by double circles in Figure 4.6, are all states at time \( t_N = t_{DTS} \) and all states \( q_{i+1} \) at transmission opportunities \( t_{i+1} \), for \( i = 0, 1, \ldots, N - 2 \), whose final observations \( o_i \) include an ACK.

![State space diagram](image)

Figure 4.6: The state space of a Markov decision process for streaming a data unit with rich acknowledgements.

We are interested in the policy \( \pi^* \) that minimizes the expected Lagrangian cost

\[
J(\pi) \equiv \epsilon(\pi) + \lambda' \rho(\pi) = \sum_{q_F} P_\pi(q_F) J_\pi(q_F),
\]

(4.8)
where \( J_\pi(q_F) \equiv \epsilon_\pi(q_F) + \lambda' \rho_\pi(q_F) \). We compute \( \pi^* \) using a dynamic programming [261–264] algorithm as follows. Let

\[
J_\pi(q_i) = \begin{cases} 
\epsilon_\pi(q_F) + \lambda' \rho_\pi(q_F) \\
\sum_{q_{i+1}} P(q_{i+1}|q_i, \pi(q_i)) J_\pi(q_{i+1}) 
\end{cases}
\]

be the expected Lagrangian of all paths through \( q_i \) (under \( \pi \)). Then let

\[
J^*(q_i) = \begin{cases} 
\epsilon_\pi(q_F) + \lambda' \rho_\pi(q_F) \\
\min_{a_i} \sum_{q_{i+1}} P(q_{i+1}|q_i, a_i) J^*(q_{i+1}) 
\end{cases}
\]  

(4.9)

By induction, \( J^*(q_i) \leq J_\pi(q_i) \) for all \( q_i \) and all \( \pi \), with equality if \( \pi = \pi^* \), where

\[
\pi^*(q_i) = \arg\min_{a_i} \sum_{q_{i+1}} P(q_{i+1}|q_i, a_i) J^*(q_{i+1})
\]  

(4.10)

for all non-final states \( q_i \). Thus the optimal policy (minimizing (4.8)) can be computed efficiently using (4.9) and (4.10).

Next, we provide the actual expressions for \( \epsilon_\pi(q_F), \rho_\pi(q_F) \) and \( P(q_{i+1}|q_i, a_i) \) used in the equations above. As given before, the transmission cost \( \rho_\pi(q_F) = \sum_{i=0}^{F-1} a_i \).

Now, if no NAKs have been received along the path that leads to \( q_F \) in the state space from Figure 4.6, then the expected error is \( \epsilon_\pi(q_F) = \prod_{i=1} a_i \prod \{FTT > t_{\text{DTS}} - t_i \} \). On the other hand, if at least one NAK has been received, then the expected transmission error is

\[
\epsilon_\pi(q_F) = \prod_{i: i < j, a_i = 1} P\{FTT > t_{\text{DTS}} - t_i | FTT > t_j - t_i \} \prod_{i: j \leq i, a_i = 1} P\{FTT > t_{\text{DTS}} - t_i \},
\]
where \( j \in \{1, \ldots, N-1\} \) is the index of the most recently sent NAK that has arrived at the server, i.e., \( NAK(t_j) \). Finally, we need to differentiate 4 possible cases for the transition probability \( P(q_{i+1}|q_i, a_i) \). As mentioned earlier, \( (a_0, o_0) \circ (a_1, o_1) \circ \cdots \circ (a_t, o_t) \) is the history, or the sequence of action-observation pairs that leads to state \( q_{i+1} \) at time \( t_{i+1} \). Then, we can have

(a) no NAK received along the path that leads to \( q_i \), i.e., \( NAK \notin \bigcup_{j=0}^{j-1} o_j \) and no NAK received during \( (t_i, t_{i+1}] \), i.e., \( NAK \notin o_i \)

\[
P(q_{i+1}|q_i, a_i) = \prod_{l=1}^{i} P\{BTT > t_{i+1} - t_l | BTT > t_i - t_l\}, \tag{4.11}
\]

(b) no NAK received along the path that leads to \( q_i \), i.e., \( NAK \notin \bigcup_{j=0}^{j-1} o_j \) and at least one NAK received during \( (t_i, t_{i+1}] \), i.e., \( NAK \in o_i \)

\[
P(q_{i+1}|q_i, a_i) = P\{t_i - t_k < BTT \leq t_{i+1} - t_k | BTT > t_i - t_k\} \times \prod_{l>k}^{i} P\{BTT > t_{i+1} - t_l | BTT > t_i - t_l\} \prod_{l : l < k, a_l = 1} P\{FTT > t_k - t_l\}, \tag{4.12}
\]

(c) at least one NAK received along the path that leads to \( q_i \), i.e., \( NAK \in \bigcup_{j=0}^{j-1} o_j \) and no NAK received during \( (t_i, t_{i+1}] \), i.e., \( NAK \notin o_i \)

\[
P(q_{i+1}|q_i, a_i) = \prod_{l>j} P\{BTT > t_{i+1} - t_l | BTT > t_i - t_l\}, \tag{4.13}
\]

(d) at least one NAK received along the path that leads to \( q_i \), i.e., \( NAK \in \bigcup_{j=0}^{j-1} o_j \) and at least one NAK received during \( (t_i, t_{i+1}] \), i.e., \( NAK \in o_i \)

\[
P(q_{i+1}|q_i, a_i) = P\{t_i - t_k < BTT \leq t_{i+1} - t_k | BTT > t_i - t_k\} \times \prod_{l>k}^{i} P\{BTT > t_{i+1} - t_l | BTT > t_i - t_l\} \times \prod_{l : l < k, a_l = 1} P\{FTT > t_k - t_l | FTT > t_j - t_l\}, \tag{4.14}
\]
where $j$ in (4.13) and (4.14) is the index of the most recently sent NAK received up to $t_i$, i.e., $NAK(t_j) \in \bigcup_{i=0}^{i-1} o_i$. Similarly, $k$ in (4.12) and (4.14) is the index of the most recently sent NAK received during $(t_i, t_{i+1}]$, i.e., $NAK(t_k) \in o_i$. Note that in (4.14), it necessarily holds $j < k$ and $P\{FTT > t_k - t_i | FTT > t_j - t_i\} = P\{FTT > t_k - t_i\}$ for $j \leq l$. 
Chapter 5

Experimental Results

In this chapter, we carry out a series of computer simulations in which via streaming experiments we examine the performance of our optimization framework for packet scheduling in the three advanced streaming scenarios considered in Chapter 4. Performance will be principally measured in terms of the end-to-end distortion and the associated transmission rate for streaming a video sequence. To simulate Internet behavior in the experiments we employ statistical models of packet delay and loss characteristics in the Internet. In addition, in a set of experiments at the end of the chapter we reproduce Internet behaviour via traces of packet delays and packet losses collected in the Internet, similar to the one shown in Figure 2.5.

A set of standard CIF test video sequences is employed in the experiments. The sequences are compressed using the 8.0 version of the state-of-the-art video codec H.264 [154] as follows. Each sequence is encoded at a frame rate of 30 fps and at three different bit-rates ranging around 320, 160 and 80 Kbps, respectively. For one of the sequences, Mother & Daughter, lower encoding rates than those declared above were sufficient to create the three encodings at a very good quality. For compression efficiency, in an encoding only the first frame of the sequence is intra-encoded (I-frame), while the rest of the frames are encoded predictively as P- and B-frames. There are three B-frames between consecutive P-frames in an encoding. The coding dependencies between the different frames in an encoding are illustrated in Figure 5.1.
Figure 5.1: Coding dependencies between frames in an encoding.

In the streaming experiments, all three encodings of a video sequence are employed. Switching between different encodings during streaming depending on the available network bandwidth is performed using SP/SI frames. In particular, in an encoding every 16-th frame is encoded as a primary SP frame, which enables us to switch between different encodings every 1/2 second. Then, when switching at a particular location of a primary SP frame needs to be made, a corresponding secondary SP frame or a corresponding SI frame is sent to the receiver. (See Section 2.1 for an illustration.) For more details on stream switching via SP/SI frames, the reader is referred to [90].

The encoding rate for an encoding is controlled via the quantization parameter \( QP \) assigned to each frame type. In an encoding, we typically selected a single value for the quantization parameter assigned to the I-, P- and B-frames in the encoding. For the SP frames, we selected a value for the primary quantization parameter that is two less than the value used for \( QP \) for the other frame types. In addition, for the secondary quantization parameter denoted \( QPSP \), we selected a value that is one less than the value used for \( QP \) for the other frame types. Through empirical experiments, we established that such chosen values for the two quantization parameters in SP-frames provide a good balance between the quality of
the primary SP-frames that are inserted in an encoding, and the size in bits of the corresponding secondary SP-frames that are not a regular part of an encoding, but are sent only when switching between encodings needs to be made. The encoding parameters, such as rate and quality, across different sequences and encodings are shown in Table 5.1.

<table>
<thead>
<tr>
<th>Sequence</th>
<th># frames</th>
<th>Average rate (Kbps)</th>
<th>Average Y-PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman</td>
<td>297</td>
<td>328.16</td>
<td>35.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>162</td>
<td>32.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>88.59</td>
<td>29.62</td>
</tr>
<tr>
<td>Mother &amp; Daughter</td>
<td>297</td>
<td>203.01</td>
<td>40.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>105.44</td>
<td>38.29</td>
</tr>
<tr>
<td></td>
<td></td>
<td>56.41</td>
<td>35.78</td>
</tr>
<tr>
<td>Carphone</td>
<td>301</td>
<td>305.89</td>
<td>35.28</td>
</tr>
<tr>
<td></td>
<td></td>
<td>154.45</td>
<td>32.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>80.70</td>
<td>29.93</td>
</tr>
<tr>
<td>Container</td>
<td>297</td>
<td>326.32</td>
<td>38.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>161.39</td>
<td>36.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td>77.19</td>
<td>33.52</td>
</tr>
<tr>
<td>News</td>
<td>297</td>
<td>332.05</td>
<td>40.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>168.58</td>
<td>36.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>86.21</td>
<td>33.37</td>
</tr>
<tr>
<td>Salesman</td>
<td>301</td>
<td>332.82</td>
<td>37.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>165.02</td>
<td>35.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>79.37</td>
<td>32.9</td>
</tr>
</tbody>
</table>

Table 5.1: Encoding rate and quality for the sequences used in the experiments.

5.1 Experiments with Statistical Channel Models

In this section, we examine the end-to-end rate-distortion performance for streaming packetized video content in the three advanced scenarios from Chapter 4. In
the experiments, Internet behavior of transmitted packets is simulated using statistical models of packet loss and delay. The time interval between transmission opportunities $T$ is set to 50 ms and the play-out delay $\Delta$ is 600 ms.

### 5.1.1 Streaming with Diversity

In this section, we examine rate-distortion optimized streaming over multiple network paths, i.e., streaming with diversity. The first scenario that we study is sender-driven streaming over multiple paths. Afterwards, we explore the second diversity scenario under consideration: receiver-driven streaming from multiple servers. Both of these scenarios were described in detail in Section 4.1. In each of them, distortion is measured as the luminance peak signal-to-noise ratio (Y-PSNR) in dB of the end-to-end MSE distortion for a video frame, averaged over all frames in the video sequence. Furthermore, rate is measured as the average aggregate transmission rate in Kbps on the forward channels of all $M$ paths between the server(s) and the client.

The forward and the backward channel on the network paths are modelled with a $K = 2$ state Hidden Markov model that was described in Section 3.2. The model parameters are kept the same over all paths and are specified in Table 5.2. In

<table>
<thead>
<tr>
<th>State</th>
<th>$\epsilon$ (%)</th>
<th>$\kappa$ (ms)</th>
<th>$\mu$ (ms)</th>
<th>$\sigma$ (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>State 1</td>
<td>3</td>
<td>25</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>State 2</td>
<td>12</td>
<td>25</td>
<td>125</td>
<td>50</td>
</tr>
</tbody>
</table>

(a) Loss and delay parameters.

<table>
<thead>
<tr>
<th>Model</th>
<th>$\pi_2$</th>
<th>$\tau_2$ (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Model 1</td>
<td>0.2</td>
<td>100</td>
</tr>
<tr>
<td>Model 2</td>
<td>0.5</td>
<td>500</td>
</tr>
<tr>
<td>Model 3</td>
<td>0.8</td>
<td>1000</td>
</tr>
</tbody>
</table>

(b) State transitions.

Table 5.2: Network path characterization.
particular, in Table 5.2a we specify the delay and loss characteristics for a channel state. We keep the same characteristics for the forward and the backward channel. The delay density is modeled using a shifted Gamma distribution specified with three parameters: shift $\kappa$, mean $\mu$ and standard deviation $\sigma^*$. Finally, the state transitions are modeled using two parameters: the stationary probability of being in State 2, $\pi_2$, and the expected duration of stay in State 2, $\tau_2$, once a transition is made to this state. We employ four sets of values for these parameters denoted Model 0 - 3 in Table 5.2b. Due to the selected values Models 0 - 3 cover a range of possibilities in terms of the loss and delay characteristics exhibited on a network path. The Lagrange multiplier $\lambda$ is fixed for the entire presentation in each of the two diversity cases.

**Sender-driven streaming**

In this scenario, we first study the performance of the proposed framework as a function of the number of paths available. The state transitions are generated using Model 2 in these experiments. It can be seen from Figure 5.2 that streaming Foreman over 2 network paths can improve performance compared to the case of streaming over a single network path. An improvement is observed over the whole range of available rates. The gains in performance range between 0.8 - 1 dB for transmission rates greater than 200 Kbps and between 1 - 1.25 dB for transmission rates smaller than 80 Kbps. In the mid range of transmission rates under consideration, 80-200 Kbps, the diversity gains are somewhat smaller, but still maintain a level of 0.6 dB. The improved performance when streaming over two network paths is due

*To the best of my knowledge the first reference where this model appears and its validity is established based on network measurements is [187].
Figure 5.2: R-D performance for streaming Foreman over $M = 1, 2$ and 3 paths.

to the fact that having an alternative path for transmission reduces dramatically
the probability of having to transmit on a forward channel that features degraded
quality (State 2) at transmission. This ultimately contributes to a higher likelihood
of delivering the media packets on time. Furthermore, it can be seen from Figure 5.2
that using further paths for streaming does not provide additional significant gains
in performance, since the performances of $M = 3$ and of $M = 2$ are quite similar.
As explained above having 2 network paths reduces substantially the likelihood of
facing a degraded channel at transmission on every path. Therefore, adding yet one
more path as an alternative does not provide further substantial benefits, given the
selected path model.

We observe a similar situation for streaming Mother & Daughter, as shown in
Figure 5.3. In particular, a noticeable improvement in performance over the whole
range of transmission rates is obtained when streaming is performed over two network
paths relative to the case of streaming over a single network path. However, adding
Figure 5.3: R-D performance for streaming *Mother & Daughter* over $M = 1, 2$ and 3 paths.

yet one more path, i.e., streaming over three network paths, does not provide further substantial gains, as was also observed in the case of *Foreman*. Note however, that the performance gains due to diversity for *Mother & Daughter* are not as large as the corresponding ones for *Foreman*. That is because *Mother & Daughter* exhibits a lower level of content and motion complexity, which allows error concealment to be applied more successfully on missing packets. Therefore, even if a few extra packets arrive at the receiver on time due to the path diversity, that does not contribute to a very significant increase in reconstruction quality.

The performances of the other sequences in these experiments follow the same pattern as those of *Foreman* and *Mother & Daughter*. They are shown in Figure 5.4.

Next, we study the performance of the framework as a function of the quality of the network paths. As explained earlier depending on the state transition model, a path can exhibit different levels of quality in terms of the loss and delay characteris-
Figure 5.4: R-D performance for streaming (top left) Carphone, (top right) Container, (bottom left) News, and (bottom right) Salesman, over $M = 1, 2$ and $3$ paths.

...tics of packet transmissions. In Figure 5.5 we show the performances for streaming CIF Foreman over $M = 1$ and over $M = 2$ network paths in the case of Models 0, 1 and 3. It can be seen that streaming over two paths does not offer any advantages in the case of Model 0. That is expected, as a path here does not switch between states and hence there is no need for an alternative routing of packets over another path. However, as the frequency of state transitions and the duration of stay in State 2 for a path increase on the average, the need for an alternative routing in order to avoid a bad quality path steadily increases. Thus, the difference in performance between $M = 2$ and $M = 1$ is largest when the state transitions on a path are governed by Model 3.
Receiver-driven streaming

Similarly to the sender-driven case, here we study first the performance of the proposed framework as a function of the number of streaming servers that are available. The state transitions are generated using Model 2 in these experiments.

It can be seen from Figure 5.6 (left) that streaming Foreman from two servers can improve performance compared to the case of streaming from a single server.
An improvement is observed over the whole range of available rates. The gains in performance are consistently around 0.6 dB and increase to a level of 1.2-1.6 dB when the transmission rate is decreased below 130 Kbps. Furthermore, it can be seen from Figure 5.6 (left) that using further servers for streaming does not provide additional gains in performance, since the performances of $M = 3$ and of $M = 2$ are almost identical. Note that the relative performances for the three cases of $M = 1$, 2, and 3 in this scenario are analogous to those observed for the sender-driven case in Figure 5.2. Moreover, the reasons behind the relative performances from Figure 5.2 also apply here.

Similar outcome is observed for streaming *Mother & Daughter*, as shown in Figure 5.6 (right). In particular, it can be seen from the figure that for the selected simulation parameters streaming from $M = 2$ servers provides an improved performance over the whole range of transmission rates. However, adding further server diversity ($M = 3$) does not lead to additional improvements in performance. Note that the performance gains due to server diversity are somewhat smaller in this case, as observed from Figure 5.6 (right). That is due to the lower complexity of this sequence relative to *Foreman*, as explained earlier in regards to the analogous results for sender-driven streaming with diversity, shown in Figures 5.2 and 5.3.

Finally, the performances of the other sequences in these experiments are shown in Figure 5.7. They basically follow the same pattern as the performances of *Foreman* and *Mother & Daughter*, as seen from the figure.

We also studied the performance of the scheduling framework as a function of the quality of the network paths between the client and the servers. The results that we obtained in these experiments in the case of *Foreman* are shown in Figure 5.8 and are analogous to those presented in Figure 5.5 for the sender-driven case. In sum, it
can be seen from Figure 5.8 that as the quality of the network paths degrades there is an increasing performance gain of streaming Foreman with server diversity.

5.1.2 Proxy-Driven Streaming from the Network Edge

Two RaDiO streaming systems are employed in the experiments in this section. Sender-driven is a system that performs RaDiO scheduling of the packet transmissions at the media server as presented in Section 4.1.1. Proxy-driven is the system proposed in Section 4.2, which also performs RaDiO packet scheduling, but at the network edge using a proxy server. The Lagrange multiplier $\lambda$ is fixed for the entire presentation for both systems. We investigate distortion-rate performance
Figure 5.8: R-D performances for streaming Foreman from $M = 1, 2$ servers and different state transition models.

$(D, R_1, R_2)$ of the two systems, where distortion $D$ is measured in terms of the average Y-PSNR in dB as a function of the average transmission rates, $R_1$ and $R_2$, in Kbps in the forward directions on Paths 1 and 2, respectively. (See Section 4.2.1 for a more detailed description of the scenario under consideration.)

Path 1 is specified as follows. Packets transmitted on this network path are dropped at random, with a drop rate $\epsilon_{F_1} = \epsilon_{B_1} = \epsilon_1$. Those packets that are not dropped receive a random delay, where for the forward and backward delay densities $p_{F_1}$ and $p_{B_1}$ we use identical shifted Gamma distributions with parameters $(n_1, \alpha_1)$ and right shift $\kappa_1$, where $n_1 = 1$ node, $1/\alpha_1 = 25$ ms, and $\kappa_1 = 25$ ms for a mean delay of $\kappa_1 + n_1/\alpha_1 = 50$ ms and standard deviation $\sqrt{n_1}/\alpha_1 = 25$ ms. Path 2 is similarly specified with $\epsilon_{F_2} = \epsilon_{B_2} = \epsilon_2$, $n_2 = 1$ node, $1/\alpha_2 = 5$ ms, and $\kappa_2 = 5$ ms for a mean delay of $\kappa_2 + n_2/\alpha_2 = 10$ ms and standard deviation $\sqrt{n_2}/\alpha_2 = 5$ ms. Note that for Sender-driven the network path between the server and the client represents a concatenation of Paths 1 and 2. Therefore, the delay distribution for the concatenation is computed numerically as the convolution of the corresponding
Figure 5.9: Proxy-driven vs. sender-driven performance for Foreman: Y-PSNR (dB) vs. Transmission rate (Kbps) in the forward directions on Path 1 and Path 2, for $\epsilon_2 = 0, 5, 10, 15\%$ and $\epsilon_1 = 10\%$.

delay distributions for Paths 1 and 2.

First we compare the efficiencies of proxy-driven and sender-driven systems in terms of the transmission rates in the forward directions of Paths 1 and 2, as the packet loss rate increases on Path 2 and remains fixed on Path 1. Figure 5.9 shows the Y-PSNR of the Foreman sequence as a function of the transmission rates on Path 1 and 2, respectively, when $\epsilon_2 = 0\%, 5\%, 10\%, \text{ and } 15\%$, while $\epsilon_1 = 10\%$. It can be seen that the proxy-driven system outperforms the sender-driven system in all cases over the whole range of rates. In particular, Figure 5.9 (left) shows that as Path 2 degrades, the transmission rate on Path 1 of the proxy-driven system...
remains approximately constant (for any given Y-PSNR), while the transmission rate on Path 1 of the sender-driven system increases. Thus the gap between them, which is essentially zero when $\epsilon_2 = 0\%$, increases as Path 2 degrades. This is because as the quality of Path 2 deteriorates, the sender-driven system must increasingly retransmit information over Path 1, while the proxy-driven system does not need to retransmit information over Path 1 to accommodate information lost on Path 2. On the other hand, Figure 5.9 (right) shows that the transmission rate on Path 2 increases for both proxy-driven and sender-driven systems as Path 2 degrades. However, even when $\epsilon_2 = 0\%$, there is a positive gap in transmission rate between the two systems. This is because the sender-driven system suffers an additional loss of acknowledgements in the backward direction of Path 1. This shows up as additional rate in the forward direction of Path 2 due to retransmissions. Indeed, the ratio of transmission rates forming this performance gap appears to remain approximately constant as Path 2 degrades, indicating that this gap is due to losses in Path 1 rather than in Path 2.

Next we compare the efficiencies of proxy-driven and sender-driven systems, as packet loss increases on Path 1 and remains fixed on Path 2. Figure 5.10 shows the Y-PSNR of the Foreman sequence as a function of the transmission rates on Paths 1 and 2, respectively, when $\epsilon_1 = 0\%, 5\%, 10\%, \text{ and } 15\%$, while $\epsilon_2 = 10\%$. As in the previous figure, it can be seen that the proxy-driven system outperforms the sender-driven system over the whole range of rates in each of the cases under consideration. Furthermore, Figure 5.10 (right) shows that as Path 1 degrades, the transmission rate on Path 2 of the proxy-driven system remains approximately constant (for any given Y-PSNR), while the transmission rate on Path 2 of the sender-driven system increases. Thus the gap between them, which is essentially zero when $\epsilon_1 = 0\%$, increases as Path 1 degrades. This is because as Path 1 degrades, the
sender-driven system must increasingly retransmit information over Paths 1 and 2, while the proxy-driven system does not need to retransmit information over Path 2 to accommodate information lost on Path 1. On the other hand, Figure 5.10 (left) shows that the transmission rate on Path 1 increases for both proxy-driven and sender-driven systems as Path 1 degrades. However, even when $\epsilon_1 = 0\%$, there is a positive gap in transmission rate between the two systems. This is because the sender-driven system suffers additional loss in both the forward and backward directions of Path 2. This shows up as additional rate in the forward direction of Path 1 due to retransmissions.
<table>
<thead>
<tr>
<th>Packets per DU</th>
<th>Path 1</th>
<th>Path 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender-driven</td>
<td>$1/[(1 - \epsilon_{F1})(1 - \epsilon_{F2})(1 - \epsilon_{B2})(1 - \epsilon_{B1})]$</td>
<td>$1/[(1 - \epsilon_{F2})(1 - \epsilon_{B2})(1 - \epsilon_{B1})]$</td>
</tr>
<tr>
<td>Proxy-driven</td>
<td>$1/(1 - \epsilon_{F1})$</td>
<td>$1/[(1 - \epsilon_{F2})(1 - \epsilon_{B2})]$</td>
</tr>
<tr>
<td>Ratio</td>
<td>$1/[(1 - \epsilon_{F2})(1 - \epsilon_{B2})(1 - \epsilon_{B1})]$</td>
<td>$1/(1 - \epsilon_{B1})$</td>
</tr>
</tbody>
</table>

Table 5.3: Number of packets (per data unit) transmitted in forward direction over Paths 1 and 2 for sender-driven and proxy-driven systems.

We find here that the performance of both proxy-driven and sender-driven systems on both Paths 1 and 2 can be accurately predicted from the distortion-rate function of the source and the values of $\epsilon_1$ and $\epsilon_2$. In particular, since our sender-driven system continues to transmit packets until it receives an acknowledgement, which it receives with probability $(1 - \epsilon_{F1})(1 - \epsilon_{F2})(1 - \epsilon_{B2})(1 - \epsilon_{B1})$ for each transmitted packet, the sender-driven system transmits on average $1/[(1 - \epsilon_{F1})(1 - \epsilon_{F2})(1 - \epsilon_{B2})(1 - \epsilon_{B1})]$ packets per data unit over Path 1. Only fraction $(1 - \epsilon_{F1})$ of these survive to cross Path 2, so the average rate on Path 2 is $1/[(1 - \epsilon_{F2})(1 - \epsilon_{B2})(1 - \epsilon_{B1})]$ packets per data unit. Similarly, the proxy-driven system transmits on average $1/[(1 - \epsilon_{B1})(1 - \epsilon_{F1})]$ requests per data unit in the backward direction of Path 1. Only fraction $(1 - \epsilon_{B1})$ of these reach the server, so the proxy-driven system transmits on average $1/(1 - \epsilon_{F1})$ packets per data unit in the forward direction of Path 1. Likewise the proxy-driven system transmits on average $1/[(1 - \epsilon_{F2})(1 - \epsilon_{B2})]$ packets per data unit in the forward direction of Path 2. These factors are summarized in Table 5.3.

To confirm that these factors accurately predict performance, we normalize the transmission rates for all the graphs in Figures 5.9 and 5.10 by the appropriate factors in the table (for example, we divide all the proxy-driven rates for Path 1 by
Figure 5.11: Y-PSNR (dB) of the Foreman sequence vs. Normalized rate (Kbps) on Paths 1 and 2 for proxy-driven and sender-driven systems.

$1/(1 - \epsilon_{F1})$ and we plot them (in gray) in Figure 5.11 along with the distortion-rate function of the source (in bold). It can be seen that all graphs lie essentially on top of the distortion-rate function for the source, meaning that we can effectively synthesize the graphs in Figures 5.9 and 5.10 by multiplying the rate-distortion function of the Foreman sequence by the factors in Table 5.3. The same correspondence is also obtained for the other sequences, as seen from Figure 5.12.

5.1.3 Streaming with Rich Acknowledgements

Two RaDiO streaming systems are employed in the experiments in this section. Conv. ACK is a system that performs RaDiO scheduling using conventional acknowledgments as presented in Section 4.1.1. Rich ACK is the system presented in Section 4.3, which also performs RaDiO packet scheduling, but using rich acknowledgments. The Lagrange multiplier $\lambda$ is fixed for the entire presentation for both systems.
Figure 5.12: Y-PSNR (dB) of the (top left) Carphone, (top middle) Container, (top right) Mother & Daughter, (bottom left) News, and (bottom right) Salesman sequences vs. Normalized rate (Kbps) on Paths 1 and 2 for proxy-driven and sender-driven systems.

The forward and backward channels between the server and the client are specified as follows. Packets transmitted on these channels are dropped at random, with drop rates \( \epsilon_F \) and \( \epsilon_B \), respectively. Those packets that are not dropped receive a random delay, where for the forward and backward delay densities \( p_F \) and \( p_B \) we use identical shifted Gamma distributions with parameters \( (n, \alpha) \) and right shift \( \kappa \), where \( n = 1 \) node, \( 1/\alpha = 25 \) ms, and \( \kappa = 25 \) ms for a mean delay of \( \kappa + n/\alpha = 50 \) ms and standard deviation \( \sqrt{n}/\alpha \approx 25 \) ms.

First, we compare the efficiencies of the two streaming systems in terms of the transmission rate on the forward channel, as the packet loss rate increases on the backward channel and remains fixed on the forward channel. Figure 5.13 shows the average Y-PSNR for the Foreman sequence as a function of the transmission
Figure 5.13: Y-PSNR (dB) vs. Transmission rate (Kbps) for streaming Foreman with rich and conventional acknowledgements.

rate on the forward channel, when $\epsilon_B = 0\%$, 5\%, 10\%, and 15\%, while $\epsilon_F = 10\%$. It can be seen that as the backward channel degrades, the transmission rate on the forward channel of Rich ACK remains approximately constant (for any given Y-PSNR), while the transmission rate of Conv. ACK increases. Thus the gap between them, which is essentially zero when $\epsilon_B = 0\%$, increases as the packet loss rate on the backward channel increases. This is because as the backward channel degrades, Conv. ACK increasingly and unnecessarily retransmits information over the forward channel, due to loss of acknowledgment packets on the backward channel. Note that Rich ACK avoids this by acknowledging every arriving media packet multiple times by consecutive rich acknowledgments.

Next, we extend our performance analysis from the previous section to the scenario considered here. In particular, as also argued in Section 5.1.2, since Conv. ACK
continues to transmit packets until it receives an acknowledgment, which it receives with probability \((1 - \epsilon_F)(1 - \epsilon_B)\) for each transmitted packet, then Conv. ACK transmits on average \(1/[(1 - \epsilon_F)(1 - \epsilon_B)]\) packets per data unit over the forward channel. On the other hand, Rich ACK transmits on average only \(1/[(1 - \epsilon_F)(1 - \epsilon_B^K)]\) packets per data unit on the forward channel, where \(K\) is the number of rich acknowledgments sent after the data unit arrives at the client. Note that for the range of values for \(\epsilon_B\) considered here, this can be approximately written as \(1/(1 - \epsilon_F)\) even for very small values of \(K\). Therefore, rich acknowledgments essentially cancel out the effect of packet loss on the backward channel.

![Figure 5.14: Y-PSNR (dB) vs. Normalized transmission rate (Kbps) for streaming (top left) Foreman, (top middle) Mother & Daughter, (top right) Carphone, (bottom left) Container, (bottom middle) News, and (bottom right) Salesman with rich and conventional acknowledgements.](image)

To confirm that these factors accurately predict performance, we normalize the transmission rates on the forward channel for all the graphs in Figure 5.13 with
$1/[(1 - \epsilon_F)(1 - \epsilon_B)]$ for Conv. ACK and with $1/(1 - \epsilon_F)$ for Rich ACK and we plot them (in gray) in Figure 5.14 (top left) along with the distortion-rate function of the source (in bold). It can be seen that as in the previous scenario also here all normalized graphs lie essentially on top of the distortion-rate function for the source. Therefore, we can effectively synthesize the graphs in Figure 5.13 by multiplying the rate-distortion function for the Foreman sequence by the appropriate factors for Conv. ACK and Rich ACK. The same conclusion regarding the prediction accuracy of these factors also holds in the case of the other sequences, as seen from the rest of the graphs in Figure 5.14.

![Graph showing rate of acknowledgements vs. transmission rate](image)

Figure 5.15: Rate of acknowledgements (Kbps) vs. Transmission rate (Kbps) on the forward channel for streaming Foreman.

Finally, we study the transmission rates of acknowledgments for both systems which simply comprise the amount of acknowledgment data sent to the client during a streaming session. In the case of Conv. ACK each acknowledgment packet contains only a header and no payload. On the other hand, a rich acknowledgment packet sent at transmission opportunity $t_i$ consists of both a header and a payload of size $w_i$.
bits, where \( w_i \) is the number of data units in the transmission window of the server at \( t_i \), as explained in Section 4.3. In our experiments, the header size in both systems is set to 320 bits. In Figure 5.15, we show the transmission rates of Conv. ACK and Rich ACK on the backward channel as a function of the transmission rate on the forward channel in the case of Foreman and \( \epsilon_F = \epsilon_B = 10\% \). Only the low range of available transmission rates on the forward channel is considered in this experiment. It can be seen from the figure that for transmission rates smaller than 60 Kbps, the rate of rich acknowledgements exceeds the corresponding rate of acknowledgments for Conv. ACK. However, as the rate on the forward channel is increased beyond 60 Kbps, the rate on the backward channel of Rich ACK remains constant, while the corresponding rate of acknowledgements for the Conv. ACK system increases in proportion to the forward rate. Therefore, Rich ACK provides savings in transmission bandwidth on the backward channel relative to Conv. ACK, which can be beneficial in certain scenarios and types of networks.

5.2 Experiments with Internet Packet Traces

Here, we present streaming results from experiments where Internet behaviour in terms of packet delay and packet loss is simulated using actual loss and delay traces recorded on the Internet. We only consider the scenario of sender-driven streaming over a single network path. It suffices to do so because this is the most commonly encountered scenario in the Internet today. Moreover, the analysis and the conclusions that will be presented here based on the results obtained for this scenario are such that they apply to the broader context of Internet streaming in general.

Two sender-driven streaming systems are examined in the experiments in this section. ARQ is a system that does not take into consideration the distortion informa-
tion of individual packets when making transmission decisions. It employs time-out driven retransmission for packets that have not been acknowledged by the receiver. Specifically, at each transmission opportunity data units are (re)transmitted by ARQ in a GOP order based on their delivery deadline. A data unit is considered for retransmission only if its arrival at the receiver has not been acknowledged within RTO\(^t\) time after its last transmission. The second streaming system under comparison here is denoted RaDiO as it performs RaDiO scheduling of the packet transmissions at the media server as presented in Section 4.1.1. The Lagrange multiplier \(\lambda\) is fixed for the entire presentation for the RaDiO system.

5.2.1 Channel Estimation

The RaDiO system still employs statistical models of the communication channels in order to perform packet scheduling as described in Chapter 3. However, since Internet behaviour in this section is reproduced based on collected channel traces, we assume that a priori the statistics of the packet delay and packet loss experienced in each direction are not known. Therefore, RaDiO estimates online the parameters of the statistical models for the forward and the round-trip communication channels. The estimation procedure works as follows.

At each transmission opportunity, at least one packet is sent to the receiver. If there are no outgoing data packets at present, then a small probe packet, comprising only a packet header and no payload, is sent instead. The receiver responds to every incoming packet on the forward channel by sending immediately an acknowledgement packet on the backward channel. The returning acknowledgement packet contains the time of the arrival of the incoming packet on the forward channel as

\(^t\)The retransmission time-out interval.
well as its sequence number. The sender then collects the information provided by returning acknowledgment packets in terms of packet delay experienced in the forward and round-trip directions. This information is used by the sender to estimate the parameters of the statistical models characterizing the delay in each direction. Furthermore, missing sequence numbers of packets are used to estimate the packet loss rate in each direction.

As in Sections 5.1.2 and 5.1.3 of this chapter, the RaDiO system assumes that packet delays are statistically characterized according to a shifted Gamma distribution with parameters \((n, \mu)\) and right shift \(\kappa\) [187]. For the first parameter, that signifies the number of routers or queues on the network path between the sender and the receiver, we can safely assume that \(n_F = n_R = 1\). That is because in the Internet the statistics of the packet delay and packet loss experienced on a network path are typically governed by the behaviour of the most congested queue along the path. The other two parameters of the shifted Gamma distribution are estimated as follows. Let \(FTT\) and \(RTT\) be respectively the measured forward-trip and round-trip times associated with the last received acknowledgment packet. Then, the running estimates of the right shift and the mean, \(\tilde{\kappa}_F\) and \(\tilde{\mu}_F\), in the case of the forward channel are updated as

\[
\tilde{\kappa}_F = \min(\tilde{\kappa}_F, FTT),
\]

\[
\tilde{\mu}_F = (1 - \tau) \tilde{\mu}_F + \tau FTT,
\]

where the factor \(\tau \in [0,1]\) helps to filter-out measurement noise present in \(FTT\). The same expressions from above applied to the round-trip time measurement are employed to update \(\tilde{\kappa}_R\) and \(\tilde{\mu}_R\) for the round-trip channel.

The ARQ system also estimates on-the-fly the mean and the standard deviation
of the round-trip time using the procedure\footnote{The algorithm is commonly known as Exponentially Weighted Moving Average (EWMA).} described above. Then, these quantities are used to set the RTO value employed by ARQ to $\bar{\mu}_R + 3\bar{\sigma}_R$. This time-out value is frequently used in ARQ systems, e.g., TCP [256].

Representatives of two types of network traces are employed in the experiments: (1) network traces that do not exhibit packet losses, and (2) network traces that exhibit packet losses uniformly over the whole duration of the trace collection procedure. We believe that conducting our experiments over network traces featuring different characteristics will provide us with a comprehensive understanding of the performance of our framework over a range of diverse networking conditions and situations. We continue this section by examining first experimental results involving a network trace representing the first category.

### 5.2.2 Traces without Packet Losses

The Internet trace that we utilize here has been collected during a course of 5 hours between a machine at Stanford University and a personal computer in a residence in Mountain View, California which connected to the Internet via a cable modem. The collection process started around 5:00pm on a working day. There were 14 Internet hops between the two machines. The downlink bandwidth from the Stanford host to the cable modem was estimated, via ftp file transfer, to be in excess of 1.5 Mbits per second. The uplink bandwidth (in the opposite direction) was approximately 230 Kbits per second. The trace collection process sent 20 packets per second in each direction (downlink and uplink) of size 32 bytes (IP header, UDP header, and sequence number), for a total of 6.4 Kbits per sec. The probe packets thus utilized less than 2.8\% of the available upstream bandwidth, and 0.42\% of the available
downstream bandwidth. Therefore, it is reasonable to assume that they did not affect the loss and delays experienced by themselves in either direction, i.e., they did not have a self-congestion effect. For more details on the methodology that was employed to collect the trace and the related issues, the reader is referred to [265]. We acknowledge the first author in [265] for providing the trace to us.

![Figure 5.16: CDF of the packet delay on the (a) Downlink and (b) Uplink.](image)

The packet delay values recorded during the trace collection process on each channel, downlink and uplink, are quite small. The same observation applies to the measured packet loss rate. In Figure 5.16 we show the statistical characterization of the delay experienced in each direction, in terms of the CDF function of the delay. It can be seen from the figure, for example, that the downlink delay is 95% of the time smaller than 100 ms and that the uplink delay is more than 95% of the time smaller than 50 ms. Furthermore, on the downlink we observe a packet loss rate of 0.0418%, while in the uplink direction this number is even lower 0.0167%. Hence, it can be concluded that for all practical purposes both communication channels provide a very good quality in terms of packet loss and delay.
In Figure 5.17 we examine the Y-PSNR performance in dB of the two systems under comparison for streaming the Foreman sequence as a function of the transmission rate in Kbps on the forward (downlink) channel. For both systems the time interval between transmission opportunities $T$ is set to 50 ms, which is the time spacing at which network measurements were collected via packet probes in the trace collection procedure, as described earlier. Furthermore, the play-out delay $\Delta$ for the video clip is 600 ms. It can be seen from the figure that ARQ and RaDiO provide almost the same performance in the two upper thirds of the range of transmission rates under consideration. Specifically, for transmission rates greater than 90 Kbps the two systems perform quite alike with RaDiO outperforming ARQ with only an insignificant margin. That is because of two reasons. First, no retransmissions of packets occur in either system, since the communication channels are quite good, meaning packet losses are very rare and packet latencies are very low, as discussed earlier. Retransmissions would certainly have a stronger impact on the performance of ARQ, as this
system does not distinguish between packets based on their distortion importance so every outstanding packet is retransmitted equally likely.

The second reason is the fact that over the range of transmission rates where the performances of the two systems are quite similar, no packet dropping, i.e., omission\(^5\), at the sender occurs in ARQ in order to account for bandwidth variability. Specifically, in this range of rates ARQ employs stream switching via SP/SI frames, rather than packet dropping at the sender, in order to reduce its transmission rate. As in the case of retransmissions, packet dropping would certainly have a stronger effect on the performance of ARQ due to the same reason explained earlier, i.e., the fact that ARQ is oblivious to packet distortion information. This is in fact obvious from the performances of ARQ and RaDiO that correspond to the lower third of transmission rates under consideration in Figure 5.17. For transmission rates below 90 Kbps no stream switching can occur, so ARQ is forced to work only with a single encoding and therefore it has to employ packet dropping in order to reduce its transmission rate further. It can be seen from Figure 5.17 that in this range of rates RaDiO provides a much more graceful degradation in performance as the transmission rate is reduced toward its low end. For example, for transmission rate of 77 Kbps there is a performance difference between the two systems that reaches almost 4.5 dB, which is quite significant.

This apparent deficiency of ARQ can be overcome by enhancing this system as follows. Instead of allowing ARQ to stream from a full source encoding, we preprocess the encoding ahead of time, i.e., before streaming. Specifically, by dropping certain packets from the encoding we can reduce the source rate of the encoding in a rate-distortion optimal way according to the available transmission rate. For that

\(^5\)Also known as source pruning.
purpose we could use an optimal pruning algorithm such as the RaDiO system itself. The original ARQ system is then used to stream from the processed encoding. We denote such an enhanced ARQ system as RaDiO ARQ. Its performance is shown in Figure 5.17 along with those of the original ARQ and RaDiO. It can be seen from the figure that now the performances of the ARQ system with preprocessing and of RaDiO are aligned over the whole range of transmission rates under consideration.

The last results presented in this section answer the question how the performances of the two systems are affected by late loss. In particular, a late loss of a packet occurs when the packet arrives at the receiver after its delivery deadline. We noticed that the packet delays experienced on the downlink during the last 1/5 of the trace duration exhibit larger values on the average relative to the rest of the trace. Moreover, there is a larger amount of variation of packet delay values for this segment of the trace. This increased activity on the downlink is because the last 1/5 of the trace roughly corresponds to the period of the day when people are back from work and use their Internet connections from home in the evening to surf the web and download (or stream) data. Therefore, we repeated our streaming experiments from before but only over this last section of the trace. To make the late loss have an impact on performance, we employed two smaller values for the play-out delay \( \Delta \) in the experiments.\(^4\) Specifically, we recorded the performances of ARQ and RaDiO for \( \Delta = 400 \) ms and for \( \Delta = 200 \) ms. These results are shown in the following figure.

It can be seen from Figure 5.18 that late loss has equal impact on the performance of both ARQ and RaDiO. This is expected since late loss does not incur retransmissions of packets in this scenario. Therefore, the packet distortion information that

\(^4\)Even with the increased activity during this period, packet delays on the downlink quite rarely exceed 500 ms.
Figure 5.18: R-D performance for streaming Foreman over a network trace without packet losses and $\Delta = 200$ ms and 400 ms.

RaDiO can exploit to its advantage is not helpful here. For example, it can be seen from the figure that for the case of $\Delta = 400$ ms, the performances of ARQ and RaDiO have equally degraded to below 34 dB at a transmission rate of 300 Kbps, relative to the corresponding performances shown in Figure 5.17. When the play-out delay is additionally reduced to 200 ms, the performances of the two streaming systems degrade uniformly even further for roughly 1 dB. In the next section, we show how packet losses can have a different impact on the performances of ARQ and RaDiO.

5.2.3 Traces with Uniformly Distributed Packet Losses

The representative Internet trace that we utilize here has been collected during a course of a few hours between a machine at Stanford University and another computer at Erlangen University in Germany. The path Stanford→Erlangen is the downlink channel, while the return path Erlangen→Stanford serves as the uplink channel. The trace collection process sent a probe packet every 20 ms in each direc-
tion (downlink and uplink) of size 32 bytes (IP header, UDP header, and sequence number). The trace has been provided to us courtesy of Y. Liang [190, 191].

The statistical characterization of the delay collected in this trace in both directions is shown in Figure 5.19. It can be seen from the figure that the delay in each direction exhibits very little variation. In particular, most of its values range in the neighbourhood of 320 ms for the forward (downlink) channel and in the neighbourhood of 70 ms for the backward (uplink) channel. Furthermore, on the downlink uniformly distributed packet losses are observed at a rate of 9.59%. On the other hand, no lost packets are recorded on the uplink.

In Figure 5.20 we examine the Y-PSNR performance of ARQ and RaDiO for streaming Foreman over this network trace. The time interval $T$ between successive transmission opportunities is set to 40 ms for both streaming systems. Furthermore, the play-out delay $\Delta$ for the video clip is 600 ms. It can be seen from the figure that now there is a significant difference in performance between the two systems over the whole range of transmission rates under consideration. For example, in the two
upper thirds of the transmission rate range, RaDiO outperforms ARQ with a margin that ranges between 1.2 - 1.7 dB. In essence, the performance gains of RaDiO in this case are due to the fact that RaDiO, contrary to ARQ, transmits more important packets earlier and frequently repeatedly on multiple occasions, without waiting (long enough) first for a returning acknowledgement packet that may potentially arrive due to previous transmissions. This increases the robustness of the more important packets to packet losses during transmission and therefore ensures their timely delivery to the client.
Chapter 6

Conclusions

In this thesis, we have addressed the challenges imposed by rate-distortion optimized streaming and modeling of packetized video. The research presented here overcomes the current limitations of rate-distortion optimized packet scheduling, such that these techniques can be applied to video streaming, and a better end-to-end performance can be achieved. Furthermore, a broader range of streaming problems that have not been considered earlier in the context of rate-distortion optimized packet scheduling has been addressed in the thesis.

An advanced framework for rate-distortion optimized scheduling of packetized video has been proposed in the thesis. The framework comprises a distortion model that for the first time takes into account error-concealment at the receiver and an iterative algorithm for computing rate-distortion optimized schedules for video packets in the presence of error-concealment.

We have formulated within the optimization framework a few advanced streaming scenarios such as streaming with diversity, where diversity can be achieved through streaming either over multiple network paths or from multiple media servers, proxy-driven streaming from the network edge, and streaming with rich acknowledgements. It has been shown through extensive simulation experiments that an improved end-to-end performance can be achieved when these scenarios are addressed in a rate-distortion optimized way. Furthermore, for two of the scenarios, the experimental results were accurately predicted through analysis. In particular, we have proposed a
simple analytical framework that relates the performance of rate-distortion optimized streaming to the packet loss probabilities of the underlying communication channels. The analysis shows that the optimization framework achieves the theoretical rate-distortion limit as provided by the equivalent packet erasure channels for the scenario under consideration and the rate-distortion characteristics of the packetized video source.

Finally, we have examined the performance of the optimization framework in the context of streaming over Internet traces featuring collected packet delays and packet losses experienced over network paths. It has been shown that depending on the quality of the network path in terms of packet loss and delay, rate-distortion optimized packet scheduling may or may not provide performance gains over simpler transmission techniques, such as time-out driven (re)transmission (ARQ), where no distortion information is taken into consideration when making transmission decisions. In addition, it was shown how ARQ can be enhanced for bandwidth adaptation via the optimization framework.
Glossary

Acronyms

CIF Common Intermediate Format
MPEG Motion Picture Experts Group
FGS Fine Granular Scalability
GOP Group of Pictures
ITU International Telecommunication Union
H.26x A series of video encoding standards by the ITU (x ∈ {1, 3, 4})
RaDiO Rate-Distortion Optimization/Optimized
ACK Acknowledgement packet
ARQ Automatic Repeat reQuest
NAK Negative acknowledgement packet
DAT Data Unit packet
SNR Signal-to-Noise Ratio
PSNR Peak Signal-to-Noise Ratio
MSE Mean Squared Error
IP Internet Protocol
TCP Transmission Control Protocol
RTP Real-Time Protocol
ISA Iterative Sensitivity Adjustment
FTT Forward Trip Time
BTT Backward Trip Time
RTT Round Trip Time

List of Symbols

\( R \) Rate
\( D \) Distortion
\( J \) Lagrangian
\( \lambda \) Lagrange multiplier
\( \pi \) Transmission policy
\( \pi \) Vector of transmission policies
\( L \) Number of data units in a presentation
\( \Pi \) Family of transmission policies
\( \mathcal{N}_c^{(l)} \) Concealment set for data unit \( l \)
\( t_{DTS, l} \) Delivery deadline for data unit \( l \) (Decoding Time Stamp)
\( B_l \) Size of data unit \( l \) in bytes
\( \Delta d_i^{(l_1)} \) Reduction in reconstruction distortion if data unit \( l \) is concealed with data unit \( l_1 \in \mathcal{N}_c^{(l)} \)
\( K \) Number of states of a Hidden Markov model
\( \mathcal{P} \) State transition probability matrix for an Hidden Markov model
\( p \) \hspace{1cm} \text{Probability density function of the packet transmission delay}

\( M \) \hspace{1cm} \text{Number of network paths}

\( \gamma_m \) \hspace{1cm} \text{Relative cost of transmitting over network path } m

\( \rho(\pi) \) \hspace{1cm} \text{Expected cost for policy } \pi

\( e(\pi) \) \hspace{1cm} \text{Expected error for policy } \pi

\( A(l) \) \hspace{1cm} \text{Set of ancestor data units for data unit } l

\( D(l) \) \hspace{1cm} \text{Set of descendant data units for data unit } l

\( D_0 \) \hspace{1cm} \text{Reconstruction distortion when no data units are received}

\( S_l \) \hspace{1cm} \text{Sensitivity to losing data unit } l

\( T \) \hspace{1cm} \text{Time interval between transmission opportunities}

\( N \) \hspace{1cm} \text{Number of transmission opportunities}

\( \mathcal{W} \) \hspace{1cm} \text{Window of transmission opportunities}

\( w_{\text{lag}} \) \hspace{1cm} \text{Lagging edge of the transmission window}

\( w_{\text{lead}} \) \hspace{1cm} \text{Leading edge of the transmission window}

\( a_{jm} \) \hspace{1cm} \text{Action at transmission opportunity } t_j \text{ on network path } m

\( C_{jm} \) \hspace{1cm} \text{Channel state at transmission opportunity } t_j \text{ on network path } m

\( \nu \) \hspace{1cm} \text{Playback speed}

\( \Delta \) \hspace{1cm} \text{Play-out delay}

\( Y \) \hspace{1cm} \text{Luminance component of a video frame}

\( \kappa \) \hspace{1cm} \text{Right shift of a Gamma distribution for the transmission delay}

\( n \) \hspace{1cm} \text{Number of hops between the sender and the receiver}
\[
\begin{align*}
1/\alpha & \quad \text{Mean queueing delay over a network hop} \\
\mu & \quad \text{Mean value of the packet transmission delay} \\
\sigma & \quad \text{Standard deviation of the packet transmission delay}
\end{align*}
\]
Bibliography


[52] M. Khansari, A. Zakauddin, W.-Y. Chan, E. Dubois, and P. Mermelstein, “Approaches to layered coding for dual-rate wireless video transmission,” in


[67] M. van der Schaar and H. Radha, “Motion compensation fine-granular scala-


[126] R. Ahlswede, “The rate-distortion region of a binary source for multiple de-


[128] Z. Zhang and T. Berger, “Multiple description source coding with no excess 
marginal rate,” IEEE Trans. Information Theory, vol. 41, no. 2, pp. 349–357, 

[129] J. Batllo and V. A. Vaishampayan, “Asymptotic performance of multiple de-
scription transform codes,” IEEE Trans. Information Theory, vol. 43, no. 2, 

[130] V. K. Goyal, J. Kovačević, and M. Vetterli, “Multiple description transform 
coding: robustness to erasures using tight frame expansions,” in Proc. Int’l 

[131] P. A. Chou, S. Mehrotra, and A. Wang, “Multiple description decoding of 
overcomplete expansions using projections onto convex sets,” in Proc. Data 

[132] V. K. Goyal, “Multiple description coding: Compression meets the network,” 


[139] Y. Su, T. Tao, J. Lu, and J. Wang, “Channel-optimized video transmission over


[203] Z. Miao and A. Ortega, “Scalable proxy caching of video under storage con-


