ENHANCING THE MULTIMEDIA EXPERIENCE IN EMERGING NETWORKS

A Thesis
Presented to
The Academic Faculty

by

Ali C. Begen

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2006

ENHANCING THE MULTIMEDIA EXPERIENCE IN EMERGING NETWORKS

To my family,

Cevdet, Sündüz, and Mehmet Atilla.

ACKNOWLEDGEMENTS

This thesis would not have been what it is without the encouragement and generous support from many individuals. I am grateful to every person who has offered help to me and contributed in some way to the process leading to my dissertation.

It is difficult to overstate my gratitude to my advisor, Prof. Yucel Altunbasak. With his enthusiasm, his inspiration, his guidance, and his great efforts to explain things clearly and simply, he has been a great mentor throughout my Ph.D. study. I would like to thank him for all of the advice he has passed along. I have also been fortunate enough to have a great thesis committee: Prof. Russell M. Mersereau, Prof. Ghassan AlRegib, Prof. Biing Hwang (Fred) Juang and Prof. Ozlem Ergun. I thank them for being very supportive of my research and for their constructive critiques that have greatly contributed to my thesis.

Since I first came to Georgia Tech in 2001, the campus and its tenants offered me a pleasant, friendly and motivating work environment. Every communications, networking and signal processing course I have taken was interesting, informative and inspirational. I want to thank the faculty of School of Electrical and Computer Engineering and College of Computing for these courses. I am also grateful to all my colleagues and friends at Georgia Tech. Particularly, I would like to thank the members of the Multimedia Computing and Communications Lab, as well as the members of the Center for Signal and Image Processing for the five wonderful years I spent in Atlanta.

Throughout my undergraduate years at Bilkent University and graduate years at Georgia Tech, I have greatly benefited from numerous insightful discussions, whether it was research related or not, with Mehmet Umut Demircin. I thank him for our 16-year friendship, his always constructive and timely advice, and continued moral support.

During Summer and Fall 2004, I interned with the Standards Engineering Group at Qualcomm Inc. in San Diego, CA. I would like to thank my managers, and particularly my mentor, Hector Vayanos, for giving me the opportunity to work in an exceptional industrial environment. This internship exposed me to the cutting-edge developments in the wireless communications technology and helped me gain practical skills in solving real-life problems. It was a rewarding experience for me.

On a different note, I would like to thank the computer game developers for creating the FPS games, the hardware engineers for making it possible for us to play those games, and my friends who played those games with me. I cannot imagine how difficult it would be to restore myself and refocus on my thesis without first relaxing with those games.

Last but not least, I want to thank my dad, mom and brother. I am forever indebted for their understanding, endless patience and encouragement when it was most required. It is to them that I dedicate this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

As multimedia processing and networking technologies, products and services evolve, the number of users communicating, collaborating and entertaining over the IP networks is growing rapidly. With the emergence of pervasive and ubiquitous multimedia services, this proliferation creates an abundant increase in the amount of the Internet backbone traffic. This brings the problem of efficient transmission of real-time and time-sensitive media content to the fore. Effective multimedia services demand appropriate application-specific and media-aware solutions, without which the full benefits of such services will not be realized. Poor approaches often lead to system performance degradations such as unacceptable presentation quality perceived by the users, possible network collapses due to the high-bandwidth nature of the multimedia applications, and poor performance observed by other data-oriented applications due to the unresponsiveness of multimedia flows.

From a networking perspective, traditional approaches consider the application data as "sacred" and do not differentiate any part of it from the rest. While this keeps the data-delivery mechanisms, namely, the transport-layer protocols, as plain as possible, it also precludes these mechanisms from interpreting the media content and tailoring their actions according to the importance of the content. Given that this naive approach cannot satisfy the specific needs of each and every one of the today's emerging applications ranging from videotelephony to video-on-demand, from distance education to telemedicine, from remote surveillance to online video gaming, the study of Multimedia Transport Protocols (MMTP) is overdue.

An MMTP solution basically integrates the multimedia content information into the responsible data-delivery mechanisms along with the requirements of the invoking application and network characteristics to deliver the highest level of service quality. In other words, an MMTP solution offers a *unified* environment where all cooperating protocol components interact with each other and make the best use of this collaboration to fulfill their respective

duties.

The focus of this thesis is on the design and evaluation of a set of end-to-end and system-level MMTP solutions for scalable, reliable, and high quality multimedia services in ever-changing, complex and heterogeneous computing and communication environments.

The main contributions of this thesis are:

- To develop an optimal overlay-based multi-path transmission framework for streaming error-resilient videos in low-delay video applications,
- To analyze the performance of single/multi-hop and single/multi-path transmission methods for streaming high-resolution videos in mesh networks,
- To formulate a mathematical framework that models on-demand video delivery from multiple content servers, and to develop a rate-distortion optimal packet scheduling algorithm for a multi-server streaming system,
- To investigate proxy-based solutions to overcome the problems associated with packet loss, large delay and delay jitter in interactive video applications,
- To characterize the packet dynamics in networked-video applications, to propose models for packet delay, to develop accurate delay and delay-boundary prediction methods, and to interpret the correlation between the packet delay and packet loss events,
- To engineer media-aware and network-adaptive error-control methods, and to lay out a framework that computes the optimal actions for error control.

In addition to providing the mathematics behind these solutions, a considerable emphasis is also given to the real-time implementation issues.

We envision a future that will involve richer and more demanding multimedia services, which will be available anywhere and anytime, and accessible from a variety of devices. We anticipate that this thesis will play an enabling role and help establish a solid foundation for the next generation multimedia services, which will doubtlessly have a great influence on how we communicate and collaborate, teach and learn, conduct business and entertain.

CHAPTER I

INTRODUCTION

The growth in the number of users that communicate, collaborate and entertain over the Internet is largely due to two main factors: On one end, the emerging technologies in media-creation, coding and processing techniques are continually driving new engaging multimedia services into the market. On the other end, the proliferation of pervasive and ubiquitous computing platforms is fueling the demand for inter-connectivity and enabling new users equipped with different bandwidth and processing-power capabilities to connect to the Internet every day. Together with the assorted multimedia applications, this creates a tremendous diversity. Not surprisingly, tackling this diversity and satisfying the particular requirements of each and every one of these applications with existing tools and protocols is difficult. Timely delivery of real-time media being a critical task, it is well understood that new application-specific, media-aware and network-adaptive solutions have to be developed to enhance the multimedia experience for the users.

In this interdisciplinary thesis, we make use of the synergy between multimedia signal processing and networking, and propose several solutions that we refer to as Multimedia Transport Protocols (MMTP). Before we get into the details of MMTP, we first give an introduction to multimedia networking along with its motivation, applications, and problems in Section 1.1[1]. We present illustrative examples and discuss the key ideas in the development of MMTP solutions in Section 1.2. In Sections 1.3 and 1.4, we briefly summarize the state-of-the-art in multimedia networking and the contributions of this thesis, respectively. Finally, in Section 1.5, we provide the thesis outline and list the problems studied in the remaining chapters.

[1]In general, “multimedia data” refers to audio, speech, still pictures, video, animation and graphics data. Within the context of this thesis, because of our focus on networked video, we often use it to refer to visual data. However, in most instances, the proposed solutions are also applicable to other media types as well.

1.1 Background on Multimedia Networking

In the last decade, multimedia services over the IP networks have achieved significant success, however, this success has remained limited for many end-users due to the pitfalls of the best-effort Internet. Unlike the conventional data services such as file transfer, e-mail and Web browsing, multimedia services demand almost-constant bandwidth, and rigid bounds on packet loss and delay to maintain an acceptable level of service quality. To this effect, traditional transport protocols and error-control methods that are primarily designed for delay-insensitive services often fall short to satisfy such requirements. For example, Transmission Control Protocol (TCP) transmits the packets in order, regardless of their delivery deadlines, and employs a rigorous window-based congestion control mechanism [1]. This behavior often causes many packets to be *obsolete* by the time they are received at the client. Furthermore, when a packet is lost, TCP cannot advance its congestion window and reduces its rate. A possible timeout and subsequent slow-start phase with retransmissions, as depicted in Figure 1, further delay the transmission of future packets, forcing the client to stop playing out the video and to re-buffer some data. These stop-and-buffer periods limit the client's ability to experience a continuous video. A TCP-driven real-time video application may avoid these annoying interruptions by pre-buffering a large amount of video content before starting to play out the video [2, 3].

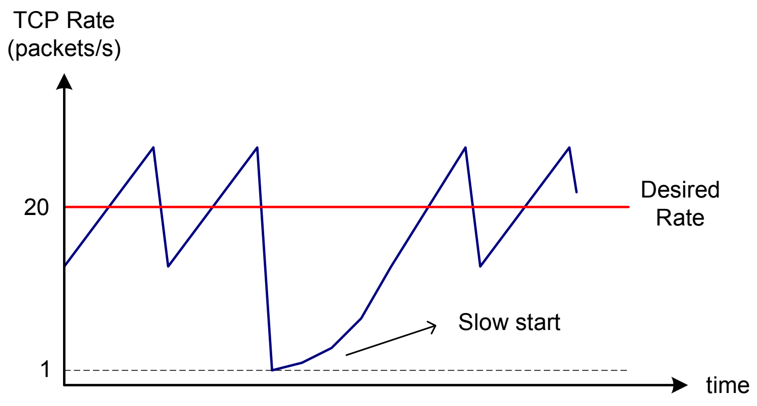


Figure 1: Rate variation in TCP.

Fortunately, real-time video applications are loss tolerant in the sense that full reliability is generally not essential. Packet losses at acceptable levels usually result in small glitches that are not highly visible to the clients. For this reason, application developers often prefer User Datagram Protocol (UDP) [4] over TCP. UDP is a lightweight transport protocol with a minimalist service model that provides an unreliable data transfer service. However, similar to TCP, UDP neither offers any timely-delivery guarantees nor has a concern with the packet deadlines. Hence, the UDP service model is also not the best solution for delay-sensitive applications. More importantly, the lack of a congestion control mechanism in UDP is a serious threat to the stability and scalability of multimedia delivery over the Internet [5]. Bandwidth-hungry video applications employing a transmission policy that is solely based on the UDP service model may easily take up a considerable share of the available bandwidth, leaving a little space for other network-friendly multimedia and data flows.

Naturally, delivering all packets to the client on time produces the highest quality of video. However, because of the stringent delay constraints, most real-time video applications have limited error-recovery capabilities against lost packets. Neither missing the delivery deadlines caused by the rate fluctuations as in TCP nor omitting the retransmissions as in UDP suits these applications well. With this in mind, it is of paramount importance to develop protocol suites that take into account the unique features of media content. We classify these features into five categories:

- *Time Sensitivity*: Needless to say, the biggest advantage of multimedia streaming applications is that the clients can download and play the media concurrently. However, this makes streaming applications highly sensitive to delay and delay variation. Depending on the nature of the application, the packets that incur a sender-to-receiver delay of more than a certain threshold are essentially useless. Likewise, in conversational applications such as voice over IP (VoIP) and videotelephony, one-way delays have to be strictly smaller than a few hundred milliseconds. For example, in a VoIP session, one-way delays smaller than 100 ms are not perceived by human listeners. Delays between 100 and 250 ms can be tolerated, however, larger delays usually result in unintelligible voice conversations.

- *Dependency Structure*: Unfortunately, timely reception of a media packet, say a video packet, is not sufficient for the client to play the content of this packet when it is the time to do so. This is because of the highly-efficient predictive-coding techniques that are used in popular video coding standards, *e.g.*, MPEG-x and H.26x. As a result of predictive coding, a dependency is created among the video frames, and this dependency determines the order in which the packets have to be decoded. For example, before decoding a predicted frame, the decoder has to receive and decode all the frames to which this particular predicted frame is referenced (called *ancestor* frames). In other words, if the decoder fails to decode a frame, the decoder will also fail to decode all the frames that are dependent on it (called *descendant* frames).

  Note that the dependency can be introduced in the spatial, temporal or frequency domain. For example, in scalable video coding, a base layer and several enhancement layers are encoded in a hierarchical way. The base layer produces a bare-minimum quality of video. With each additional enhancement layer, the quality is further improved. However, the decoder cannot decode any of the enhancement layers without first decoding the base layer.

- *Unequal Importance*: Intuitively, packets belonging to the ancestor frames or the base layer should be prioritized over the ones belonging to the descendant frames or the enhancement layers. However, the dependency structure is inadequate by itself to solve the problem of prioritization among the *sibling* packets, *e.g.,* the packets belonging to the same frame/layer. A powerful approach to address this problem in an analytical framework is to quantify the contribution of each packet to the achieved quality as its *importance*. However, quantification of the packet importance is challenging in practice as it requires the knowledge of per-packet distortion information, which can only be extracted during the encoding process. Furthermore, the importance of a packet is also a function of time. For example, an ancestor packet becomes more important as a larger number of its descendant packets are delivered to the client, because its successful delivery will now enable the decoding of several packets.

- *Loss Tolerance*: As discussed above, the presentation quality suffers from missing important packets. However, multimedia applications can tolerate losses or delays experienced by less important packets. The effects of such erasures can usually be repaired by the help of advanced error-concealment techniques.
- *Scalability*: While there can only exist a unique representation of the conventional data, one can represent media content at different rates and qualities. The scalability feature allows us to adapt the content according to the bandwidth and processing-power capabilities of the clients.

It is the purpose of this thesis to develop a set of application-specific, media-aware and network-adaptive solutions. We refer to these solutions as Multimedia Transport Protocols (MMTP). In the next section, we present illustrative scenarios to demonstrate what actions MMTP takes against lost and late packets in contrast to the ones that TCP/UDP would take under the same circumstances.

1.2 Illustrative Examples

Let us consider a server-driven streaming system where the client streams real-time video from the server, and the server is the sole decision point for taking the necessary actions to ensure the timely delivery of the video packets. Assume that the raw video is encoded with a standard MPEG-like source coder at 20 frames per second. We assume a simple group-of-pictures (GOP) structure of IPPPPPPPPP, in which decoding a P-frame requires the decoding of previous frames in the same GOP. For the sake of illustration, each frame is assumed to be transmitted in a single video packet. We label each packet with the frame type (written inside the rectangular boxes representing the packets) and its deadline (written above the packets at the client side and below at the server side). The client sends an acknowledgement (ACK) packet to the server for each packet it receives.

1.2.1 Packet Interdependencies

For the GOP structure above, we observe a strict dependency among the packets. Consequently, any lost or late packet impedes the decoding the rest of the frames in the same

GOP. Recall that in case of streaming over TCP, a video packet might be delayed because of the TCP's rate control algorithm, while under the regime of an unreliable UDP-based service model, it never finds a chance to be retransmitted, if it gets lost. Now, consider the lost I-frame in Figure 2, whose absence halts the decoding process until the decoder receives the subsequent I-frame, and hence, causes a severe quality degradation. One naive way to overcome this problem is to suspiciously transmit this I-frame multiple times at the beginning. Although it is clear that more and preferably earlier transmission attempts would increase the chance of on-time delivery of this packet, from the networking point of view, it is not practical to transmit the same packet several times. A better approach would be to do a timely retransmission for the lost I-frame before its decoding deadline passes.

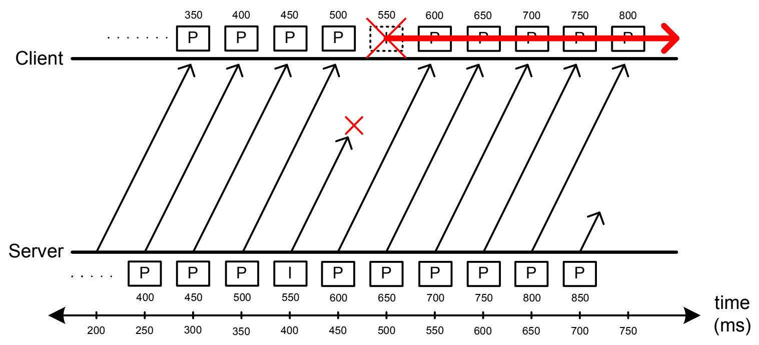


Figure 2: Illustration of error propagation in case of streaming over UDP with no retransmissions.

1.2.2 Early/Late Transmissions

Determining the retransmission time of a packet is difficult yet extremely important. It is never a clear-cut decision whether to transmit a new packet or retransmit an already-transmitted packet since it is impossible to determine whether the missing packet has been lost. Retransmitting a packet at an early stage can be redundant, as it may generate duplicate packets, whereas a late attempt may fail to deliver the packet on time, even if it was successful. On the other hand, a well-timed retransmission increases the chance of timely delivery of the packet, and consequently, decreases the expected distortion experienced at the client side. Naturally, there exists a rate-distortion trade-off that depends on when the video packets are (re)transmitted.

To elaborate more on this point, let us consider an I-frame packet with a delivery deadline of 550 ms. Assume that the server transmits this packet for the first time at $t = 250$ ms and for the second time at $t = 400$ ms as shown in Figure 3-(a). Such a greedy policy for an I-frame might be justified by the importance of this particular packet. However, if the first transmission is successful, the server will have wasted a transmission opportunity, which could have been used for another packet. Now, consider the opposite scenario where the server wants to reduce the chance of wasting bandwidth and waits for a long time before doing the retransmission. As depicted in Figure 3-(b), if the first transmission attempt fails to deliver the packet, the second transmission will likely be too late.

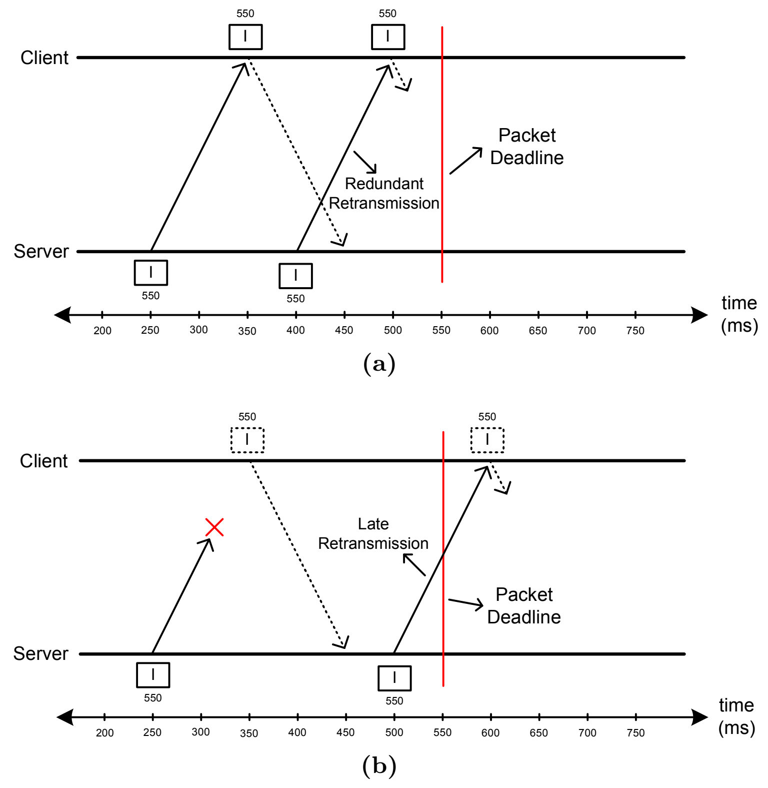


Figure 3: Illustration of redundant and late retransmissions. A retransmission can be redundant if it is too early (a), or ineffective if it is too late (b).

Recall that the loss of an I-frame renders the rest of the frames in the same GOP totally useless. In this case, the client must freeze the last-displayed frame for the duration of the entire GOP. To prevent such an impairment, the server has to ensure the successful delivery of this frame to the client. Therefore, against a possible loss event the server should have an extra transmission opportunity for the I-frame. To this effect, the server can transmit the I-frame at an earlier time. For example, in Figure 4-(a), the I-frame with a deadline of 550 ms is transmitted at $t = 250$ ms before the frames with sooner deadlines. Although the first transmission attempt for the I-frame fails, the server retransmits it at $t = 450$ ms, and the I-frame is successfully delivered to the client before its deadline. The remaining packets are also received by the client before their deadlines despite the packet reordering. As a result, the client does not experience any quality degradation.

1.2.3 Inferring Delay and Loss Characteristics

The recent experimental studies on both Internet data [6] and video [7] traffic show that packet loss events occur in bursts rather than in isolation. Particularly, when packets are transmitted at high bitrates, *i.e.*, when the inter-packet spacing is short, loss events on consecutive packets are highly correlated. Hence, when there is congestion, it is likely that several packets will be lost. In practice, flows experiencing loss reduce their transmission rates to help the network mitigate the congestion. However, this rate reduction prohibits the streaming server not only from transmitting new packets but also from retransmitting the lost ones. Even if the server does not decrease its rate and keeps transmitting, these packets may be dropped or excessively delayed during a long-lasting congestion. To obviate such situations, the server has to take not only *reactive* but also *proactive* actions.

Consider Figure 4-(b) where the successive packets start experiencing longer delays. At $t = 320$ ms, the server receives a late ACK for the P-frame with a deadline of 300 ms, and cautiously identifies the incipient congestion. Consequently, it immediately transmits the I-frame with a deadline of 550 ms, skipping the P-frame with a deadline of 500 ms. This is because the congestion may get worse, and delaying the I-frame until the next transmission opportunity may render its transmission useless. At the end, the I-frame is delivered before its decoding deadline, as are the succeeding frames. With this proactive approach, the server avoids a severe drop in the video quality at the expense of only skipping the least important P-frame in the previous GOP.

It is evident from Figure 4-(b) that if the network congestion lasts for a longer duration,

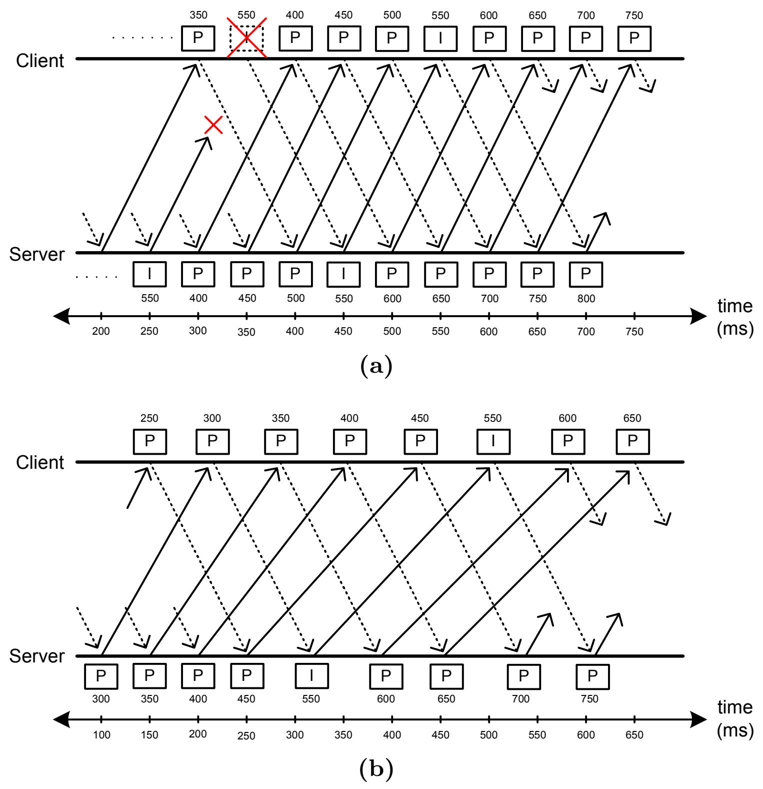


Figure 4: Transmitting more important packets at earlier stages (a) and inferring path conditions (b) allow us to improve the error-control performance.

the server may not be successful in delivering the I-frame on time no matter in which order it schedules the (re)transmissions. In other words, clever design has its limitations when the network performs very poorly. However, even for such circumstances we challenge ourselves in developing novel methods that are capable of coping with the poor network performance to the greatest extent possible. We will study some of these techniques in Chapter 5.

1.3 Relation to the State-of-the-Art

In the last decade, there has been a tremendous interest in multimedia communications from both media-processing and networking research communities. While our goal in this section is to provide an overview of the prior work on different topics in multimedia networking, we can only give a glimpse of it. However, in the subsequent chapters, we will survey the related work about the problems under discussion in detail.

We can classify the prior art on handling errors and losses in multimedia communications into two main categories, namely transport-level and application-level approaches [8].

Error control at the transport level is essential to maintain a certain quality-of-service (QoS) level for multimedia applications. Forward error correction (FEC) and automatic repeat request (ARQ) are the two basic error-control methods. Studies using FEC focus on optimizing the redundancy level according to the channel and media-source parameters [9]. Channel-adaptive FEC is widely used in wireless networks, which are characterized by high bit-error rates [10–12]. Unequal importance of the media packets is effectively handled by the incorporation of unequal error protection (UEP) techniques into FEC [13–20]. Delay-constrained ARQ-based error recovery attempts to meet the real-time requirements of multimedia [21–24]. Hybrid combinations of FEC and ARQ schemes have also been demonstrated to improve the performance of media applications over wireless networks [25, 26].

Robustness to transmission errors can further be improved by application-level techniques. These approaches are effective for those communication scenarios where the error-control mechanisms deployed in the underlying system cannot be optimized for media. Encoder-side optimizations focus on providing spatial and temporal error-resiliency features, which can be accomplished by the use of error isolation, robust binary encoding and error-resilient prediction methods [27–33]. At the client side, the decoder may also perform error concealment to predict the missing parts of the video or audio from the intact parts [29, 34–39]. The use of layered coding with UEP [14, 40, 41] and multiple description coding with diversity techniques [42–46] have been shown to be other effective application-level error-resiliency methods.

Streaming scalable video with optimized packet scheduling drew attention from researchers because of the hierarchical dependency among the layers. First, Podolsky *et al.* [47] used an offline Markov chain analysis to estimate the expected distortion of each candidate transmission policy. An online and efficient solution algorithm to a similar problem was later proposed by Miao and Ortega [48–50]. They presented an algorithm called *expected run-time distortion* to estimate the time-varying importance of media packets according to the feedback. Cuetos and Ross also presented an infinite-horizon Markov decision process (MDP)-based joint scheduling and error-concealment algorithm for MPEG-4 FGS video [51]. Motivated by [47], Chou and Miao proposed a generic rate-distortion optimized packet scheduling framework for ARQ-based loss recovery in [52], followed by [53–57]. The best packet to (re)transmit was decided by an MDP analysis that was based on the prior transmission history, channel statistics and source characteristics. Kalman *et al.* used the same framework with adaptive playout and multiple deadlines [58, 59]. Chakareski and Girod investigated diversity schemes within a similar framework [60, 61]. Finally, Kang and Zakhor proposed a different packet scheduling technique for wireless video that determined the importance of a frame by its relative position within the GOP and a motion-texture discriminator [62].

1.4 Contribution of the Thesis

The theme of this thesis is to develop a collection of end-to-end and system-level MMTP solutions for a variety of multimedia applications, and conduct a comparative performance analysis against the state-of-the-art methods. Considering that today's multimedia services offer an assortment of applications that can run over different types of networks, it is exceedingly difficult to look into every possible scenario. Instead, in the remainder of the thesis, we delve into the most fundamental scenarios. Nonetheless, the lessons we learn from these studies should also shed light on more specific scenarios.

We make the following contributions in this thesis:

- We develop an optimal multi-path transmission framework for streaming error-resilient videos over overlay networks in low-delay video applications,
- We experiment with and analyze the performance of the single/multi-hop and single/multi-path transmission methods for streaming high-resolution videos in mesh networks,
- We formulate a mathematical framework that models on-demand video delivery from multiple content servers, and develop a packet scheduling algorithm that achieves the rate-distortion optimal performance within a multi-server streaming system,
- We investigate the use of proxies to overcome the problems associated with packet loss, large delay and delay jitter in interactive video applications,

- We characterize the packet dynamics in networked-video applications, propose models for packet delay, develop accurate delay and delay-boundary prediction methods, and interpret the correlation between the packet delay and packet loss events,
- We engineer media-aware and network-adaptive error-control methods, and lay out a framework that computes the optimal actions for error control.

In addition, we discuss several issues from an implementation point of view, and summarize our own experiences.

1.5 Organization of the Thesis

The rest of this thesis is organized as a series of chapters, where we study a separate research problem in each chapter. At the beginning of every chapter, we motivate the problem and examine the related work in detail. We provide experimental and/or simulation results along with a thorough discussion to support the proposed solutions. We conclude each chapter with a summary of the main findings.

The outline of the thesis is as follows:

Chapter 2 studies optimal streaming of multiple description video over multiple paths. Models for multi-path streaming and methods for optimal multi-path selection are discussed. Several simulation results are provided to show the effectiveness of the proposed approaches.

Chapter 3 explores the potential benefits of wireless mesh networks for residential video applications that require high bandwidth and low latency. Results from experiments that involve single-hop, multi-hop, single-path and multi-path transmission methods are presented.

Chapter 4 studies streaming on-demand video from multiple servers. A mathematical framework is laid out, and a client-driven rate-distortion optimal packet scheduling algorithm is developed. Simulation results are presented to show the advantages of multi-server streaming over single-server streaming.

Chapter 5 tackles the problem of communicating interactively over large distances and proposes a proxy-based solution to deliver sub-second latency to users faraway from each other. Results from Internet experiments between the U.S. and Europe are presented to demonstrate the effectiveness and potential benefits of the proposed approach.

Chapter 6 investigates the packet delay characteristics in networked-video applications and studies autoregression-based delay and delay-boundary prediction methods. Several simulation results and a detailed comparative analysis of different prediction methods are provided.

Chapter 7 investigates the correlation between the packet delay and packet loss events, and presents a mathematical framework that optimizes the error-control actions based on the urgency and importance of the media packets.

Chapter 8 summarizes the contributions of this thesis and provides a brief discussion on future research directions.

CHAPTER II

OPTIMIZED MULTIPLE DESCRIPTION VIDEO COMMUNICATION OVER OVERLAY NETWORKS

Real-time video communication over the Internet poses several challenging problems due to its stringent delay/loss requirements and complex network dynamics. A promising approach to alleviate the severe impacts of these dynamics is to transmit the video over diverse paths. For such an environment, multiple description (MD) coding has been previously proposed to produce multiple independently-decodable streams that are routed over partially link-disjoint (non-shared) paths for combatting bursty packet losses and error propagation. However, selecting these paths appropriately is fundamental to the success of MD streaming and path diversity. Hence, in this chapter we develop models for MD streaming over multiple paths, and based on these models we propose a multi-path selection method that chooses a set of paths maximizing the overall quality at the client under various constraints. The simulation results with MPEG-2 videos show that sizeable average peak signal-to-noise ratio (PSNR) improvements can be achieved when the source video is streamed over intelligently-selected multiple paths as opposed to over the shortest path or maximally link-disjoint paths. In addition to the PSNR improvement, end-users experience a less-interrupted streaming quality. Our work also considers the architecture and mechanisms by which multi-path streaming can be accomplished in a conventional IP network.

2.1 Introduction[†]

An elusive goal has been to find effective solutions for low-delay video communication applications over the IP networks with time-varying conditions. The timeliness requirements in this definition can only be satisfied through an adaptive time-sensitive approach in the

[†]Parts of this chapter were previously published in [63–66].

best-effort networks. Specifically, real-time video communication demands uninterrupted bandwidth, and rigid bounds on the packet loss and delay to achieve a minimally-acceptable quality. As discussed in Chapter 1, the predominant transport protocols of the Internet are inadequate to provide such guarantees.

To address this problem, a number of error-resilient streaming techniques have been proposed to date. These techniques mainly focus on the source-coding level solutions. One such popular approach is scalable video (SV) coding, where a base layer and several enhancement layers are encoded in a hierarchical way. The base layer provides a low but acceptable level of quality, while each additional enhancement layer refines the quality. Despite its high redundancy rate, SV coding can be employed when there are several clients with different bandwidth and processing-power capabilities streaming from a single source. For instance, while a desktop client with a broadband Internet connection subscribes to all layers, a relatively lower-bandwidth wireless client may subscribe to only the base layer. As its connection speed improves later, it may also receive the enhancement layers as well. This scalability feature makes SV coding a practical solution to support video multicasting in heterogeneous environments [67–69]. However, any erasure in the base layer reception still interrupts the decoding process and renders other layers completely useless, resulting in the underutilization of the network resources.

A specific approach to SV coding is multiple description (MD) coding. Similar to SV coding, MD coding also generates several substreams, which are called *descriptions*. What differentiates MD coding from SV coding is that MD coding does not impose any dependency among its descriptions so that each extra successfully-received description improves the quality further regardless of what has been received so far[1]. This self-reliance of the descriptions provides MD streaming highly-efficient error-resiliency features. However, to fully utilize these features, one should ensure that the description erasures are not concurrent. A conceptually-straightforward way to achieve this is to stream the descriptions over diverse paths. The resulting weak correlation between the packet loss probabilities on diverse

[1]In Appendix B, we provide a detailed comparative analysis of MD and SV streaming.

paths makes MD streaming a good choice for streaming applications that cannot support the well-known error-control/protection methods, such as automatic repeat request (ARQ), forward error correction (FEC) and packet interleaving. For this reason, MD streaming is envisioned as a promising solution for multimedia applications that demand interactivity such as videotelephony, videoconferencing, virtual environments and VoIP over lossy networks [42].

With this motivation, Apostolopoulos studied path diversity along with MD coding to improve the reliability of the streamed video [43]. In this work, he showed how an erasure in a description could be recovered by other descriptions as long as the errors did not occur simultaneously. He further analyzed the effects of bursty loss behavior on the video quality and concluded that minimizing the dependency between the paths resulted in a better quality of video [70]. Several other studies have also been proposed recently [71–78]. As a common goal, these studies attempt to alleviate the severe effects of the congested links or link failures. With the exception of [70, 76] and [77], these studies mainly focus on transmitting the substreams over multiple statistically-independent paths between two end-hosts. However, as we demonstrate in Section 2.5, these paths or the default Internet paths do not necessarily make the best use of the error-resiliency features of MD streaming. In other words, avoiding joint links does not guarantee the best quality of video. "Good" joint links should not be sacrificed - one should try to make the best possible use of them. More importantly, totally link-disjoint paths are rarely available in today's Internet[2]. Hence, using joint links is inevitable in many cases. Consequently, the following question arises: "How should one select the multiple paths for MD streaming?"

In the literature, there are only few studies that have attempted to answer this question. Apostolopoulos *et al.* present path diversity models and path selection methods based on only the packet loss characteristics of the paths in point-to-point [70] and multipoint-to-point networks [76]. Liang *et al.* study video streaming using rate-distortion optimized reference picture selection and path diversity. Given two paths, the proposed path selection

[2]Streaming over link-disjoint paths might be achievable in ad hoc networks (See [71, 72, 79, 80] for details). However, we consider only the Internet in this study.

approach is to send the packet over the path from which the most recent acknowledgement is received [74]. In a work by Nguyen and Zakhor [77], a practical approach is used to select a redundant path in addition to the default Internet path such that the number of joint links between these paths is minimized. If more than one such paths are found, the one with the minimum latency is selected. Despite the fact that these studies propose reasonable path selection methods, they do not consider several important aspects of the media characteristics, network conditions (*e.g.*, bandwidth, packet loss rate, burst length, delay and jitter) and application requirements. All these aspects should be jointly taken into account in the multi-path selection along with the network-connectivity information. To address this point, we first develop a framework that models MD streaming over multiple paths and then use this framework to select the optimal set of paths in a given network such that the streaming quality at the client is maximized, or equivalently, the distortion is minimized[3]. The simulation results with the optimal multi-path selection clearly demonstrate the quality improvements over previously-proposed path selection methods. In addition, optimal multi-path MD streaming is shown to perform better than the best single-path single description streaming in most cases, as is discussed in Section 2.5.2.

The rest of the chapter is organized as follows: In Section 2.2, we provide an overview of MD coding. Section 2.3 introduces the network model envisioned in our work. Multi-path selection is discussed in detail in Section 2.4. Simulation results are presented in Section 2.5. Finally, Section 2.6 concludes the chapter.

2.2 Multiple Description Coding

Multiple description (MD) coding is a source-coding technique that generates several descriptions such that different levels of reconstruction qualities can be obtained from different subsets of these descriptions. In contrast to scalable coding, there is no hierarchy among the descriptions so that each description may be independently decoded. This property makes MD coding highly suitable for packet networks where no prioritization exists among

[3]It should be mentioned that the proposed multi-path selection method is *optimal* within the context of the network models and objective functions that are considered in this study.

the packets. MD coding provides this robustness at the expense of some reduction in the compression efficiency.

A typical MD coding system is shown in Figure 5. A sequence of source symbols $\{X_i\}_{i=1}^{N}$ is input to the MD encoder. This encoder produces two descriptions, which are then transmitted over possibly different channels. If both descriptions are received intact at the client, the central decoder is used to produce the highest quality of signal. However, if only one description is received free of errors, the corresponding side decoder is used to produce a signal of acceptable quality. The reconstructed signal and distortion at the k^{th} decoder are represented by $\{\hat{X}_i^{(k)}\}_{i=1}^{N}$ and D_k, respectively. r_k denotes the rate in terms of the number of bits per source sample on the k^{th} channel.

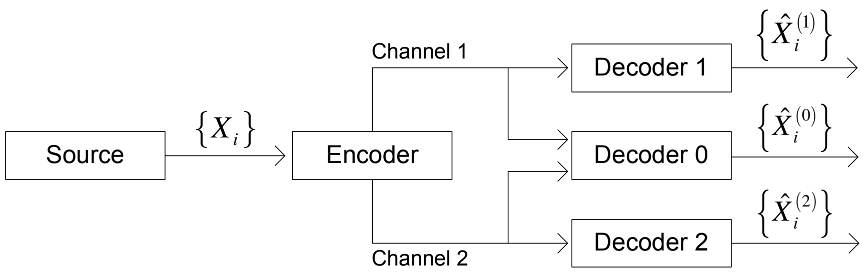


Figure 5: A typical MD codec with two descriptions.

Our objective is to estimate the end-to-end video quality in terms of the path parameters. We start our derivation by relating the rates of the descriptions to the reconstructed signal quality, or equivalently, to the distortion. Let us consider two descriptions generated at the source. We define the average distortion at the k^{th} decoder as

$$D_k = E\left\{\frac{1}{N}\sum_{i=1}^{N} d\left(X_i, \hat{X}_i^{(k)}\right)\right\} \qquad k = 0, 1 \text{ and } 2, \tag{1}$$

where $d(.)$ can be any distortion measure. Because of its popularity, we take $d(.)$ as the squared euclidian norm. To develop analytic expressions for the average distortion, we need to choose a media type and source model. For that purpose, we assume that the source is a compressed-video stream. However, the methodology presented here can be applied to other media sources, *e.g.*, voice and 3-D graphics, with their representative distortion models.

For compressed-video applications, several models (*e.g.*, [81–84]) have been developed to estimate the distortion for a given source rate. Naturally, incorporating these models into our framework would allow us to make more accurate distortion estimates. However, we seek an easy-to-implement, yet representative model since the ones introduced in [81–84] are computationally expensive, thereby reducing the practicality of the proposed multi-path selection method.

Let us consider the illustrative rate-distortion curves in Figure 6. These figures are produced by encoding various test sequences by a standard MPEG-2 encoder (TM5 [85]) with default settings. For distortion estimation, we use the following equation, where κ is a model parameter:

$$D(r_k) \approx \frac{\kappa}{r_k}. \tag{2}$$

For offline-encoded video sequences, the value of κ can be pre-computed. However, for real-time encoding, an initial value is assumed for κ, and this value is refined as more frames are encoded.

We note that the expression in (2) is the first-degree approximation of the distortion model proposed by Chiang and Zhang [86]. Based on the empirical results of 10 video test sequences, we observe that (2) fits the actual rate-distortion curves quite well, and its accuracy is sufficient for our purposes. In Section 2.4, we derive a relation between the source rate (r_k) and end-to-end bandwidth on a path. Subsequently, by using (2) we will estimate the distortion when the video is streamed over a given path.

There are various techniques for generating multiple descriptions. One of the more straightforward methods is time-domain partitioning [43], which separates the even and odd-numbered frames of a video sequence into two groups, and encodes them individually to produce two descriptions. Likewise, the sequence may also be partitioned in the spatial and frequency domain [87]. Other popular approaches are multiple description scalar quantization (MDSQ) [88, 89] and multiple description transform coding (MDTC) [90]. In MDSQ, multiple quantizers are used to generate descriptions, whereas correlating transforms are used in MDTC. For brevity, we do not go into the further details of MD coding. For an excellent survey, interested readers are referred to [42].

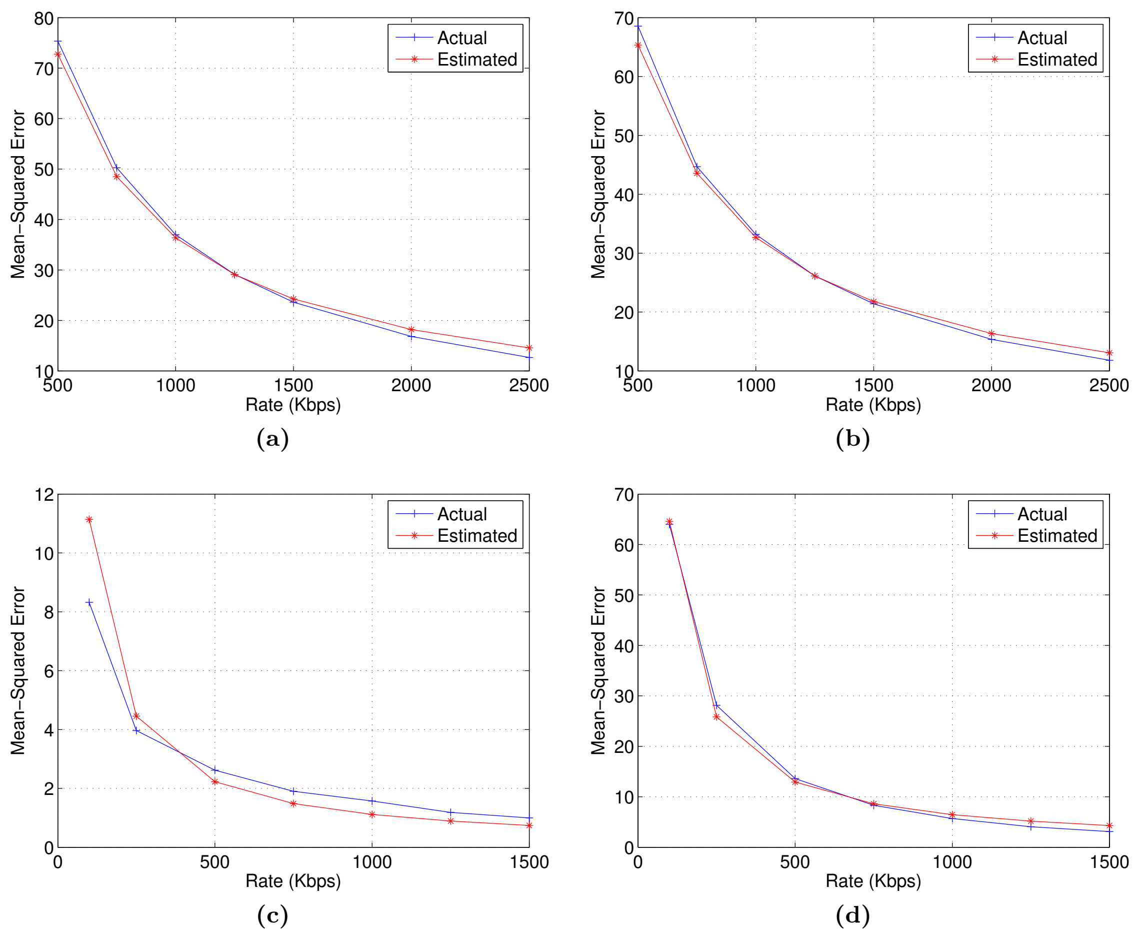


Figure 6: Actual and estimated rate-distortion curves for the Table Tennis (a), Football (b), Miss America (c) and Foreman (d) sequences.

2.3 Envisioned Network Model

Achieving MD streaming over a conventional network involves two main issues: First, we require a mechanism to continually monitor the network conditions. The collected information is then evaluated to compute the application-specific routes. Second, we need an infrastructure to transmit the descriptions over these routes, which would necessitate an advanced routing mechanism in today's Internet. Fundamentally, constraint-based routing can only be achieved by going beyond the conventional destination-based routing algorithms and providing mechanisms to explicitly manage the traffic inside the network. Because of the current structure of the Internet, it is not easy to deploy such routing mechanisms at a large scale. However, by abstracting the underlying network, one can emulate the desired routing algorithm at higher layers. A popular way to achieve this is the *overlay model*. Overlays are emerging rapidly and envisioned to be an effective solution for futuristic streaming applications [91]. With the overlay model, a virtual network is built over the real physical topology, and the nodes in these virtual networks are connected with logical links. In this section, we give an overview of our envisioned network model and briefly discuss how we implement MD streaming within this model.

2.3.1 Overlay Model

A large number of emerging Internet applications are not able to utilize the network resources at their full potential because of the fact that the underlying network does not support many of the functionalities required by these applications. While the knowledge of the physical topology and network status is essential for such applications to perform well, this kind of information is generally not available to the end-hosts. To overcome these difficulties, we consider an overlay infrastructure in this study. While there exist several overlay studies such as Resilient Overlay Networks (RONs) [92] and X-Bone [93], a good example for our case is Narada [94], which was proposed to support end-system multicast.

Narada is a protocol that constructs an overlay structure among participating nodes in a self-organizing and fully-distributed manner. It has been shown that Narada is robust to changes in group membership and node failures. Participating overlay nodes, which we call O-nodes, start building the network without the knowledge of the physical topology. Using constant probe messages, Narada continually refines the overlay structure in a controlled fashion.

In building the overlay, Narada is mainly interested in the latency, whereas our goal is to fully characterize the physical topology, and use this information to identify the joint and disjoint links for a given set of paths and measure the link characteristics. This can be achieved by assigning additional functionalities to the Narada protocol. Armed with these capabilities, the O-nodes can identify the joint and disjoint links, measure the bandwidth, packet loss rate, delay and jitter statistics, and exchange this information among themselves with a routing protocol to construct and maintain the overlay network. Since the O-nodes periodically probe each other, any change in this information is reflected to the end-hosts so that they can take the necessary actions in the multi-path selection process. The details of implementing such protocols can be found in [92, 95].

Given an overlay network, we first find the optimal set of paths from the server to the client[4]. Then, we encode the path information, *i.e.*, the addresses of the O-nodes to be traversed, into the packet payload. This information is used by the O-nodes to forward the incoming packets to the next O-node on the path. Note that we leverage the already-deployed algorithms for routing the data and probe packets between the O-nodes. Hence, the overall procedure is transparent to the physical network.

An advantage of using an overlay infrastructure is the savings in the path-computation time. The number of O-nodes is typically a small percentage of the number of nodes in the entire network. Hence, the processing power and computation time required for path selection decreases dramatically. However, deploying a larger number of O-nodes might produce alternative paths. This, in turn, can result in a better streaming quality at the expense of increased computational complexity.

In this section, we briefly outlined the envisioned network model. The further design and performance issues are beyond the scope of this study. Given that our primary focus is selecting multiple paths, henceforth, we assume the availability of an infrastructure that runs the required protocols.

2.3.2 Definitions

An overlay network is characterized by a set of nodes $\mathcal{N} = \{\mathcal{N}_1, \mathcal{N}_2, ..., \mathcal{N}_n\}$ and a set of links, $\mathcal{L}$. We denote the directed link between nodes $\mathcal{N}_u$ and $\mathcal{N}_v$ by $l_{u,v}$. A *path* $\mathcal{P}_{S \to C}$ is defined by a set of nodes and links connecting nodes $\mathcal{N}_S$ and $\mathcal{N}_C$. The set of the nodes on this path is denoted by $\mathcal{N}_{S \to C}$, and the set of the links on this path is denoted by $\mathcal{L}_{S \to C}$. Let $\mathcal{P}^1_{S \to C}$ and $\mathcal{P}^2_{S \to C}$ be any two paths. These paths are said to be totally link-disjoint if and only if $\mathcal{L}^1_{S \to C}$ and $\mathcal{L}^2_{S \to C}$ do not have any common elements. If the sets $\mathcal{L}^1_{S \to C}$ and

[4]In a video communication application, both end-users send and receive video packets. However, for the sake of clarity, in the rest of the chapter, we consider video transmission in one direction, *i.e.*, from the server to the client.

$\mathcal{L}^2_{S\to C}$ intersect, the links belonging to both paths are referred to as *joint links*, whereas the remaining links are referred to as the *disjoint links*.

2.3.3 Link Parameters

The parameters for link $l_{u,v}$ are defined as follows:

- $b_{u,v}$ denotes the bandwidth between nodes $\mathcal{N}_u$ and $\mathcal{N}_v$.
- $p_{u,v}$ denotes the observed packet loss probability between nodes $\mathcal{N}_u$ and $\mathcal{N}_v$. $p_{u,v}$ values for different links are assumed to be independent since the corresponding observations are autonomous.
- $t_{u,v}$ denotes the minimum delay between nodes $\mathcal{N}_u$ and $\mathcal{N}_v$. This includes processing, transmission and propagation delays, but not varying queuing delays.
- $j_{u,v}$ denotes the jitter between nodes $\mathcal{N}_u$ and $\mathcal{N}_v$. It is defined as the difference between the longest delay experienced within a pre-defined time period and $t_{u,v}$. The collection of several $j_{u,v}$ values defines the variation in the forward-trip times.

Note that since the links are directed, even though links $l_{u,v}$ and $l_{v,u}$ have the same bandwidth and delay, their packet loss rate and jitter values may differ depending on the amount of traffic flowing in each direction.

2.4 Multi-Path Selection

In this section, we introduce our multi-path selection method. Let us consider two paths, $\mathcal{P}^1_{S\to C}$ and $\mathcal{P}^2_{S\to C}$, between nodes $\mathcal{N}_S$ and $\mathcal{N}_C$, which are referred to as a *path pair*. We do not require the paths to be different, *i.e.*, $\mathcal{P}^1_{S\to C}$ may be the same as $\mathcal{P}^2_{S\to C}$. Our goal is to estimate the expected end-user quality in terms of the link parameters making up these paths. In other words, we want to derive a function $\mathbf{F}$ that estimates the expected distortion at the client, $E\{D\}$, based on the link parameters. That is,

$$E\{D\} = \mathbf{F}\left(b_{u,v}, p_{u,v}, t_{u,v}, j_{u,v}\right) \text{ for } \forall l_{u,v} \in \mathcal{L}^k_{S\to C},\ k = 1, 2. \tag{3}$$

In this study, we consider only the two-description case due to its efficacy. Although (3) can be generalized to a larger number of descriptions in a straightforward manner, finding more than two good diverse paths in the Internet is not practical. Moreover, generating more than two descriptions increases the overhead of MD coding, and the gained diversity is usually insufficient to justify this overhead.

Recall that our ultimate goal is to find the path pair that minimizes the cost function, **F**. For two descriptions, we have four possible cases determined by the description on-time arrivals. Let $D_{1,1}$ ($D_{0,0}$) denote the achieved distortion when both descriptions arrive intact on time (are lost or delayed). Similarly, let $D_{1,0}$ ($D_{0,1}$) denote the achieved distortion when the first (second) description arrives intact on time, but the other one is lost or delayed. In this case, the missing description is concealed from the received description. Given these distortions, we can write the expected distortion at the client as

$$\begin{aligned} E\{D\} \quad = \quad & \overbrace{P\{\text{Both received on time}\}}^{P_{1,1}} \times D_{1,1} \\ & + \overbrace{P\{1^{\text{st}}\text{ received on time }\&\ 2^{\text{nd}}\text{ lost or delayed}\}}^{P_{1,0}} \times D_{1,0} \\ & + \underbrace{P\{1^{\text{st}}\text{ lost or delayed }\&\ 2^{\text{nd}}\text{ received on time}\}}_{P_{0,1}} \times D_{0,1} \\ & + \underbrace{P\{\text{Both lost or delayed}\}}_{P_{0,0}} \times D_{0,0}. \end{aligned} \tag{4}$$

The success probabilities, $P_{1,1}$, $P_{1,0}$, $P_{0,1}$ and $P_{0,0}$, are derived later in this section.

The shared parameters due to the joint links play a critical role in the multi-path selection process. Therefore, the selection process must consider all possible path pairs. One cannot evaluate the paths individually, rank them, and then choose the best two; this does not necessarily select the optimal pair because of the dependencies between the paths. For the sake of clarity, throughout the section we use k to denote the k^{th} path and its parameters, where $k \in \{1, 2\}$. We start our derivations with distortion.

2.4.1 Bandwidth - Distortion Relation

In this study, we adopt the time-domain partitioning method with two descriptions to illustrate the relation between the distortion terms ($D_{1,1} - D_{0,0}$) and link bandwidths. Although the presented distortion models are developed for this particular MD coding scheme, (4) can be applied to any MD coding scheme by incorporating its respective distortion model, *i.e.*, by replacing the distortion terms $(D_{1,1} - D_{0,0})$ with the respective ones.

Let B_k denote the end-to-end bandwidth on path $\mathcal{P}_k$. Naturally, B_k is the minimum bandwidth over all the links on $\mathcal{P}_k$. That is,

$$B_k = \min \left\{ b_{u,v} : \forall l_{u,v} \in \mathcal{L}_k \right\}. \tag{5}$$

In Section 2.2, we provided an equation to estimate the distortion at a given source rate for a compressed-video source. Now, we give the relation between r_k in (2) and B_k. Assuming that the k^{th} description is encoded with a rate of r_k bits per pixel (bpp) at a frame rate of F (frames/second), and the source video has a resolution of $W \times H$ (pixels/frame), we have

$$r_k = \frac{B_k}{W \times H \times F \times c} \quad \text{(bpp)}, \tag{6}$$

where c is a known constant that depends on the chroma sub-sampling format. Given this relation, we first compute $D_{1,1}$, *i.e.*, the distortion rendered at the client when both descriptions are received intact on time, as the average of the individual description distortions. That is,

$$D_{1,1} = \frac{1}{2} \times \frac{\kappa'}{B_1} + \frac{1}{2} \times \frac{\kappa'}{B_2}, \tag{7}$$

where κ' is a constant that represents the constant terms in (2) and (6). Note that in (7) we assume that the bandwidth of the bottleneck joint link, say B^*, can support the description rates of B_1 and B_2 simultaneously. If this is not the case, the bandwidth of the bottleneck joint link should be shared by the descriptions in such a way that $B^* = B_1 + B_2$.

Second, to derive the expression for $D_{1,0}$ we need to consider the error concealment at the client. Recall that the time-domain partitioning method separates the even and odd-numbered frames, and encodes them individually to produce two descriptions. When only the first description (even-numbered frames) is received successfully, the missing odd-numbered frames can be reconstructed at the client by using the even-numbered frames. A common technique used for this purpose is motion-compensated temporal interpolation [27]. As expected, the quality of the reconstructed odd-numbered frames increases with the rate at which the even-numbered frames are encoded. However, because of the imperfections in the reconstruction, their qualities will be lower compared to the even-numbered frames.

To incorporate this increase in the distortion of the reconstructed frames, we use a scaling factor γ and express $D_{1,0}$ as follows:

$$D_{1,0} = \frac{1}{2} \times \frac{\kappa'}{B_1} + \frac{1}{2} \times \left(\gamma \times \frac{\kappa'}{B_1} \right). \tag{8}$$

Our experiments indicate that this expression provides a good approximation to the realized distortion. For the test sequences we use in the simulations in Section 2.5, we quantify the value of γ for different rates. We observe that although its value varies depending on the test sequence, γ is fairly robust against the rate variations. In our case, γ numerically equals 2.3, 3.4 and 1.5 for the Table Tennis, Football and Paris sequences, respectively. For non-interactive applications, the value of γ can be pre-computed. For interactive applications, a default γ value, reflecting the application characteristics (*e.g.*, a value of 1.5 for videotelephony applications) can be used. Note that $D_{0,1}$ is also expressed in a similar way by replacing B_1 with B_2 in (8).

Recall that in this study we employ the time-domain partitioning method to generate two descriptions. Naturally, this method is suitable for the balanced mode of MD coding, where the rates of the descriptions are likely to be equal or close to each other in order to avoid flickering [96]. However, in some cases the candidate paths may have unequal bandwidths, and if the γ is sufficiently low, reconstructing the low-quality description from the high-quality description may result in lower distortion in terms of the mean-squared error than decoding the original low-quality description. In such cases, one can argue that transmitting the low-quality description is useless. However, transmitting both descriptions, regardless of their rates, is still essential in our system because of two reasons: First, decoding the video from only the high-quality description causes drift error and disturbs the perceptual video quality. Our quality-assessment tests show that the video produced from both descriptions is usually preferred over the video decoded from only the high-quality description since the former one gives a more natural, continuous, and hence, more pleasant video. Second, the low-quality description enables us to decode and display the video when the high-quality description is lost or late for decoding. If it is not transmitted at all, the error-resiliency features of MD streaming cannot be exploited, and the system becomes a pure single description streaming system.

The final distortion term in (4) is $D_{0,0}$, which is the distortion incurred when none of the descriptions are decoded at the client. To compute $D_{0,0}$, we assume an average value for each pixel and compute the corresponding expected error. The resulting large value motivates our method to reduce the number of simultaneous description losses.

2.4.2 Computation of Success Probabilities

The next step in the computation of (4) is to find the success probabilities ($P_{1,1} - P_{0,0}$). Each of these probabilities is composed of two components: The first component computes the probability that the description arrives at the client in a finite duration, *i.e.*, the description is not lost during the transmission. We refer to this component as the *arrival probability.* On the other hand, for the second component we are interested in computing the probability that the description arrival occurs before a pre-specified deadline. This component is referred to as the *on-time delivery probability.* While in a typical streaming application, the deadline constraint may be relaxed by pre-buffering a large amount of data, this ability is often limited in interactive applications. Consequently, large variations in the packet arrival times pose impediments to the timely delivery, rendering the on-time delivery probability as important as the arrival probability. A detailed discussion about the impact of end-to-end path delay on the streaming quality is presented in [7, 97].

2.4.2.1 Arrival Probability

We adopt the well-known two-state Markovian Gilbert-Elliott (GE) model [98, 99] to describe the temporal behavior of packet losses on a link [6, 7]. The pertaining state-transition diagram is given in Figure 7. In this figure, NC (non-congested) and C (congested) represent the packet arrival and loss states, whose steady-state probabilities are given by $\pi_{\text{NC}} = \frac{\beta}{\alpha+\beta}$ and $\pi_{\text{C}} = \frac{\alpha}{\alpha+\beta}$, respectively. For the link $l_{u,v}$, we compute the transition probabilities from

$$\alpha_{u,v} = \frac{p_{u,v} \times \beta_{u,v}}{1 - p_{u,v}} \quad \text{and} \quad \beta_{u,v} = \frac{1}{L_{u,v}}, \tag{9}$$

where $L_{u,v}$ is the average burst length depending on the inter-packet spacing [7]. We note that if the dependency between the loss events on a link was ignored, the optimal solution would be to send all descriptions over the best single path (assuming that the bandwidth on this path could support both descriptions simultaneously). The GE model provides a simple expression for the characterization of individual links. However, the derivation of the end-to-end arrival probabilities on partially link-disjoint paths is not trivial. Next, we focus on this derivation.

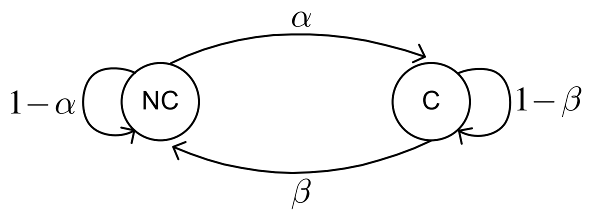


Figure 7: State-transition diagram for the Gilbert-Elliott model.

Preliminaries:

Consider the generic topology depicted in Figure 8, where two candidate paths split at node $\mathcal{N}_Z$, and later merge again at node $\mathcal{N}_Q$. The joint links between the server and segregation point ($\mathcal{N}_Z$) form a sub-network denoted by Network$_{joint}$. Beyond node $\mathcal{N}_Z$, the disjoint links traversed by each path until node $\mathcal{N}_Q$ are also grouped to form the corresponding sub-networks denoted by Network$^k_{disjoint}$. Suppose that the server generates two correlated descriptions at every Δt units of time, and the descriptions produced at the time instant t_i are denoted by $X_1^{t_i}$ and $X_2^{t_i}$.

In (4), we consider the loss dependency between the correlated descriptions $X_1^{t_i}$ and $X_2^{t_i}$, but not the dependency between the descriptions transmitted at successive time instants (*e.g.*, $X_1^{t_i}$ and $X_1^{t_i+\Delta t}$). A direct implication of this is that we do not take into account the temporal error propagation in the cost function. However, this assumption is well justified in our scenario since modeling the temporal error propagation does not affect the path-selection results. Recall that our goal is to minimize the average distortion rather than the absolute distortion. Consideration of the absolute distortion and temporal error propagation would only be necessary if the path-selection decision was given on a per-packet basis (See Chapter 4 for details). However, in our case, until a new selection is made upon receiving new feedback from the O-nodes, the selected paths are used to transmit all packets in each description regardless of the packet content.

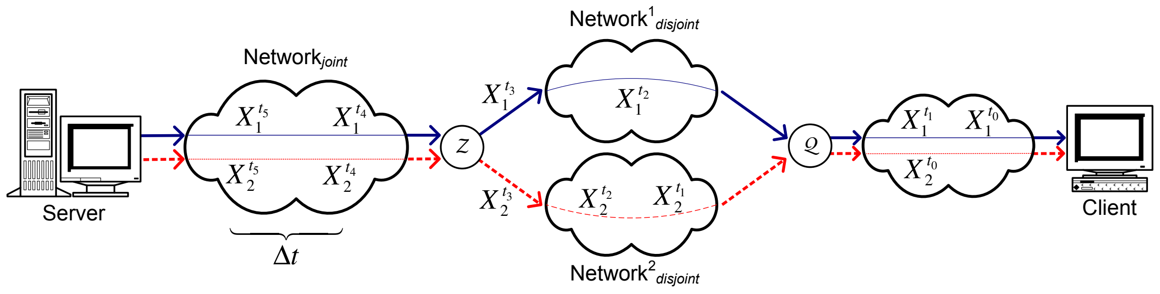


Figure 8: A sketch describing the path segregation method.

After descriptions $X_1^{t_i}$ and $X_2^{t_i}$ are generated, they are transmitted back-to-back in packets over Network$_{joint}$. If no loss occurs, these packets arrive at node $\mathcal{N}_Z$ back-to-back approximately at the same time. In this period, it is likely that these packets are in the same burst period. Hence, we employ a GE model for Network$_{joint}$. However, once the paths split at node $\mathcal{N}_Z$, these two packets are routed over different paths. In [70], Apostolopoulos *et al.* have shown that the bursty packet losses experienced by a description have the same impact as those of the random losses, provided that the descriptions are not concurrently lost. Considering that the probability of simultaneous loss on Network$^k_{disjoint}$ is small, we model the loss behavior on disjoint links with a Bernoulli distribution, where the loss events are independent of each other.

Path Segregation:

When traversing Network$^k_{disjoint}$, descriptions $X_1^{t_i}$ and $X_2^{t_i}$ experience uneven delay and jitter. Thus, they are likely to arrive at node $\mathcal{N}_Q$ at different times. The critical question is whether or not these descriptions will meet at node $\mathcal{N}_Q$ within the same burst period. This is important to our modeling since their loss probabilities will remain independent of each other if there exists a sufficient time gap between them, even though they are traversing the same links beyond node $\mathcal{N}_Q$. In fact, providing a time gap between the descriptions in such a way that they are not within the same burst period is the essence of packet interleaving techniques for MD streaming [100].

To quantify this point, we conduct simulations on several topologies with various link delay and jitter attributes. The simulation setup is given in detail in Section 2.5. Our goal is to determine the percentage of the correlated descriptions that fall into the same burst period at node $\mathcal{N}_Q$. The results show that (*i*) only 7% of the descriptions fall into the same burst period at node $\mathcal{N}_Q$, (*ii*) 3% fall into the same burst period between node $\mathcal{N}_Q$ and the client, although they were not in the same burst period at node $\mathcal{N}_Q$, and (*iii*) 90% do not fall in the same burst anywhere between node $\mathcal{N}_Z$ and the client. The reason behind this result is that the proposed multi-path selection method naturally chooses good links with low packet loss rates. These links tend to have short burst periods (See [6, 7] for details). Hence, the delay difference experienced over $\text{Network}^k_{disjoint}$ is sufficient for the correlated descriptions to arrive in different burst periods. Based on this observation, we deduce that once the paths split, the loss events on the rest of the paths should be considered independent despite a possible merging at a later node. Hence, we replicate any joint link beyond the segregation point in each $\text{Network}^k_{disjoint}$ and employ individual Bernoulli models for $\text{Network}^k_{disjoint}$. In fact, modeling these loss events by a GE model (rather than Bernoulli models) would have underestimated the arrival probabilities. We refer to this method as *path segregation.*

In the simulations, an ill-posed scenario, when path segregation overestimates the arrival probability, is the case when there is a long-lasting congestion on the access link of the client, *i.e.,* on the last link of the path. If the client is not multi-homed, *i.e.*, it is connected to the Internet over a single interface, path diversity is unfortunately not adequate by itself to successfully deliver the packets on time during this congestion. However, we anticipate that with the increasing popularity of emerging broadband technologies, the performance of access links will improve considerably.

Link Aggregation:

Generally, the joint sub-path in Network_{joint} is a combination of several Markovian links in series. Although it is straightforward to compute the arrival probabilities by considering each of these links individually, this computation would be largely simplified if the combination of these links could be approximated by a single Markovian link. To do so, one has to find the joint GE parameters for the approximated link, namely α_{joint} and β_{joint}. Recall that the packet loss rates of consecutive links are independent. Hence, we can directly compute the packet loss rate for Network$_{joint}$ from

$$p_{joint} = 1 - \prod_{\forall l_{u,v} \in \text{Network}_{joint}} \Big(1 - p_{u,v}\Big). \tag{10}$$

However, estimating the joint average burst length, L_{joint}, is not as trivial. This is because the bursty periods in a stream accumulate as the stream goes over more links, *i.e.*, the effective average burst length increases. Although there are several approaches to handle this problem [98, 99], they usually require a higher-order state analysis. To estimate L_{joint} empirically, we have conducted simulations on various topologies with different loss characteristics. From studying these simulations, we came up with the following approximation:

$$L_{joint} \approx \frac{1}{p_{joint}} \times \sum_{\forall l_{u,v} \in \text{Network}_{joint}} \Big(p_{u,v} \times L_{u,v}\Big). \tag{11}$$

Once we have p_{joint} and L_{joint}, we compute α_{joint} and β_{joint} by plugging p_{joint} and L_{joint} into (9). This method is called *link aggregation* since it combines several Markovian links into a single one. The simulation results show that (11) successfully estimates L_{joint}. For a wide range of link parameters, the difference between the measured and estimated burst lengths was observed to be between 0.1% and 4.1%. On the other hand, since the disjoint sub-paths in Network$^{k}_{disjoint}$ are modeled as Bernoulli, we can directly convert these links into a single Bernoulli link whose loss probability is given by

$$p^{k}_{disjoint} = 1 - \prod_{\forall l_{u,v} \in \text{Network}^{k}_{disjoint}} \Big(1 - p_{u,v}\Big). \tag{12}$$

Finally, the arrival probability for a path is the product of the arrival probabilities on its joint and disjoint sub-paths.

2.4.2.2 On-Time Delivery Probability

The second step in computing the success probabilities is to derive the on-time delivery probabilities. Let T_k denote the minimum end-to-end delay for path $\mathcal{P}_k$. We can write T_k as

$$T_k = \sum_{\forall l_{u,v} \in \mathcal{L}_k} t_{u,v}. \tag{13}$$

We require T_k to be smaller than a value that is required by the application. Otherwise, it means that all packets transmitted over this path will be late by the time they are received. For many two-way interactive applications, a forward-trip time less than 200 ms can be considered as the tolerance limit. A recent empirical study [97] reports that modem users can experience a jitter of 200 ms, or more, with a probability of 50%. That is, in half of the communication session the interactivity is hindered. For the high speed connections such as DSL and LAN, this probability reduces to 20%, but has still a high value. This variation in the delay adversely affects the on-time delivery probability.

Let J_k denote the end-to-end jitter on path $\mathcal{P}_k$. We divide J_k into two components, j_{joint} and $j^k_{disjoint}$, for the jitters experienced in Network$_{joint}$ and Network$^k_{disjoint}$, respectively. Assuming that jitters on the consecutive links are independent, we compute J_k as follows:

$$J_k = \underbrace{\sum_{\forall l_{u,v} \in \text{ Network}_{joint}} j_{u,v}}_{j_{joint}} + \underbrace{\sum_{\forall l_{u,v} \in \text{Network}^k_{disjoint}} j_{u,v}}_{j^k_{disjoint}} . \tag{14}$$

To visualize the importance of J_k, consider Figure 9, where a typical probability density function for the forward-trip times (FTT) of the successfully-received packets is plotted as a function of time (τ) [101]. By definition, T_k is the minimum value that FTT can take on path $\mathcal{P}_k$. In our model, we do not assume any particular density function for FTT. However, for the simulations in Section 2.5, we model this density by a shifted Gamma distribution with parameters n_k and λ_k [101], which is given by

$$p_{FTT}(\tau) = \begin{cases} 0, & \tau \leq T_k; \\ \frac{\lambda_k{}^{-n_k}}{\Gamma(n_k)}(\tau - T_k)^{n_k-1} e^{-(\tau - T_k)/\lambda_k}, & \tau > T_k. \end{cases} \tag{15}$$

In (15), n_k and λ_k denote the number of the links and the average link jitter on path $\mathcal{P}_k$, respectively. Thus, we have $\lambda_k = J_k/n_k$. (15) can also be viewed as the distribution of a random variable that is equal to a constant T_k plus the sum of n_k independent and identically-distributed exponential random variables each with mean λ_k [101]. Note that, in our framework n_k, T_k and J_k are known parameters, hence, (15) is completely defined.

Let T_{max} in Figure 9 represent the maximum tolerable delay for the target application. It follows that the on-time delivery probability is equal to the area of the shaded region.

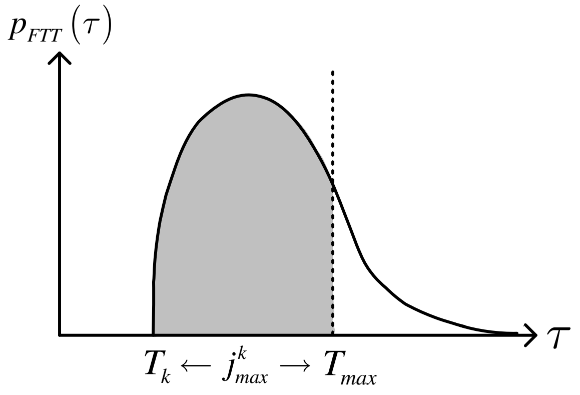


Figure 9: A typical density function for the forward-trip times of the successfully-received packets.

Often, different paths have different end-to-end delays, and hence, different tolerances for jitter. We define j_{max}^k as the maximum jitter that can be tolerated on path $\mathcal{P}_k$. By definition, $j_{max}^k = T_{max} - T_k$. Then, it can be seen that the on-time delivery probability can also be given as the probability of J_k being smaller than j_{max}^k.

Finally, we have all the information required to write the success probabilities in terms of the path parameters. Recall that in (4), $P_{1,1}$ computes the probability that both descriptions arrive at the client intact and on time. To find its value, we first compute the arrival probability of transmitting both descriptions intact over Network$_{joint}$ and individual arrival probabilities of transmitting each description intact over Network$^k_{disjoint}$. Since these events are independent, we multiply them to find the arrival probability of both descriptions. Second, we compute the probability that the descriptions will arrive at the client before the pre-specified deadline. Finally, we multiply this value with the arrival probability to get $P_{1,1}$.

$P_{1,0}$ considers the cases where the first description is received intact on time and the second description is (i) lost in Network$_{joint}$ or in Network$^2_{disjoint}$, or (ii) delayed beyond the decoding deadline despite being transmitted intact. Hence, we separately compute the corresponding probabilities of these three events and add them up to get $P_{1,0}$. $P_{0,1}$ can also be computed in the same manner. Finally, $P_{0,0}$ is the complement of the sum of $P_{1,1}$, $P_{1,0}$ and $P_{0,1}$. The corresponding equations are given in (16) - (19), where we denote the joint GE parameters by α and β instead of α_{joint} and β_{joint}, respectively. The list of our notation is given in Table 1.

$$\begin{aligned} P_{1,1} &= \Big((\pi_{\text{NC}}(1-\alpha)^2 + \pi_{\text{C}}\beta(1-\alpha)) \times (1 - p^1_{disjoint}) \times (1 - p^2_{disjoint})\Big) \\ &\times Prob(j_{joint} + j^1_{disjoint} \le j^1_{max} \text{ and } j_{joint} + j^2_{disjoint} \le j^2_{max}) \end{aligned} \tag{16}$$

$$\begin{aligned} P_{1,0} &= \Big((\pi_{\text{NC}}(1-\alpha)\alpha + \pi_{\text{C}}\beta\alpha) \times (1 - p^1_{disjoint})\Big) \\ &\times Prob(j_{joint} + j^1_{disjoint} \le j^1_{max}) \\ &+ \Big((\pi_{\text{NC}}(1-\alpha)^2 + \pi_{\text{C}}\beta(1-\alpha)) \times (1 - p^1_{disjoint}) \times p^2_{disjoint}\Big) \\ &\times Prob(j_{joint} + j^1_{disjoint} \le j^1_{max}) \\ &+ \Big((\pi_{\text{NC}}(1-\alpha)^2 + \pi_{\text{C}}\beta(1-\alpha)) \times (1 - p^1_{disjoint}) \times (1 - p^2_{disjoint})\Big) \\ &\times Prob(j_{joint} + j^1_{disjoint} \le j^1_{max} \text{ and } j_{joint} + j^2_{disjoint} > j^2_{max}) \end{aligned} \tag{17}$$

$$\begin{aligned} P_{0,1} &= \Big((\pi_{\text{NC}}\alpha\beta + \pi_{\text{C}}(1-\beta)\beta) \times (1 - p^2_{disjoint})\Big) \\ &\times Prob(j_{joint} + j^2_{disjoint} \le j^2_{max}) \\ &+ \Big((\pi_{\text{NC}}(1-\alpha)^2 + \pi_{\text{C}}\beta(1-\alpha)) \times p^1_{disjoint} \times (1 - p^2_{disjoint})\Big) \\ &\times Prob(j_{joint} + j^2_{disjoint} \le j^2_{max}) \\ &+ \Big((\pi_{\text{NC}}(1-\alpha)^2 + \pi_{\text{C}}\beta(1-\alpha)) \times (1 - p^1_{disjoint}) \times (1 - p^2_{disjoint})\Big) \\ &\times Prob(j_{joint} + j^1_{disjoint} > j^1_{max} \text{ and } j_{joint} + j^2_{disjoint} \le j^2_{max}) \end{aligned} \tag{18}$$

$$P_{0,0} = 1 - P_{1,1} - P_{1,0} - P_{0,1} \tag{19}$$

2.4.3 End-to-End Multi-Path Model

At the beginning of this section, we defined the average distortion function at the client in (4). Subsequently, we derived each component in this equation step by step in terms of the path parameters. In doing so, we proposed two methods, namely path segregation and link aggregation. Together these two methods enable us to analyze any given two paths by a 3-link model as depicted in Figure 10. This model reduces the computation time required by the multi-path selection process.

Table 1: List of the notation and model parameters.

$\mathcal{N}_k$	Node k
$l_{u,v}$	Link between nodes $\mathcal{N}_u$ and $\mathcal{N}_v$
$b_{u,v}$	Bandwidth on link $l_{u,v}$
$p_{u,v}$	Packet loss rate on link $l_{u,v}$
$t_{u,v}$	Minimum delay on link $l_{u,v}$
$j_{u,v}$	Jitter on link $l_{u,v}$
$\mathcal{P}_k$	Path k
$D_{m,n}$	Distortion. m and n denote the arrivals of the 1^{st}
	and 2^{nd} descriptions, respectively
	$1 \rightarrow$ arrived on time, $0 \rightarrow$ lost or excessively delayed
$P_{m,n}$	Probability corresponding to $D_{m,n}$
B_k	End-to-end bandwidth on path $\mathcal{P}_k$
π_{NC}	Steady-state prob. for state NC (non-congested). $\pi_{\text{NC}} = \frac{\beta}{\alpha+\beta}$
π_{C}	Steady-state prob. for state C (congested). $\pi_{\text{C}} = \frac{\alpha}{\alpha+\beta}$
α	Transition probability from state NC to state C
β	Transition probability from state C to state NC
$L_{u,v}$	Average burst length on link $l_{u,v}$
p_{joint}	Packet loss rate on Network$_{joint}$
L_{joint}	Average burst length on Network$_{joint}$
$p^k_{disjoint}$	Packet loss rate on Network$^k_{disjoint}$
j_{joint}	Jitter on Network$_{joint}$.
	$j_{joint} = \sum_{\forall l_{u,v} \in \text{ Network}_{joint}} j_{u,v}$
$j^k_{disjont}$	Jitter on Network$^k_{disjoint}$.
	$j^k_{disjoint} = \sum_{\forall l_{u,v} \in \text{Network}^k_{disjoint}} j_{u,v}$
T_k	Minimum end-to-end delay on path $\mathcal{P}_k$.
	$T_k = \sum_{\forall l_{u,v} \in \mathcal{L}_k} t_{u,v}$
J_k	End-to-end jitter on path $\mathcal{P}_k$.
	$J_k = j_{joint} + j^k_{disjoint}$
T_{max}	Maximum tolerable delay
j^k_{max}	Maximum tolerable jitter on path $\mathcal{P}_k$.
	$j^k_{max} = T_{max} - T_k$

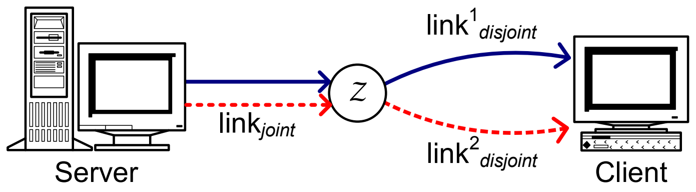


Figure 10: End-to-end multi-path model. 3-link model can be used to approximate any given two paths in a network.

In developing our multi-path model, we considered the packet loss probability independent from the delay jitter. In fact, both the jitter and packet loss experienced by a flow depend on the states of the queues encountered along the path. This implies a correlation between the delay jitter and packet loss probability (See Chapter 7 for details). However, in this study we used the time-averaged statistics for the packet loss probability. That is, we were interested in the average packet loss probability, not in the individual packet loss probabilities. Thus, we considered the delay jitter and packet loss probability on a link independent of each other.

2.5 Simulation Results

We have conducted two sets of simulations to demonstrate the efficacy of the proposed multi-path selection method in a real-time application. To this end, we first compared the performances of single description (SD) and multiple description (MD) encoded video streaming over a one-hop topology. This simple topology shed light on the unique features of MD coding and multi-path routing. Second, we generated a random Internet topology to compare different streaming methods in terms of their performances.

2.5.1 Methodology

We used three standard test sequences Table Tennis (352×240), Football (352×240) and Paris (352×288) in our simulations to stream from a server to a client with a delay tolerance of 200 ms. The Table Tennis and Football sequences exhibited large temporal variations and were comprised of 150 frames. In contrast, the Paris sequence contained 1050 frames, which possessed a relatively less temporal variation. To obtain statistically-reliable results, we streamed 10 minutes of video by concatenating each sequence several times. To make quantitative comparisons, we used the peak signal-to-noise ratio (PSNR) measure on the luminance (Y) channel, given by $\text{PSNR} = 10 \times \log_{10}(\frac{255^2}{\text{MSE}})$, where MSE stands for the mean-squared error between the original and decoded luminance frames. Among various multiple description video encoders, we chose the time-domain partitioning method with two descriptions.

The SD and MD encoded streams were produced with a standard MPEG-2 encoder (TM5 [85]) with default parameters, and a GOP structure consisting of an I-frame and nine P-frames. The video format was 4:2:0, and the frame rate was 30 frames per second. During the transmission of each video stream, the server created 576-byte Real-time Transport Protocol (RTP) packets [102, 103]. Note that we were not interested in sophisticated concealment techniques that could be applied to increase the streaming quality. In case of a bursty loss, we merely concealed the missing slices with the corresponding ones in the previously-decoded frame, whereas for the isolated losses, we used motion-compensated temporal interpolation to conceal the errors. Naturally, more sophisticated encoding and concealment techniques would have resulted in higher streaming qualities for both the SD and MD encoded streams.

In the simulations, we considered two types of links, namely "good" and "bad" links. We adopted consecutive NC (non-congested) and C (congested) states for each of these links to describe the bursty loss behavior as discussed in Section 2.4. While the duration of the non-congested states lasted for a random duration between 21 and 27 seconds for all links [7], we varied the durations of the congested states on different scales to characterize the good and bad links. In particular, this duration was chosen randomly between 20 and 200 ms for the good links, and between 0.5 and 4 seconds for the bad links [7]. These values, in conjunction with the inter-packet spacing, are then used to compute the corresponding GE parameters.

2.5.2 Single-Hop Topology Simulations

Let us consider the topology in Figure 11, where there are two 750 Kbps links (l_1, l_2) and one 1.5 Mbps link (l_0) between the server and client each with a delay of 50 ms. Suppose that the SD encoded video is streamed over the link l_0, and the MD encoded video is streamed over the links l_1 and l_2. When all the links are lossless, the average SD encoded stream (MD encoded stream) qualities are 33.02 dB (32.12 dB), 35.25 dB (34.19 dB) and 34.16 dB (33.32 dB) for the TABLE TENNIS, FOOTBALL and PARIS sequences, respectively. The 0.9 dB, 1.06 dB and 0.84 dB reductions in the respective PSNRs stem from the fact that the correlation between every other frame in the MD encoded video is less than that between every frame in the SD encoded video. However, this disadvantage is quickly compensated when the link loss rates are larger than 2% in case of the TABLE TENNIS and FOOTBALL sequences, and 3% in case of the PARIS sequence. Recall that the PARIS sequence does not exhibit a large temporal variation, which makes it more robust to longer bursts since its concealment is easier. For a 10% link loss rate, the respective average PSNR differences become around 2.06 dB, 1.40 dB and 0.9 dB in favor of the MD encoded video. Several average PSNR values versus different loss rates are given in Figure 12. We observe that if losing or delaying 2 - 3% or more of the packets is inevitable, MD streaming evidently outperforms SD streaming even when the best path is selected for SD streaming.

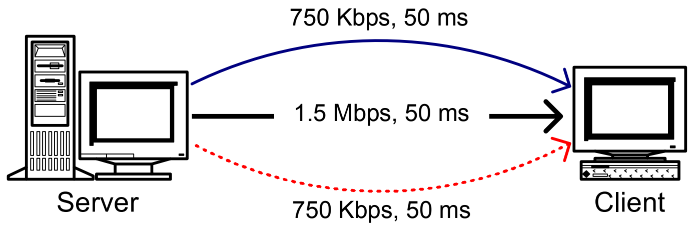
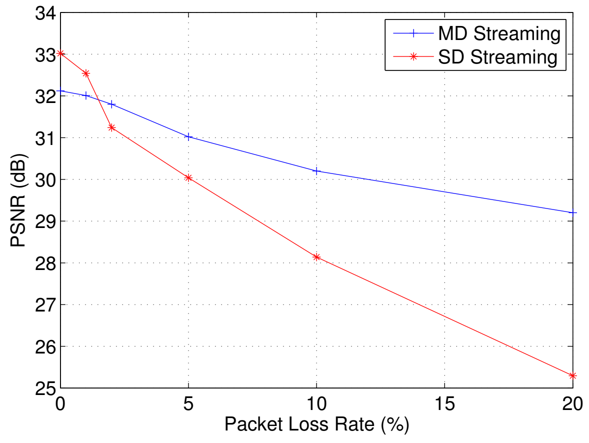


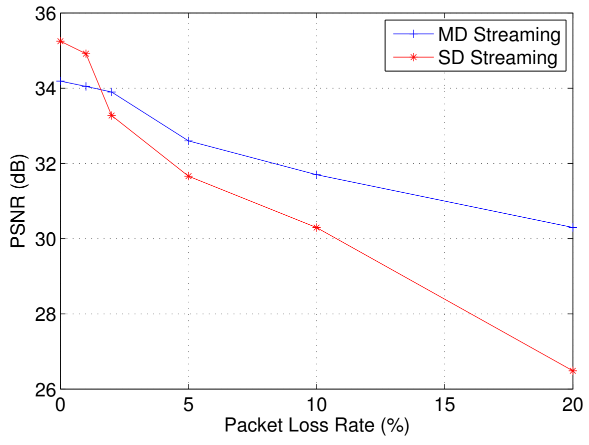
Figure 11: Single-hop topology with identical delays.

2.5.3 Internet Topology Simulations

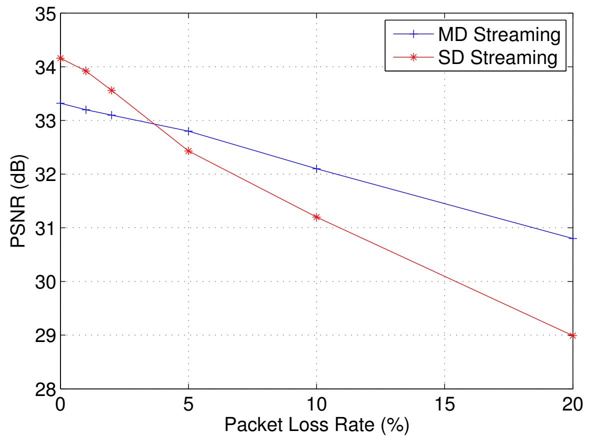
In the second set of simulations, we use a random Internet topology generated by the Georgia Tech Internetwork Topology Models (GT-ITM) [104]. The topology consists of 192 nodes with four mesh-connected transit domains, eight nodes per transit domain, one stub per transit-domain node and five nodes in a stub domain. An overlay node (O-node) is associated with each transit and stub-domain node to enable multi-path routing. A simplified topology with four transit domains and two selected paths are shown in Figure 13.



(a)



(b)



(c)

Figure 12: Variation of PSNR with the packet loss rate for the single-hop topology shown in Figure 11 for the Table Tennis (a), Football (b) and Paris (c) sequences.

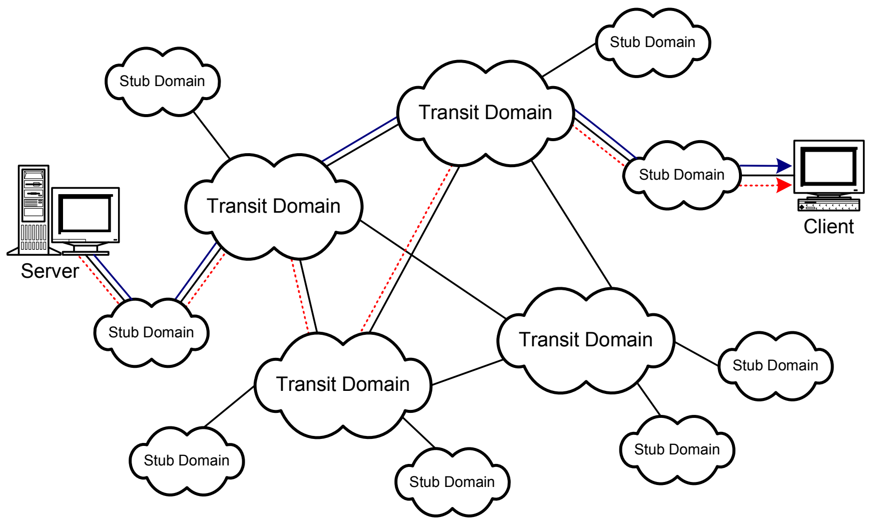


Figure 13: An Internet topology with four transit domains. Only a subset of the stub domains are shown for the demonstration purposes.

In this topology, the transit-domain links have a capacity of 10 Gbps and a delay between 10 - 30 ms, whereas the transit-to-transit and transit-to-stub edges have capacities of 1.0 Gbps and 100 Mbps, respectively, with a delay between 5 - 10 ms. On the other hand, stub-domain links are assigned 2 or 5 Mbps for the link capacity. For these links, the delay values are chosen between 5 - 10 ms.

To simulate a realistic Internet environment, we associate a random amount of background traffic with each link depending on its location in the range of 20% - 80% of its capacity. Because of the hierarchy in the Internet infrastructure, traffic flowing through different domains is aggregated on the inter-domain links (*e.g.*, transit-to-transit edges and transit-to-stub edges). Consequently, some of these links carry more traffic and experience congestion more often. Hence, we characterize these links as bad, whereas we consider the rest as good links. The loss parameters of each link are then determined as discussed in Section 2.5.1. Similarly, jitter values are assigned between 0 - 10 ms and 10 - 30 ms for the good and bad links, respectively.

On this topology, we select a server-client pair such that the server and client are separated by two transit domains. We compare four different streaming methods: (*i*) shortest-path, (*ii*) maximally link-disjoint path, (*iii*) redundant-path, and (*iv*) optimal multi-path streaming. For shortest-path streaming, we choose the path with the minimum number of hops; if this metric is equal for two or more paths, we select the one with the minimum end-to-end delay. For maximally link-disjoint path streaming, we first generate all possible path pairs. Then, we search for the pairs with the minimum number of overlapping links to minimize the statistical dependency between the paths and select the pair with the minimum total end-to-end delay. Note that in doing so, we not only approximate a totally link-disjoint path pair, but also select a shorter one to avoid the unnecessary quality degradation due to the excessively-delayed packets. The third streaming method is adopted from [77], where a redundant path is selected to minimize the number of joint links with the default Internet path. For simulations, we assume that the default Internet path is identical to the shortest path used in the first method. Finally, for optimal multi-path streaming, we select the pair among all possible ones that minimizes (4). We stream the MD encoded video in all methods except the first one, where we use the SD encoded video.

Finding the optimal path pair that minimizes (4) is computationally intensive. The full-enumeration approach we use in the simulations clearly does not scale well as the number of O-nodes, and hence, the number of possible paths increases. However, it is important to note that the contribution of this chapter is to provide a framework on modeling MD streaming over an overlay network and prove the concept of multi-path selection with the simulations. Yet, in Appendix A, we develop a fast heuristics-based solution for the optimal multi-path selection problem. This heuristic, although still runs in exponential time in the number of transit domains, is much more efficient for real systems where the number of the nodes in each transit domain is substantially larger than the number of transit domains.

For each test sequence, we run 10 simulations over the same topology with a new set of link attributes at each run. However, these attributes are not altered during a particular run. Recall that the O-nodes periodically probe each other and the server is notified about the changes in link attributes. Hence, the effect of these changes in our results is expected to be small. In the last two simulations (runs #9 and #10), we convert the stub domain to which the server is connected, into a multi-homed stub domain in order to enhance the path-diversity capability, as depicted in Figure 14. This allows us to observe the performances of different streaming methods with varying network conditions. Note that we keep the total rate equal for both the SD and MD encoded videos, although multi-path routing might be exploited to increase the bandwidth available to the user [105]. To make a fair comparison, at each run, the total rate for all streaming methods is set to the rate of shortest-path streaming[5].

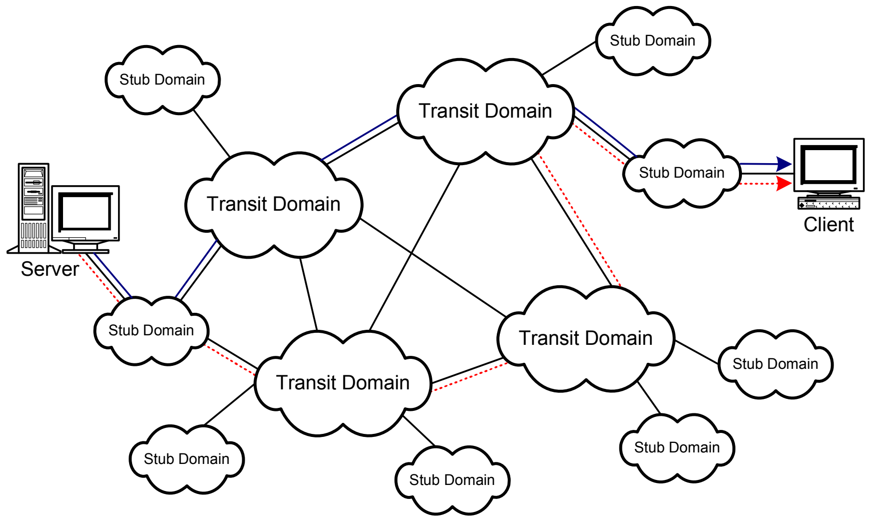


Figure 14: Enhanced path-diversity capability when the server is located in a multi-homed stub domain.

The results of all three test sequences are tabulated in Tables 2 - 4, respectively. In each row, we present the total rate and average PSNR of the streamed video for a different run. The results are also plotted in Figures 15 - 17. Optimal multi-path streaming outperforms shortest-path, maximally link-disjoint path and redundant-path streaming in all of these independent runs. There are prominent conclusions that can be drawn from these results. We elaborate on them next.

2.5.4 Discussion

When we examine the results, we observe that shortest-path streaming often degrades the quality to an unacceptable level. This occurs when a link on the shortest path is heavily congested. Naturally, maximally link-disjoint path and redundant-path streaming are

[5]The actual rate of streaming is kept lower than the available end-to-end bandwidth in order not to overwhelm the bottleneck link(s).

Table 2: Simulation results for the Table Tennis sequence.

Run #	Total Rate (Mbps)	Shortest-path PSNR (dB)	Max. Link-dis. Path PSNR (dB)	Redundant-path PSNR (dB)	Opt. Multi-path PSNR (dB)
1	1.60	31.50	30.74	31.45	31.66
2	1.25	25.70	28.01	27.62	28.53
3	1.45	30.50	29.52	30.63	31.60
4	1.60	30.23	30.35	29.89	32.02
5	1.40	25.70	29.30	28.65	31.32
6	1.35	29.30	30.10	30.03	30.93
7	1.50	30.60	29.95	29.77	31.29
8	1.45	25.77	30.20	29.11	31.44
9	1.50	30.12	31.11	30.88	31.39
10	1.40	26.23	29.05	28.94	30.89
Avg.	1.45	28.57	29.83	29.70	31.11
Max	1.60	31.50	31.11	31.45	32.02
Min	1.25	25.70	28.01	27.62	28.53

Table 3: Simulation results for the Football sequence.

Run #	Total Rate (Mbps)	Shortest-path PSNR (dB)	Max. Link-dis. Path PSNR (dB)	Redundant-path PSNR (dB)	Opt. Multi-path PSNR (dB)
1	1.60	33.60	32.65	33.32	33.97
2	1.25	27.09	29.66	28.88	30.33
3	1.45	32.42	31.55	32.73	33.48
4	1.60	31.98	32.22	31.45	33.92
5	1.40	27.22	31.60	30.79	33.29
6	1.35	31.26	32.13	32.27	32.76
7	1.50	32.93	31.84	31.66	33.45
8	1.45	27.56	31.96	31.32	33.46
9	1.50	31.96	33.09	32.97	33.33
10	1.40	27.49	30.86	30.22	32.67
Avg.	1.45	30.35	31.76	31.56	33.07
Max	1.60	33.60	33.09	33.32	33.97
Min	1.25	27.09	29.66	28.88	30.33

Table 4: Simulation results for the Paris sequence.

Run #	Total Rate (Mbps)	Shortest-path PSNR (dB)	Max. Link-dis. Path PSNR (dB)	Redundant-path PSNR (dB)	Opt. Multi-path PSNR (dB)
1	1.60	32.70	31.32	32.42	33.10
2	1.25	29.03	29.42	29.23	29.76
3	1.45	31.55	31.63	30.67	32.56
4	1.60	30.76	31.17	30.33	32.95
5	1.40	29.29	30.44	29.88	32.43
6	1.35	30.46	31.21	30.59	31.98
7	1.50	31.37	30.96	30.43	32.26
8	1.45	29.62	31.09	30.66	32.20
9	1.50	31.02	32.15	31.74	32.34
10	1.40	29.52	30.01	29.88	31.82
Avg.	1.45	30.53	30.94	30.58	32.14
Max	1.60	32.70	32.15	32.42	33.10
Min	1.25	29.03	29.42	29.23	29.76

expected to reduce the adverse effects of the congested links by exploiting the path diversity and improve the average quality over shortest-path streaming. However, despite the enhanced path-diversity capability in runs #9 and #10, maximally link-disjoint path and redundant-path streaming are still not able to provide the steady performance of optimal multi-path streaming. Hence, we conclude that minimizing the dependency between the paths without considering their characteristics does not necessarily result in the maximal streaming quality.

Based on our simulations, we cannot conclude whether maximally link-disjoint path streaming is superior to redundant-path streaming, or not. Generally speaking, redundant-path streaming performs better when the shortest path is not heavily congested. If the shortest path is exposed to bursty losses or long delays, maximally link-disjoint path streaming delivers a better quality of video.

Interestingly, in run #2 for all three test sequences, it is observed that the video quality suffers in all four streaming methods. By examining the link attributes, we identify the cause as being a heavy congestion on the link over which the server's stub domain is connected

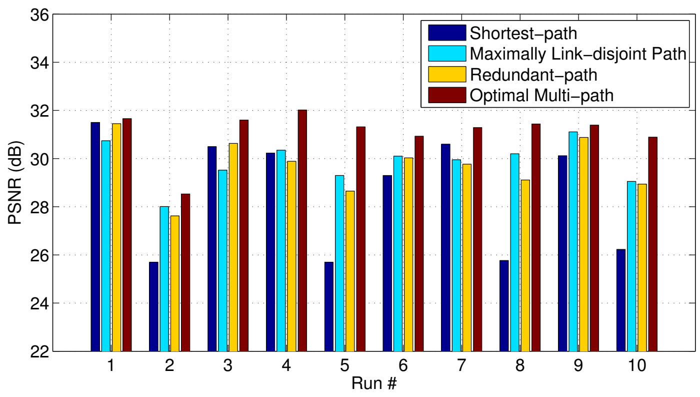


Figure 15: Average PSNRs for different streaming methods for the Table Tennis sequence.

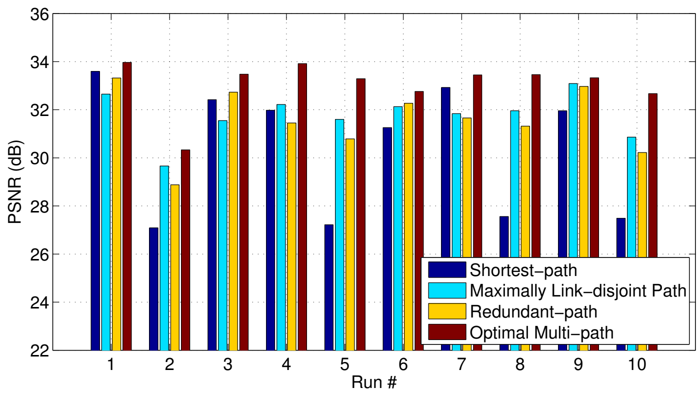


Figure 16: Average PSNRs for different streaming methods for the Football sequence.

to its transit domain. The poor video quality is inevitable since all streams have no choice other than to be routed over the same congested link. Obviously, path diversity becomes less effective when the congestion causes simultaneous losses on both descriptions.

Another interesting result is the variation of the individual frame PSNRs. The plots in Figures 18 - 20 compare the PSNR values for a set of 150 frames for the videos obtained from different streaming methods. These illustrative plots are extracted from run #3 for the Table Tennis sequence. Figure 18 shows that the quality of the SD encoded video suffers from bursty losses and the subsequent error propagation. The resulting wide fluctuations further degrade the perceptual quality. Similarly, the maximally link-disjoint path and redundant-path streaming methods also degrade the quality because of the late and lost packets, but to a lesser extent, as depicted in Figures 19 and 20, respectively. However, the video quality slightly deviates from its average value for optimal multi-path streaming unless

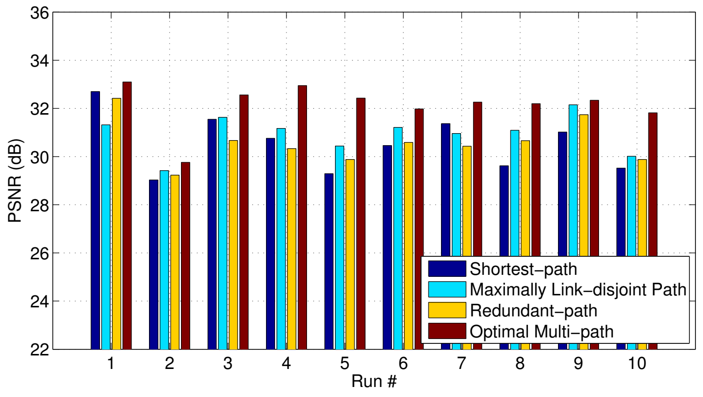


Figure 17: Average PSNRs for different streaming methods for the Paris sequence.

both descriptions are lost simultaneously. This feature delivers a more stable streaming quality to the client.

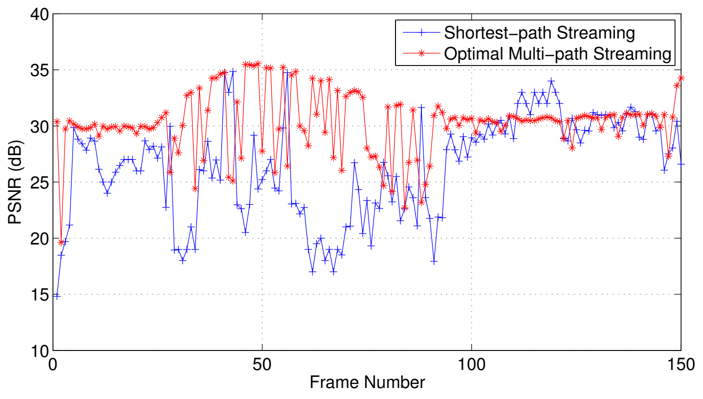


Figure 18: Variation of the frame PSNRs for the shortest-path and optimal multi-path streaming methods for the Table Tennis sequence.

As a final remark, we observed the tendency of multi-path routing to load balancing throughout the simulations. Although we did not analyze the resource usage quantitatively in this study, we observed that multi-path routing helps load balancing, which in turn results in better resource utilization. Rather than overloading the shortest path, the total load is shared between different paths.

2.6 Conclusions

In this study, we first developed models for multi-path streaming and verified the validity of these models through extensive simulations based on Internet topologies and traffic characteristics. Second, we proposed an optimal multi-path selection method for MD streaming.

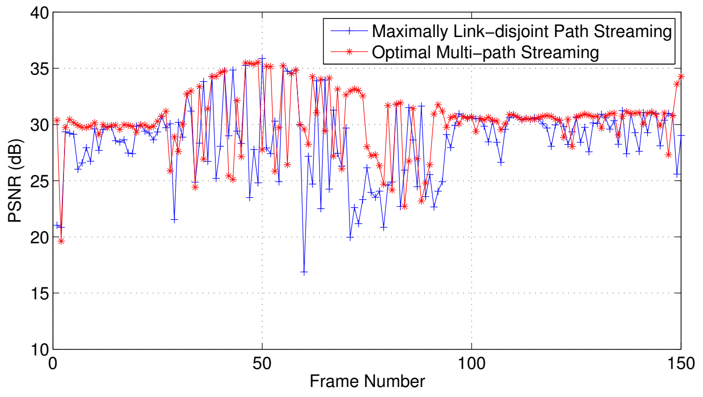


Figure 19: Variation of the frame PSNRs for the maximally link-disjoint path and optimal multi-path streaming methods for the TABLE TENNIS sequence.

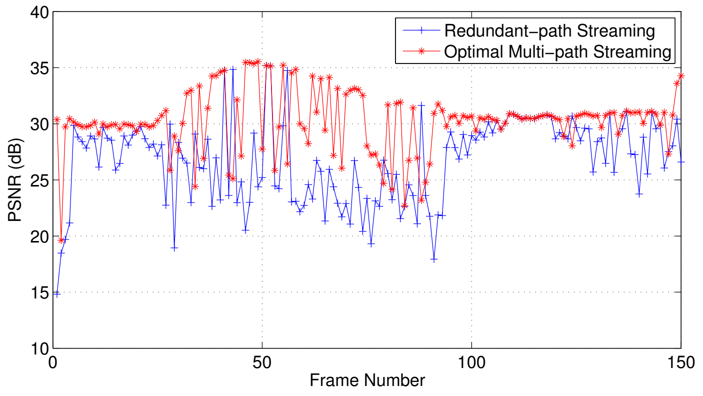


Figure 20: Variation of the frame PSNRs for the redundant-path and optimal multi-path streaming methods for the TABLE TENNIS sequence.

This application-aware method yields the maximal video quality at the client by evaluating the application requirements and network conditions. The simulations with several test sequences show that the optimal multi-path selection attains a considerable amount of quality improvement over the previously-proposed path selection methods. On the implementation side, we carry out MD streaming with the use of an overlay infrastructure. Hence, we do not advocate any particular routing mechanism in the underlying physical network.

In this study, we paid attention to the two-path case due to its simplicity and efficacy, however, it is straightforward to generalize the proposed method to more than two paths at the expense of increased computational complexity. Moreover, the framework can be adapted to other streaming media such as audio and 3-D graphics by incorporating their respective distortion models.

CHAPTER III

HIGH-RESOLUTION VIDEO STREAMING TECHNIQUES IN MESH-NETWORKED HOMES

Wireless mesh systems offer several advantages for emerging high-bandwidth networks because of their cooperative routing capabilities. In this study, we consider the potential benefits of using wireless meshes within residential networks for applications that require high bandwidth and low latency. In particular, we consider high-bitrate video transmission inside a mesh-networked home. To quantify our findings, we present experimental results obtained from a high-resolution video streaming application. Our experiments involve single-hop, multi-hop, single-path and multi-path transmission methods, and two types of video-encoding techniques, namely single description coding and multiple description coding. In the light of our results, we discuss the pros and cons of each streaming method.

3.1 Introduction[†]

Wireless mesh networks (WMNs) are ad hoc networks in the general form of a full or partial mesh topology. A fundamental feature of a WMN is that the mesh nodes can relay a connection to any other node within their ranges. In other words, a mesh node operates not only as an end-point but also as a router that forwards packets belonging to other nodes that cannot directly transmit their packets to their destinations. Generally speaking, there are two types of nodes in a WMN: *mesh routers* and *mesh clients*. Mesh routers have minimal mobility and form the backbone of the network. In addition to routing, these nodes provide network access for both mesh and conventional clients, *e.g.*, desktops, laptops, PDAs and PocketPCs, by bridging connections over multiple wireless interfaces. In contrary, mesh clients do not possess bridging and gateway functionalities, although they

[†]Parts of this chapter were previously published in [106].

may still perform routing.

A WMN is dynamically self-organized and self-configured, with the nodes in the network automatically establishing and maintaining mesh connectivity among themselves. Similar to the overlay networks in the Internet, a mesh network offers multiple redundant communication paths throughout the topology. If one link fails for any reason such as node failure or strong RF interference, messages are routed through alternative paths. If there are mobile nodes inside a WMN, they can change their positions over time to improve the connectivity of the network. In this sense, WMNs are self-forming and self-healing networks. Another important feature is that WMNs are decentralized, *i.e.*, there is no central control unit.

Advantages of WMNs include rapid deployment due to simplified installation and configuration, flexibility, fault tolerance, low upfront costs due to minimal cabling needs, easy network maintenance, and adaptability. These advantages enable many new application areas in broadband home networks, community and neighborhood networks, enterprise networks, and metropolitan area networks. In this study, we particularly focus on the WMNs deployed inside homes. As our target application, we consider high-resolution video transmission over wireless links, which, from a consumer point of view, will be one of the most influential developments in the home entertainment market. We can presumably expect that the proliferation of high-quality video systems such as the high-definition television (HDTV) and next generation DVDs, will be further boosted by a technology that will enable the consumers to transmit these videos to anywhere inside their homes over the air. Not far in the future, the range of high-bandwidth WMNs will be beyond residential areas. For a discussion on the experiences from a pilot metropolitan mesh network, see [107].

With various physical-layer technologies, and medium access control (MAC) and routing protocols available today, deploying a WMN is not too difficult. However, it is quite challenging to make all the mesh components cooperate together and work seamlessly. Without taking into account the unique characteristics of mesh networking and developing specific protocols, the success of WMNs will be limited. Ongoing research efforts indicate that all existing protocols from the application layer to the physical layer require enhancements, if not a total re-engineering from scratch [108]. Cross-layer integration is also an important issue for WMNs. Given that multiple data and media flows coexist in a WMN with different QoS requirements, it is a difficult task to allocate the available resources to the individual connections. Transport-layer protocols such as TCP and UDP, and routing protocols such as dynamic source routing (DSR) [109] have to adapt themselves to topology and link-quality changes. Topology/link monitoring and coordination among the mesh nodes can be accomplished by using Jini [110], universal plug and play (UPnP) [111] and other link-level schemes [108, 112][1].

Having given a brief introduction to WMNs, we continue our discussion with the details of the application scenario and methodology adopted in this study in Section 3.2. In Sections 3.3 and 3.4, we present several results obtained from single-path and multi-path video streaming experiments, respectively. We conclude the chapter in Section 3.5.

3.2 Residential Applications

A promising application of WMNs is wireless broadband home networking. Currently, majority of the broadband home networks are realized through IEEE 802.11x WLANs, where 802.11x-capable clients are interconnected by the help of a single router, which also functions as a gateway to the Internet. Since there exists only one communication path between any client pair and each of these paths goes through the same router, the scalability is severely limited. The interference at the physical layer and the contention at the MAC layer limit the maximum throughput achievable by a single flow. Hence, bandwidth-intensive applications can be hardly supported. In contrast, if all network-capable devices are interconnected with each other through multiple mesh routers, the resulting mesh network provides more flexible and robust paths between the clients, and increases the end-to-end throughput. An example of a wireless mesh home network is shown in Figure 21.

[1]Universal plug and play (UPnP) is a set of protocols developed for allowing devices to connect seamlessly and simplifying the implementation of data sharing, communications, and entertainment in home networks and corporate environments.

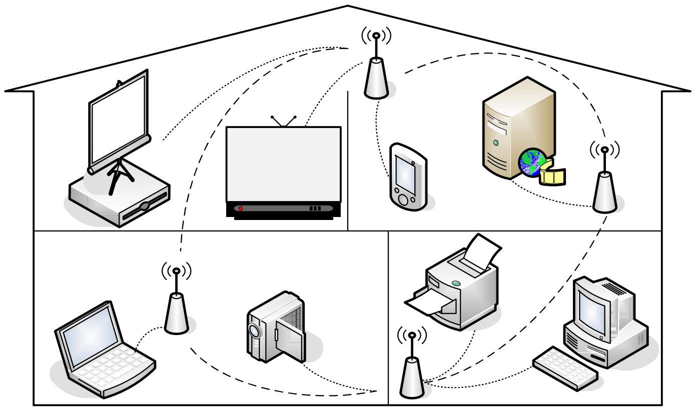


Figure 21: A wireless mesh home network.

3.2.1 HDTV Video Transmission

HDTV is a digital television standard that offers high-resolution, superior video and sound distribution via satellite, terrestrial and cable TV systems. Although there are several different recommendations for HDTV video signal format, transmission of a single HDTV stream often requires a bandwidth larger than 10 Mbps. This high-bandwidth requirement renders HDTV streaming applications almost impractical over existing Internet links, however, it is not as difficult over mesh-networked homes as we today have the technologies that can physically accommodate such demanding applications. Note that in addition to broadcast HDTV, end-users can also stream high-resolution videos from other sources such as DVD players, PVRs and computers, and play them on an HDTV-capable display.

3.2.2 Methodology

In order to conduct a performance analysis of different streaming methods in wireless mesh networks, we generated a six-node mesh topology inside an office of 1,800 sq. ft. Each node was fixed during the experiments and located such that it sustained a good channel quality over 802.11a links with at least two other nodes in the network. A sketch of our topology is given in Figure 22, where we have a video source node, a destination (*e.g.*, display) node and four intermediate relay nodes.

We developed a server/client application suite for streaming high-resolution pre-encoded video sequences over UDP. We encoded the test sequence Soccer (4CIF, 704×576 pixels) with a standard MPEG-2 encoder (TM5 [85]) at a bitrate of 10 Mbps and a frame rate of 30 frames per second. Each GOP consisted of one I-frame, four P-frames and 10 B-frames. The encoded stream had an average frame quality of 38.9 dB. The test sequence consisted of 300 frames, however, we played the same sequence 10 times to get a 100-second video that corresponded to over 85,000 IP packets. Note that at the physical layer, 802.11a has a peak rate of 54 Mbps, however, this rate can degrade significantly due to the potential interference from other sources operating at the same frequency. In addition, the bursty nature of many video sources can further degrade the channel utilization. Hence, although the network can theoretically sustain higher transmission rates, we are still challenged to transmit video sequences at 10 Mbps.

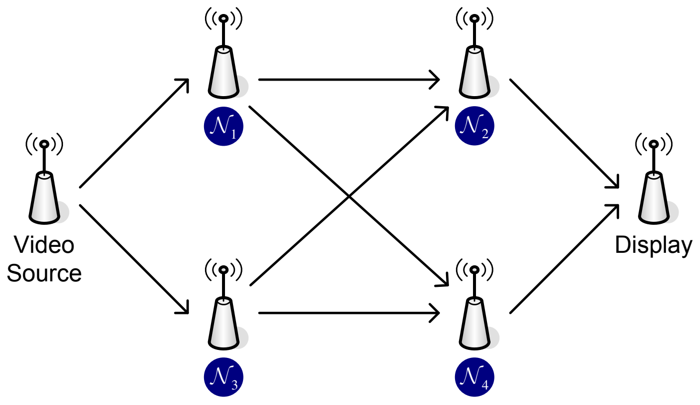


Figure 22: Experimental 802.11a mesh topology.

3.3 Single-Path Streaming

In this section, we study video transmission over a single path. Due to their mesh structure, mesh nodes potentially offer several different paths between a node pair. For example, in Figure 22, a path between the video source and display can be established via nodes $\mathcal{N}_1$ and $\mathcal{N}_2$. Another one can be established via nodes $\mathcal{N}_3$ and $\mathcal{N}_4$. We will refer to these paths as Path-1 and Path-2, respectively. The performance of each path depends on the quality of the channels between the nodes making up the path. In Figure 23, the delay distributions of the packets that are transmitted over each path are plotted. In addition, we also plot the delay distribution corresponding to the single-hop path between the video source and display. That is, the source transmits the video packets directly to the display node without relaying over any other node. Although this might initially seem a better transmission scheme, as it actually avoids relaying delays, the results in Figure 23 suggest that the single-hop path is subject to a larger amount of channel interruptions due to its increased range. The resulting increased packet delays inevitably disturb the video quality under small playout-delay requirements. This experiment is an example where using multi-hop transmission performs better than single-hop transmission. In fact, wireless mesh nodes exploit this feature to enable higher throughput and reduced latency for bandwidth-demanding applications [108].

Due to the interference at the physical layer, some packets will not go through in their first transmission attempt at the MAC layer. Fortunately, 802.11a MAC recovers most of these packets by doing retransmissions. Although this behavior reduces the effective packet loss rate as seen from the application layer, recovered packets may experience large delays (up to several hundreds of milliseconds) since the transmitter backs off exponentially after each retransmission. If the application cannot tolerate such late packets, they will be considered as lost. Note that despite the retransmissions at the MAC layer, some packets may still not be delivered to the other end, if the maximum retransmission limit is reached. In our experiments, we observed that less than 1% of the packets failed after 16 transmission attempts. If desired, higher-layer error-control/protection methods can be used to recover these packets at the expense of increased delay.

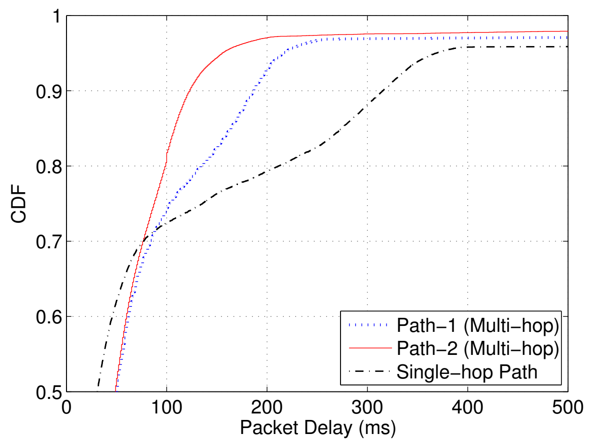


Figure 23: Delay distributions for the packets transmitted over multi-hop and single-hop paths.

An interesting observation in Figure 23 is that Path-2 performs significantly better than Path-1, when the video application adopts a playout delay of 200 ms or less, since Path-1 goes through a larger number of obstacles. This disparity brings the following question to the fore: "How can we determine the path that will deliver the highest video quality?" Given the highly dynamic nature of wireless channels, it is often difficult to determine which path will perform better at a given time.

Two common approaches to base the path selection/switching decision are to (*i*) actively probe the channels, and (*ii*) passively collect channel statistics via monitoring. After evaluating the information gathered from the measurements, a proper path selection and switching (if necessary) can be made. For example, consider Figure 24 where we plot the packet delay traces for Path-1 and Path-2. Suppose that we seek the path that delivers the most packets within 200 ms. Since Path-2 statistically delivers more packets on time compared to Path-1, we easily decide to stream the video packets over Path-2. However, Figure 24 shows that Path-1 delivered some packets, *e.g.*, packets #3235, #3238 and #3293, within 200 ms, which actually could not be delivered on time by Path-2. In other words, Path-2 has a higher on-time delivery rate on the average, but Path-1 may still deliver some packets on time that are not delivered by Path-2.

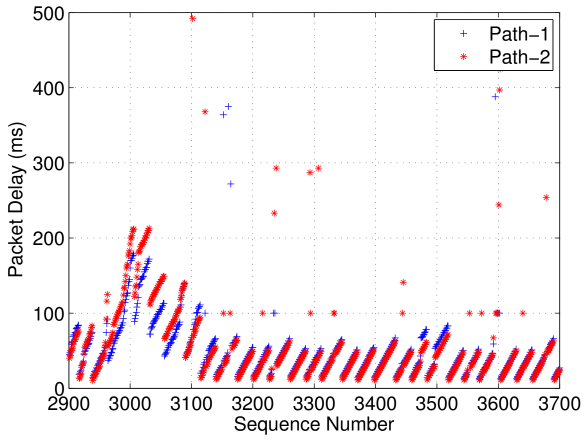


Figure 24: Packet delays observed from two different paths.

Although channel interruptions of long durations can be easily identified via channel probing or passive measurements, detecting the short ones is not as trivial. By the time the feedback/status information is received, the validity of this information may have already expired, and it would be unprofitable to take an action, *e.g.*, switch to another path. A particularly useful technique to overcome the shortfalls of single-path streaming is to utilize multiple paths between the end-points. In the next section, we study multi-path video transmission in two different scenarios.

3.4 Multi-Hop Multi-Path Streaming

As we discussed in Chapter 2, exploiting path diversity can be an effective technique to provide robustness and improved reliability against the network congestions in the Internet. In low-delay favoring applications, the availability of a secondary path becomes particularly useful for maintaining a continuous video, when the primary path fails or performs poorly. With the same motivation, multi-path video transmission can be applied to wireless mesh networks. Actually, the existence of a mesh topology and the ability of source routing are two enabling factors for the multi-path transmission in mesh networks.

Based on Figure 23, we quantify that under the requirement of a playout delay of 200 ms, Path-1 and Path-2 have an on-time delivery rate of 93.5% and 97.0%, respectively. Majority of the remaining packets are late for decoding. These packets cause a buffer underrun at the receiver, and consequently, the decoder freezes the video. Moreover, Figure 24 shows that some packets are delivered on time only by Path-1 and some others only by Path-2. Thus, if we simultaneously send each packet over both paths, the number of on-time packets can be increased. In doing so, we actually achieve an on-time delivery rate of 98.0%, which corresponds to 4.5% and 1% improvement over streaming over only Path-1 and only Path-2, respectively, as shown in Figure 25. Although duplicate-packet transmission leads to a bandwidth waste and potentially harms other flows in the network, its use can be still justified in mesh-networked homes because of the following two reasons: First, home networks are private networks. The residents have the full ownership and control over all network resources. These resources can be exploited to the maximum extent to satisfy the QoS requirements of the users. Second, with the introduction of recent advances in wireless technologies such as the ultra wideband systems, the bandwidth will be less of an issue, and mesh-networked homes will soon offer large bandwidths that can sustain multiple video flows along with several other data flows.

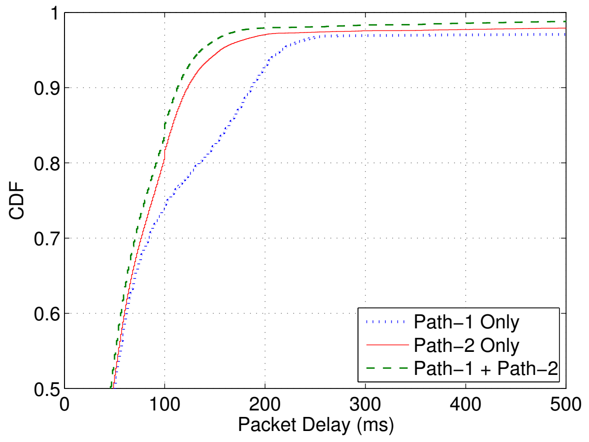


Figure 25: Delay distributions for the packets transmitted over single and multiple paths.

Nevertheless, in some legacy mesh networks where the bandwidth is a scarce resource, duplicate-packet transmission capability can be limited. In that case, we can exploit the advantages of multi-path transmission by distributing the video packets among different paths while keeping the total transmission rate unchanged with respect to the single-path transmission. To this effect, an elegant approach is to encode the source video with a multiple description (MD) encoder [42] and generate multiple independently-decodable descriptions. Each of these descriptions individually produces a basic quality of video. With additional description(s), the quality is refined via joint decoding. An important feature of MD decoders is that the successful reception of at least one description suffices for a continual decoding operation.

As discussed in Section 2.2, there are several ways for generating multiple descriptions. A straightforward method is time-domain partitioning [43], which separates the even and odd-numbered video frames into two groups, and encodes them individually to produce two descriptions. This method is naturally suitable for balanced mode of MD coding (B-MD), where each description is encoded at the same bitrate. However, if there is a substantial performance difference between the paths, the sender may opt to encode the descriptions at unequal bitrates. This is called unbalanced mode of MD coding (UB-MD). Recall that the single description (SD) encoded SOCCER sequence had an average frame quality of 38.9 dB. The average frame quality drops to 37.8 dB in case of a B-MD encoder. The 1.1 dB reduction stems from the fact that the SOCCER sequence exhibits large temporal variation, and the correlation between every other frame in the MD encoded video is less than that between every frame in the SD encoded video. We present our experimental results for SD and MD video transmission in Figure 26, where we plot the average video quality in terms of the peak signal-to-noise ratio (PSNR) measure on the luminance (Y) channel against the playout delay. We elaborate on the results next.

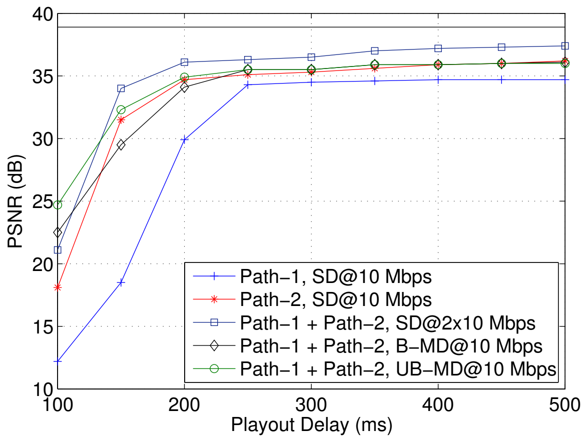


Figure 26: Average PSNR variation with the playout delay.

Our first observation is the rapid increase in the number of late packets when a playout delay shorter than 200 ms is required. Under a delay requirement of 100 ms, both UB-MD and B-MD encoded videos achieve a higher (but still not pleasing) video quality than all three SD encoded videos because of the better error-concealment capability of the multiple descriptions. In addition, UB-MD encoded video outperforms its B-MD counterpart by 2.2 dB as the latter one is disturbed by the inferior performance of Path-1 to a larger extent. In case of playout delays of 150 and 200 ms, both Path-1 and Path-2 deliver more packets on time. Yet, existing shortfall of Path-1 causes the B-MD encoded video to produce 0.6 - 2.0 dB lower quality than the SD encoded video streamed over Path-2, although the UB-MD encoded video still outperforms the SD encoded videos streamed over Path-1 and Path-2 by 5.0 - 13.8 dB and 0.2 - 0.8 dB, respectively. When we increase the playout delay beyond 250 ms, both paths perform sufficiently well. Consequently, UB-MD coding loses its advantage over SD coding. At the same time, B-MD encoded video closes the gap with the SD encoded video and produces an equal quality of video.

Not surprisingly, duplicate-packet transmission delivers the highest quality of video at the expense of doubling the consumed bandwidth. In all cases except when the playout delay is shorter than 200 ms, duplicate-packet transmission offers a decent video quality (within 2 dB of the original video quality), without employing any high-layer error control/protection method.

3.5 Conclusions

In this study, we explored different methods for transmitting high-resolution video streams inside a mesh-networked home. By building a six-node network, we investigated the potential of each method. Namely, we compared single-path streaming with multi-path streaming, and single description coding with multiple description coding. Generally speaking, exploiting path diversity with either single or multiple description coding provides higher quality and more reliable video. Our experiments showed that the achieved video quality heavily depended on the path(s) over which the video packets were streamed. In parallel to the results presented in Chapter 2, it is clear that a proper path selection/switching is critical to the success of video transmission in mesh networks.

CHAPTER IV

IMPROVING VIDEO-ON-DEMAND OVER IP EXPERIENCE IN MULTI-SOURCE NETWORKS

In this chapter, we study the simultaneous streaming of packetized video from multiple servers to a single client over an IP network. We explore the problem of multi-server streaming in two parts:

- In the first part, we do not limit ourselves to a particular video-encoding scheme. We derive a generalized framework and develop a client-driven rate-distortion optimal packet scheduling algorithm that decides which packets to be requested from which servers at a given request opportunity such that the rendered video quality at the client is maximized while the rate constraints dictated by the flow, window and congestion control mechanisms are satisfied for each server. With simulation results, we demonstrate the advantages of multi-server streaming and show that our packet scheduling algorithm achieves a higher video quality compared to optimized single-server streaming.

- In the second part, we discuss how the problem of rate-distortion optimized multi-server streaming evolves with the choice of a special video-encoding scheme and illustrate how this new problem can be solved by using media-processing methods.

4.1 Introduction[†]

True video-on-demand (VoD) has long been thought of as the ultimate consumer video application. Currently deployed technologies, such as near video-on-demand (n-VoD), have left consumers with few content choices and without the interactivity necessary to allow the content providers to exploit all the potential in this market[1]. Similarly, VoD over IP and IP

[†]Parts of this chapter were previously published in [113, 114].

[1]n-VoD allows the clients to select and view a movie scheduled for a given time. It enables operators to increase their revenue while keeping the bandwidth requirements low.

television (IPTV) could not scale to a point to be of interest to consumers because of the perpetual lack of a stable bandwidth. Unavoidably, the fluctuating bandwidths resulting from network congestions in IP networks disturb the continuity of the delivered video. This problem restricts the content providers to low encoding rates, which are not high enough to support the picture quality they want to deliver. Moreover, these low rates also do not fully utilize the capacity of broadband subscribers.

As a design feature, VoD networks are generally equipped with several video servers at a location, each of which is synchronized with the same content. This ensures that if a primary server fails, a secondary backup server automatically substitutes for the primary one. A more practical approach, on the other hand, is to deploy content servers at different locations inside the network. In addition to being a failure-recovery solution, this type of content replication can be an efficient and effective method to reduce the adverse impact of network congestion. For example, content delivery networks (CDNs) are used to deliver the content on behalf of the origin content servers by utilizing caches at points close to end-users. In doing so, CDNs increase the robustness and reliability of the service. However, clients are still served by a single server within the CDN. Unfortunately, in a streaming scenario, no matter how well it is optimized, the service of a single server may still suffer from inevitable intermittent congestions, and server or link failures. One way to maintain a good level of service quality is to exploit the potential benefits of using multiple servers concurrently. Multi-server support virtually eliminates the large buffering delays, allows instant-on/always-on playback and instantaneous channel changes, and reduces the number and duration of the interruptions. Since the required infrastructure and the content servers are already deployed, we believe that this solution will be readily applicable in supporting the desired QoS for VoD services.

To deal with the scalability and fairness issues, multi-server streaming requires a protocol that manages the servers simultaneously. Let us crystalize this point on a VoD system with one client and two servers, where the client explicitly sends request packets to the servers asking them to send particular video packets. Note that VoD systems require minimal pre-buffering at the client side to support full interactivity between the client and VoD servers, and enable the VCR-like functionalities such as fast forwarding and rewinding. In the most straightforward case, the client may request every packet from each server. Subsequently, two copies of each packet are introduced into the network, making an inefficient use of the available resources. This approach explicitly attempts to improve the expected streaming quality at the expense of an increased rate. A more conservative approach would be to distribute the packets evenly among the servers to keep the total consumed rate equal to that of a single-server streaming system. In this case, the achieved quality depends on the loss and delay characteristics of the paths between the servers and the client. Yet, another approach might be to request the packets whose decoding deadlines are approximately within one round-trip time duration from both servers to maximize their chances of being received on time, to request those packets whose decoding deadlines are in the distant future from only one server, and not to send any request at all for the packets whose decoding deadlines have almost passed. These scenarios plainly present a rate-distortion trade-off, and this trade-off has to managed optimally by the client. Generally speaking, with multi-server support one can achieve high aggregated streaming rates. However, it is imperative to comply with the imposed rate constraints so that multi-server streaming does not impair other network-friendly flows.

Although the use of multiple servers has long been studied and exploited for data applications [115, 116], it has only found interest recently in real-time video streaming. To this end, the most closely related works are [62, 117]. These studies propose network-friendly algorithms for distributed video streaming. In [117], an MD-FEC (multiple description coding through forward error correction codes) encoding framework is used to achieve a target quality with the minimum bandwidth usage. In [62], the authors apply a rate allocation algorithm to determine the rate for each server, and a packet partition algorithm to ensure that no packet is sent by more than one server. These algorithms, in conjunction with FEC, try to minimize the probability of lost and late packets. Although both [62, 117] use the idea of utilizing multiple servers for video streaming, our work substantially furthers these studies in several aspects.

The rest of the chapter consists of two main sections. In Section 4.2, we develop a generalized rate-distortion optimal packet scheduling algorithm for single description video streaming in a multi-server environment and analyze its performance. With simulation results, we show that multi-server single description streaming achieves a higher video quality compared to optimized single-server streaming. In Section 4.3, we illustrate how a special video-encoding scheme can be incorporated into the rate-distortion optimization. In particular, we consider rate-distortion optimized multiple description video streaming where each description is streamed from a different server. We compare the performance of optimized multi-server multiple description streaming to that of its non-optimized counterpart. Our analysis shows that the former can still deliver a good quality of video, while the latter cannot, when the network paths leading to the client share the same bottleneck link. We conclude the chapter in Section 4.4.

4.2 Multi-Server Single Description Video Streaming

4.2.1 The Problem and Definitions

Consider the streaming system sketched in Figure 27, where there are one client and two servers, which are denoted by $\mathcal{S}_1$ and $\mathcal{S}_2$. Suppose that the encoded video is packetized into video packets, and these packets are replicated at both servers. In this system, the client determines the order in which it will request the video packets from the servers at given request opportunities. The servers are completely passive in this scheduling process, and they merely respond to the client's requests. In doing so, no extra burden is imposed on the servers.

We refer to the paths from each server to the client and the paths from the client to the servers as forward and backward paths, respectively. Experimental studies on the Internet video traffic [7, 99] show that the packet loss events occur in bursts. To capture this temporal dependency between the packet losses, we adopt the well-known two-state Markovian loss model [98, 99], where states NC and C represent the non-congested and congested states, respectively. The delay that a packet experiences is also correlated to the congestion level on the path. The more congested the network becomes, the longer the delays experienced by the packets. Illustrative sketches of the delay characteristics for states NC and C of

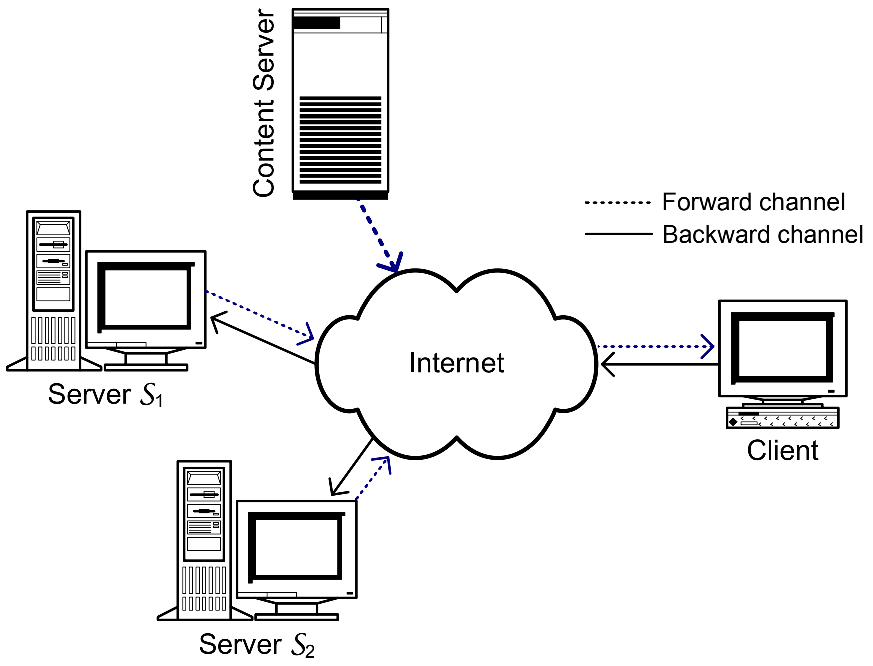


Figure 27: An illustration of a multi-server VoD streaming system.

the forward path are shown in Figure 28 [7]. Since a loss event might be perceived as the packet is delayed infinitely, it is convenient to indicate the packet loss rates, denoted by ϵ_{NC} and ϵ_{C}, as impulses at the infinity on the density functions. For the sake of the simplicity, we assume that the path does not change its state once the transmission starts. Hence, it follows that the cumulative distribution of the forward-trip time (FTT) for any path state can be given by

$$P\{\mathrm{FTT} \leq \tau\} = \int_0^\tau \left(1 - \epsilon_{\{\mathrm{NC,C}\}}\right) p_{\{\mathrm{NC,C}\}}(t)dt, \tag{20}$$

and its complement can be given by

$$P\{\mathrm{FTT} > \tau\} = \epsilon_{\{\mathrm{NC,C}\}} + \int_\tau^\infty \left(1 - \epsilon_{\{\mathrm{NC,C}\}}\right) p_{\{\mathrm{NC,C}\}}(t)dt. \tag{21}$$

Similar expressions can also be derived for the backward-trip time (BTT), and the distribution of the round-trip time (RTT) can be computed by convolving the distributions of FTT and BTT. Throughout our presentation, we use k as a superscript to indicate the path parameters associated with server $\mathcal{S}_k$. Although there have been studies that modeled the packet delay distributions [7, 101], we will not assume a specific distribution during our analysis. However, we will use some of these models in our simulations.

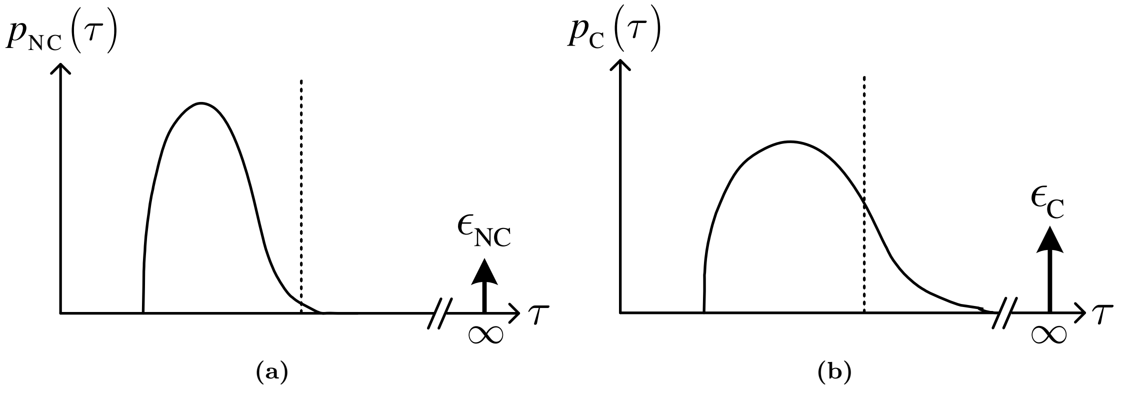


Figure 28: Delay distributions for states NC (a) and C (b) of the forward path.

We start our analysis by introducing three properties associated with each video packet. For video packet l, we denote the packet size (in bytes), importance and decoding deadline by B_l, Δd_l and $t_{D,l}$, respectively. Δd_l is a measure of the amount by which the distortion decreases when the client decodes packet l by its decoding deadline. Generally, in a client-driven system, B_l and Δd_l are unknown to the client. Nonetheless, the servers can convey this information to the client by using the Session Initiation Protocol (SIP) [118] before the streaming session starts.

Before we get into the details, let us first define the terms of rate and distortion. *Rate* is the expected cost of streaming the entire video, which equals the sum of the costs of all video packets transmitted[2]. *Distortion* refers to the expected distortion incurred in the entire video stream. As mentioned above, this distortion decreases by Δd_l when video packet l is decoded by its decoding deadline.

Consider a single video packet. Under the request policy π, the cost per byte for this packet is denoted by $\rho(\pi)$, and $\epsilon(\pi)$ is the measure of the distortion incurred if this packet cannot be decoded on time. Our goal is to identify the request policy that minimizes the Lagrangian $\epsilon(\pi) + \lambda\rho(\pi)$. In this minimization, by adjusting λ, we find the request policy that achieves the minimum distortion for a given ρ. In practice, the rate is generally given as a constraint for each server. Hence, we seek the particular request policy for each

[2]There is also a cost associated with sending the request packets. Although this cost does not directly count towards the streaming rate, we have to ensure that it conforms with the rate constraints, if any, on the backward paths.

server under the set of such constraints. Before focusing on streaming the entire video, we first study the transmission of a single packet to understand the dynamics involved in our problem.

4.2.2 Scheduling Algorithm for a Single Video Packet

Let $t_0, t_1, ..., t_{N-1}$ be N discrete request opportunities at which the client can send request packet(s), and let t_D be the decoding deadline for the target video packet. Suppose that $a_0^k, a_1^k, ..., a_{N-1}^k$ represents a request pattern, where $a_i^k = 1$ if a request packet is sent to server $\mathcal{S}_k$ at time t_i, and $a_i^k = 0$ otherwise. Any set of a_i^k defines a *request policy.* Since this is a client-driven system, the server (re)transmits the same video packet as a result of each successfully-received request. Hence, assuming that the client cancels any further request upon receiving the video packet, for any request policy π, the expected cost (per byte) for server $\mathcal{S}_k$ is given by

$$\rho^k = \sum_{i:a_i^k=1} \phi(i) \times P\left\{\mathrm{BTT}^k < \infty\right\}, \tag{22}$$

where $\phi(i)$ is the probability that the requested video packet has not been received by time t_i. One can compute $\phi(i)$ from

$$\phi(i) = \prod_{j \leq i} \prod_{k:a_j^k=1} P\left\{\mathrm{RTT}^k > t_i - t_j\right\}. \tag{23}$$

Subsequently, the probability of not receiving packet l before its decoding deadline equals

$$\theta = \phi\left(i : t_i = t_D\right). \tag{24}$$

Finally, the expected distortion can be expressed as

$$\epsilon = \theta \times \Delta d. \tag{25}$$

It is straightforward to compute (22) and (25) for any request policy π, and select the request policy that minimizes (25) for the given constraints on ρ^k. However, for large N, evaluating the expected cost and distortion of every possible policy can be intractable. Hence, at this point we use a Markov decision process (MDP) framework [119]. In Figure 29, we show the decision tree for the MDP used in our problem. On this tree, a request policy

(π) represents the set of actions taken at each state. At the initial state (q_0), the client has four possible action choices, *e.g.*, requesting or not requesting from either of the two servers. We denote the actions taken by the client at time t_i by $\boldsymbol{a}_i = [a_i^1\ a_i^2]$. Just before taking a new action at the next request opportunity, say t_{i+1}, the client may or may not have received the requested packet. This observation is represented by o_i. $o_i = 1$ indicates the reception of the video packet, and consequently, the process enters a final state. In contrast, if the video packet is not received by time t_{i+1}, $o_i = 0$ in this case, the client again chooses one of the four possible actions.

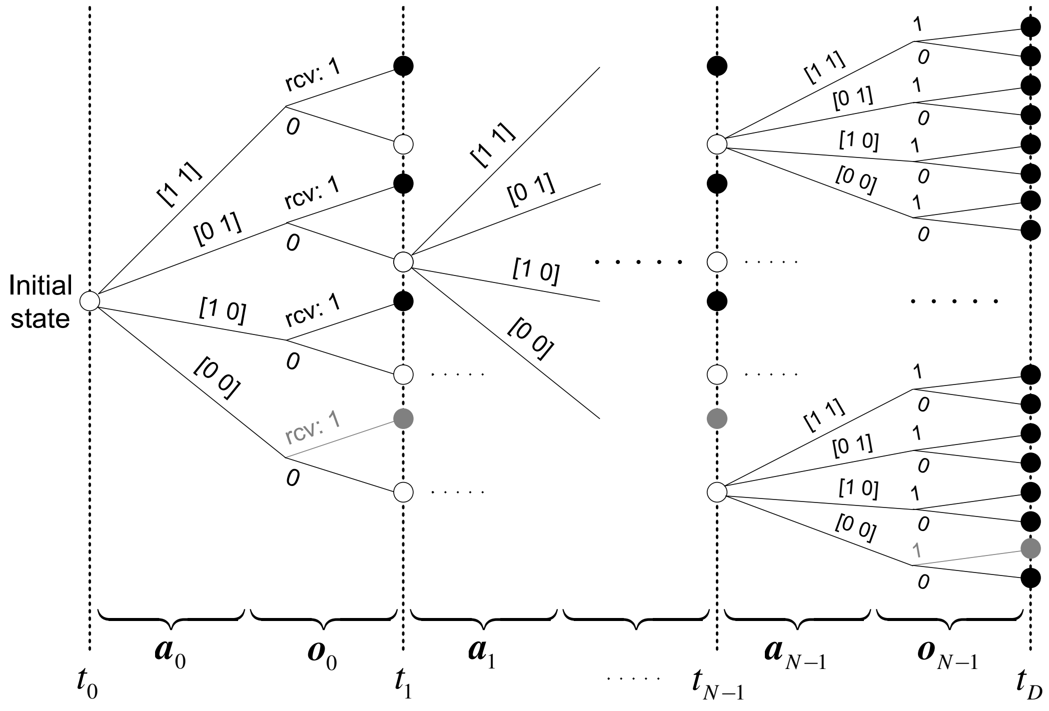


Figure 29: Decision tree for the MDP for two-server streaming system. Final states are indicated with filled circles. Infeasible actions, observations and states are gray-colored.

Let us denote the state at time t_i by q_i. The sequence of the visited states under a request policy forms a path in the tree, and state q_{i+1} is totally characterized by the actions and observations up to time t_{i+1}. Consequently, we can give the state transition probabilities as

$$P\left(q_{i+1}|q_i, \boldsymbol{a}_i\right) = \begin{cases} \psi, & \text{if } o_i = 0; \\ 1-\psi, & \text{if } o_i = 1, \end{cases} \tag{26}$$

where ψ is the probability that no video packet will be received by time t_{i+1} given that no video packet has arrived by time t_i. Hence, we have

$$\psi = \prod_{j \leq i} \prod_{k:a_j^k=1} P\left\{\text{RTT}^k > t_{i+1} - t_j | \text{RTT}^k > t_i - t_j\right\}. \tag{27}$$

Indeed, following the request policy π forms a Markov chain whose state transition probabilities are given by

$$P_\pi\left(q_{i+1}|q_i\right) \equiv P\left(q_{i+1}|q_i, \boldsymbol{a}_i\right). \tag{28}$$

There can be several feasible outcomes of this particular request policy through the Markov chain. Let $\boldsymbol{q} = (q_0, q_1, ..., q_{F-1})$ be one of the feasible paths with length F. Note that, we have $o_i = 0$ for $i < F-1$ on $\boldsymbol{q}$. By the Markovian property, it is straightforward to show

$$P_\pi(\boldsymbol{q}) = \prod_{i=0}^{F-1} P_\pi\left(q_{i+1}|q_i\right). \tag{29}$$

The corresponding cost of transmitting the video packets for server $\mathcal{S}_k$ on path $\boldsymbol{q}$ is given by

$$\rho_\pi^k(\boldsymbol{q}) = \sum_{i:a_i^k=1}^{F-1} P\left\{\text{BTT}^k < \infty\right\}. \tag{30}$$

Finally, if the video packet is received on path $\boldsymbol{q}$, *i.e.*, if $o_{F-1} = 1$, the distortion becomes zero. Otherwise, the distortion is equal to Δd. That is,

$$\epsilon_\pi(\boldsymbol{q}) = \begin{cases} 0, & \text{if } o_{F-1} = 1; \\ \Delta d, & \text{ow.} \end{cases} \tag{31}$$

Given $\rho_\pi^k(\boldsymbol{q})$ and $\epsilon_\pi(\boldsymbol{q})$, we can compute the expected cost (per byte) and distortion over all realizations of $\boldsymbol{q}$ for request policy π as follows:

$$\rho^k(\pi) \equiv \sum_{\boldsymbol{q}} P_\pi(\boldsymbol{q})\rho_\pi^k(\boldsymbol{q}) \quad \text{and} \quad \epsilon(\pi) \equiv \sum_{\boldsymbol{q}} P_\pi(\boldsymbol{q})\,\epsilon_\pi(\boldsymbol{q}). \tag{32}$$

One can compute $\rho^k(\pi)$ and $\epsilon(\pi)$ for any request policy by using (32). However, it would not be feasible to enumerate the pairs of $\left\{\rho^k(\pi), \epsilon(\pi)\right\}$ for all policies and select the one that minimizes the Lagrangian $j(\pi) = \epsilon(\pi) + \sum_k \lambda^k \rho^k(\pi)$. Instead, we can express $j(\pi)$ as

$$j(\pi) = \sum_{\boldsymbol{q}} P_\pi(\boldsymbol{q}) j_\pi(\boldsymbol{q}), \tag{33}$$

where $j_\pi(\boldsymbol{q}) \equiv \epsilon_\pi(\boldsymbol{q}) + \sum_k \lambda^k \rho_\pi^k(\boldsymbol{q})$, and minimize it with dynamic programming. To do so, we need to define $j_\pi(\boldsymbol{q})$ for the incomplete paths as well. The expected Lagrangian for all paths through q_i can be written as

$$j_\pi(q_i) = \begin{cases} \epsilon_\pi(\boldsymbol{q}) + \sum_k \lambda^k \rho_\pi^k(\boldsymbol{q}), & \text{if } q_i \text{ is final in } \boldsymbol{q}, \textit{ i.e.}, i = F; \\ \sum_{q_{i+1}} P\left(q_{i+1}|q_i, \boldsymbol{a}_i\right) j_\pi(q_{i+1}), & \text{ow.} \end{cases} \tag{34}$$

By induction, one can show that $j^*(q_i) \leq j_\pi(q_i)$ for all q_i and π, where

$$\pi^*(q_i) = \arg\min_{\boldsymbol{a}} \sum_{q_{i+1}} P\left(q_{i+1}|q_i, \boldsymbol{a}\right) j^*(q_{i+1}), \tag{35}$$

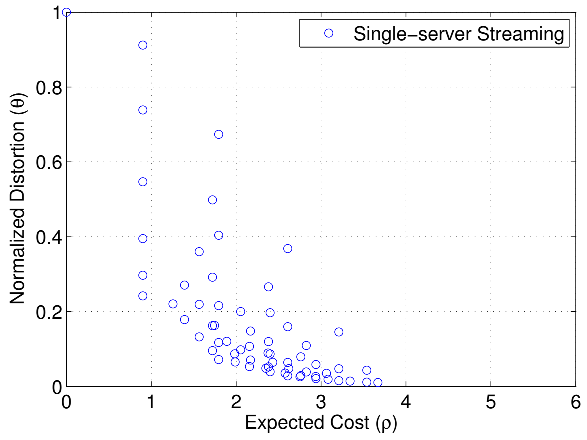
for all non-final states q_i. With (34) and (35), it is straightforward to minimize (33) under the rate constraint given for each server.

To compare the rate-distortion performance of the single and two-server streaming systems, we plot the normalized distortion, *i.e.*, θ, against the expected cost in Figure 30 for every possible request policy that can be adopted for a single video packet. In producing these plots, we used 100 ms, 100 ms and 350 ms for the mean FTT, mean BTT and playout delay, respectively. The client had six request opportunities at every 50 ms. The packet loss rate was set to 10% for each path. When we examine the achievable points on the rate-distortion plots in Figure 30, we see that the convex hull for the two-server streaming system is closer to the origin, and the two-server system can attain smaller distortion at the same cost compared to the single-server streaming system. In other words, the two-server streaming system can achieve a higher streaming quality under the same rate constraints provided that the optimal request policy is selected for each server.

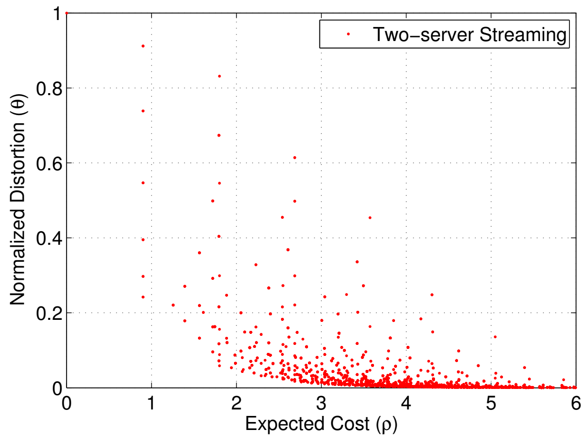
4.2.3 Scheduling Algorithm for a Group of Video Packets

In this section, we generalize the same idea to a group of packets. Let $\mathcal{G}$ denote this group, and assume that the client adopts a request policy π_l for packet $l \in \mathcal{G}$. It follows that the expected cost for server $\mathcal{S}_k$ is computed by

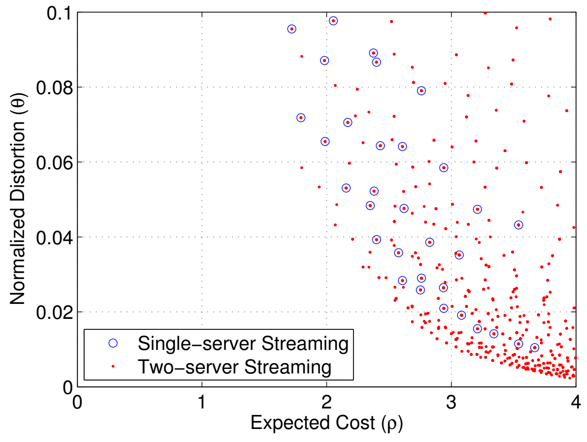
$$R^k(\boldsymbol{\pi}) = \sum_{l \in \mathcal{G}} B_l \rho^k(\pi_l), \tag{36}$$



(a)



(b)



(c)

Figure 30: Normalized rate-distortion plots for the single (a) and two-server streaming systems (b). The comparison is given in (c).

where $\boldsymbol{\pi} = (\pi_1, \pi_2, ..., \pi_L)$ represents the request policy vector for group $\mathcal{G}$. On the other hand, to express the expected distortion we have to consider the packet interdependencies. Recall from Section 1.1 that packet l can only be decoded successfully provided that all video packets to which packet l is referenced are already received and decoded. We use the notation $\mathcal{A}_l$ to indicate the ancestors of packet l. Hence, $(1-\theta(\pi_l))\prod_{l'\in\mathcal{A}_l}(1-\theta(\pi_{l'}))$ gives the probability of being decodable for packet l. If we deduct the distortions of all decodable packets from the sum of the distortions of all packets in group $\mathcal{G}$, denoted by D_0, we get the expected distortion for group $\mathcal{G}$. That is,

$$D(\boldsymbol{\pi}) = D_0 - \sum_{l\in\mathcal{G}} \Delta d_l \times \left(1-\theta(\pi_l)\right) \prod_{l'\in\mathcal{A}_l} \left(1-\theta(\pi_{l'})\right). \tag{37}$$

Having specified $R^k(\boldsymbol{\pi})$ and $D(\boldsymbol{\pi})$, we seek the optimal group request policy vector by minimizing the Lagrangian

$$J(\boldsymbol{\pi}) = D(\boldsymbol{\pi}) + \sum_k \lambda^k R^k(\boldsymbol{\pi}). \tag{38}$$

This minimization problem can be solved by using iterative techniques. Interested readers are referred to [52, 57] for further details.

4.2.4 Simulations and Results

We present 30-minute simulation results to demonstrate the performance bounds of the single and two-server streaming systems. We used two standard test sequences Table Tennis (352×240) and Flower Garden (352×240) in our simulations. These sequences were encoded with a standard MPEG-2 encoder (TM5 [85]), and a GOP structure consisting of an I-frame and nine P-frames at 30 frames per second. During the simulations, the frames were displayed after an initial buffering of 500 ms to smooth out the delay jitter. The frames missing their display deadlines were concealed by the last successfully-displayed frame. However, the packets arriving after their decoding deadlines were still used to decode the subsequent predictively-coded frames. For comparison purposes, we provide our results in terms of the peak signal-to-noise ratio (PSNR) measure on the luminance (Y) channel. The PSNR value for each simulation was computed by averaging the PSNR values of 54000 individual frames.

Figure 31 shows that the achievable video quality improves for both systems as the rate increases. However, the two-server streaming system achieves up to 1.4 dB superior performance over the single-server streaming system. It is important to note that while the single-server streaming system has to suffer in the case of heavy congestion, the two-server streaming system can exploit server diversity and use the less-congested path to maintain a more stable quality. This way, the two-server streaming system is able to endure long-lasting congestions. This feature makes multi-server streaming more robust and a good choice particularly for the low-delay favoring applications.

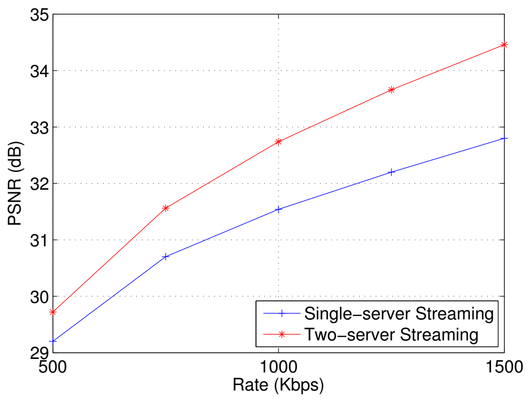
An important point here is that in producing the results given in Figure 31, we used two servers such that the resulting forward paths were totally link-disjoint, and the packet loss events observed on each path were uncorrelated. Not surprisingly, when we tried to use a set of servers that shared the same bottleneck link on their forward paths, multi-server streaming lost its edge and performed barely better than single-server streaming. In the next section, we study multiple description streaming in multi-server environments and demonstrate its benefits when the forward paths have a common bottleneck link.

4.3 Multi-Server Multiple Description Video Streaming

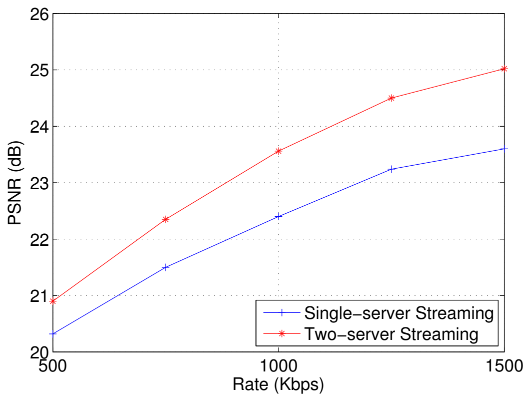
In the previous section, we introduced the concept of multi-server streaming and showed its advantages over single-server streaming by simulations. In the theoretical analysis, we have not limited ourselves to any particular video-encoding scheme, we merely assumed a general dependency structure among the encoded video packets. In this section, we illustrate how a special video-encoding scheme can be incorporated into the rate-distortion optimization. In particular, we consider rate-distortion optimized multiple description video streaming where each description is streamed from a different server. We start our discussion with a brief introduction to the error-resiliency features of multiple description streaming in multi-server environments.

4.3.1 System Overview

The self-reliance of the descriptions provides multiple description (MD) streaming highly-efficient error-resiliency features. The descriptions deliver a basic quality of video when they



(a)



(b)

Figure 31: Comparison of the single and two-server streaming systems for the Table Tennis (a) and Flower Garden (b) sequences when the forward paths are totally link-disjoint.

are individually decoded, and each additional description further refines the quality. Hence, as long as packet losses do not occur simultaneously in multiple descriptions, the client is guaranteed a continuous video. A conceptually-straightforward way to reduce the chance of concurrent description losses is to stream the descriptions over diverse paths. With the path diversity approach, MD streaming has been shown to be effective in combatting bursty packet losses and the subsequent error propagation among the video frames [43, 75]. However, unless the underlying routing protocol supports some sort of source-routing functionality, achieving path diversity over a conventional network such as the Internet requires an additional infrastructure for transmitting the descriptions over diverse routes[3].

[3]See Chapter 2 for further details on MD coding and MD streaming with path diversity.

A more practical end-to-end approach to imitate path diversity without requiring the physical network support is to stream the video from different servers. This approach is referred to as server diversity. As long as the servers are not co-located, streaming from different servers can offer the advantages of path diversity to the clients. With this motivation, Apostolopoulos *et al.* studied MD streaming from multiple servers within a CDN [76]. The authors investigated the performance of MD streaming for different CDN topologies and compared it with the performance of single description streaming. Besides proving the superior performance of MD streaming, their results also showed that the streaming quality could be improved as a larger number of servers were deployed inside the network. Intuitively, as more servers became available, the chance of finding diverse servers increased for the clients.

Clearly, streaming MD video from multiple servers is an effective approach. Nonetheless, the performance of this approach can be further improved by transmitting the video packets in a time-sensitive and network-adaptive manner. For this reason, we propose a client-driven packet scheduling algorithm, which is particularly designed for MD streaming in multi-server environments. The primary goal of this algorithm is to maximize the streaming quality by jointly considering the timeliness requirements of the application, dependency structure of the streamed video, network conditions and error-resiliency features of MD coding.

Suppose that the source video is encoded offline by an MD encoder to produce two descriptions. The descriptions are packetized into video packets. The packets corresponding to the first (denoted by l^1) and second descriptions (denoted by l^2) are served by the servers $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively, as depicted in Figure 32. From a networking point of view, there exists a limit on the transmission rate to keep the consumed bandwidth at a desired level. For a set of given rate constraints, the intriguing problem is to find the optimal request policy for each server such that the expected quality rendered at the client is maximized.

We develop an expression to evaluate the expected distortion that will be incurred for a given request policy. This expression takes into account the packet decoding deadlines and interdependencies as well as the unique error-resiliency features of the MD encoded video. For example, if the client receives one of the descriptions corresponding to a video frame,

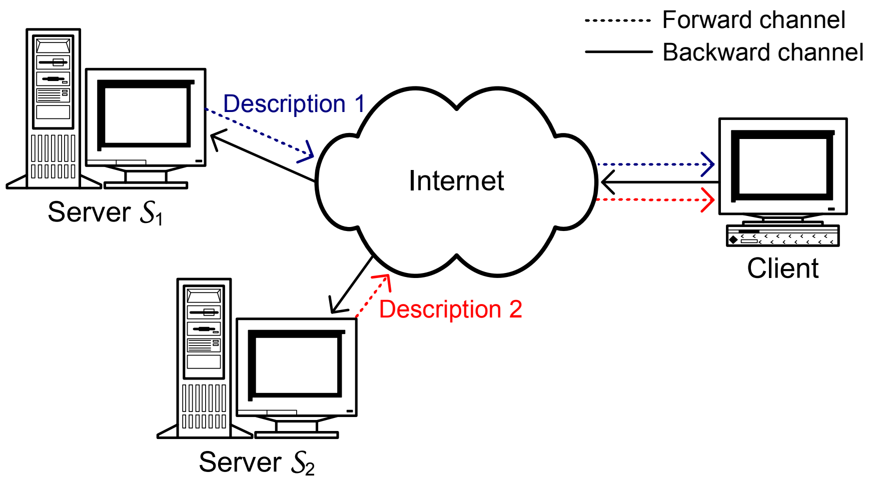


Figure 32: An illustration of a two-server multiple description video streaming system.

the client's tendency to request the other description reduces since its contribution to the video quality would be less. Instead, the client might prefer requesting a description that belongs to another video frame.

In the rest of the section, following a similar analysis presented in Section 4.2, we first solve the problem for a single set of descriptions, and then, formulate the case of multiple sets of descriptions.

4.3.2 Scheduling Algorithm for a Single Set of Descriptions

Let $t_0, t_1, ..., t_{N-1}$ be N discrete request opportunities at which the client can send request packet(s), and let $\pi = (\boldsymbol{a}_0, \boldsymbol{a}_1, ..., \boldsymbol{a}_{N-1})$ represent a request pattern, where $a_i^k = 1$ if a request is sent to server $\mathcal{S}_k$ asking the k^{th} description at time t_i, and $a_i^k = 0$, otherwise. Suppose that the client adopts the request policy π. Since any further request for a description will be cancelled upon its reception, the expected cost (per byte) for server $\mathcal{S}_k$ is given by

$$\rho^k = \sum_{i:a_i^k=1} \phi(i,k) \times P\left\{\mathrm{BTT}^k < \infty\right\}, \tag{39}$$

where $\phi(i,k)$ is the probability that the k^{th} description has not been received by time t_i, and it is given by

$$\phi(i,k) = \prod_{j:j\leq i;a_j^k=1} P\left\{\mathrm{RTT}^k > t_i - t_j\right\}. \tag{40}$$

The expected distortion rendered at the client can be computed by considering all possible cases of the description on-time arrivals. Following a similar approach presented in Section 2.4, we can write the expected distortion at the client as

$$\epsilon = \theta^1\theta^2 D_{0,0} + \theta^1\left(1-\theta^2\right)D_{0,1} + \left(1-\theta^1\right)\theta^2 D_{1,0} + \left(1-\theta^1\right)\left(1-\theta^2\right)D_{1,1}, \tag{41}$$

where θ^k is the probability of not receiving the k^{th} description before its decoding deadline, which is denoted by t_D^k. θ^k is given by

$$\theta^k = \prod_{i:a_i^k=1} P\left\{\mathrm{RTT}^k > t_D^k - t_i\right\}. \tag{42}$$

The values of the distortion terms, $D_{1,1}$, $D_{1,0}$, $D_{0,1}$ and $D_{0,0}$, depend on the specific MD codec used to produce the descriptions. In the sequel, we adopt the time-domain partitioning method with two descriptions, and borrow the distortion terms derived in Chapter 2.

To evaluate the expected cost and distortion of any request policy, we use a Markov decision process (MDP) framework similar to the one discussed in Section 4.2.2. The corresponding decision tree is built as shown in Figure 33. Starting with the initial state, the client has four possible action choices at every request opportunity. The actions taken by the client at time t_i are denoted by $\boldsymbol{a}_i = [a_i^1\ a_i^2]$. Before taking a new action at time t_{i+1}, the client may or may not have received the requested packet(s). We represent this observation by $\boldsymbol{o}_i$. $\boldsymbol{o}_i = [1\ 1]$ indicates the reception of both descriptions, and consequently, the process enters a final state. In contrast, if only one or none of the descriptions are received, the client again chooses an action.

Let q_i denote the state at time t_i. Noting that state q_{i+1} is totally characterized by the actions and observations up to time t_{i+1}, one can express the state transition probabilities as

$$P\left(q_{i+1}|q_i,\boldsymbol{a}_i\right) = \begin{cases} (1-\psi^1)\times(1-\psi^2), & \text{if } \boldsymbol{o}_i = [1\ 1]; \\ \psi^1\times(1-\psi^2), & \text{if } \boldsymbol{o}_i = [0\ 1]; \\ (1-\psi^1)\times\psi^2, & \text{if } \boldsymbol{o}_i = [1\ 0]; \\ \psi^1\times\psi^2, & \text{if } \boldsymbol{o}_i = [0\ 0], \end{cases} \tag{43}$$

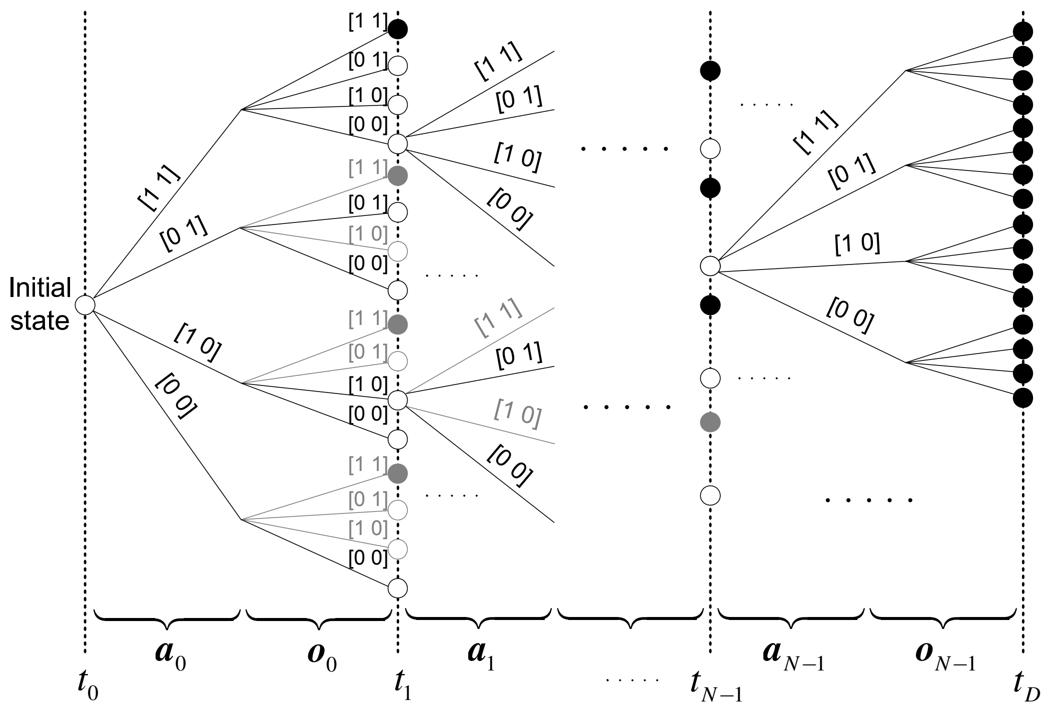


Figure 33: Decision tree for the MDP for two descriptions. Final states are indicated with filled circles. Infeasible actions, observations and states are gray-colored.

where ψ^k is the probability that the k^{th} description will not be received by time t_{i+1} given that it has not arrived by time t_i. That is,

$$\psi^k = \prod_{j:j\leq i;a_j^k=1} P\left\{\mathrm{RTT}^k > t_{i+1} - t_j | \mathrm{RTT}^k > t_i - t_j\right\}. \tag{44}$$

Recall that following a particular request policy π forms a Markov chain whose state transition probabilities are given by

$$P_\pi\left(q_{i+1}|q_i\right) \equiv P\left(q_{i+1}|q_i, \boldsymbol{a}_i\right). \tag{45}$$

Now, let $\boldsymbol{q} = (q_0, q_1, ..., q_{F-1})$ be one of the feasible paths, concluding at a final state, with length F through this Markov chain. By the Markovian property, it is straightforward to show that the probability of traversing this path equals

$$P_\pi(\boldsymbol{q}) = \prod_{i=0}^{F-1} P_\pi\left(q_{i+1}|q_i\right). \tag{46}$$

The cost for server $\mathcal{S}_k$ and the distortion associated with $\boldsymbol{q}$ are given by

$$\rho_\pi^k(\boldsymbol{q}) = \sum_{i:a_i^k=1}^{F-1} P\left\{\mathrm{BTT}^k < \infty\right\} \tag{47}$$

and

$$\epsilon_\pi(\boldsymbol{q}) = \begin{cases} D_{1,1}, & \text{if } \boldsymbol{o}_{F-1} = [1\ 1]; \\ D_{0,1}, & \text{if } \boldsymbol{o}_{F-1} = [0\ 1]; \\ D_{1,0}, & \text{if } \boldsymbol{o}_{F-1} = [1\ 0]; \\ D_{0,0}, & \text{if } \boldsymbol{o}_{F-1} = [0\ 0], \end{cases} \tag{48}$$

respectively. Given $\rho_\pi^k(\boldsymbol{q})$ and $\epsilon_\pi(\boldsymbol{q})$, the expected cost (per byte) and distortion are computed over all realizations of $\boldsymbol{q}$ for request policy π as follows:

$$\rho^k(\pi) \equiv \sum_{\boldsymbol{q}} P_\pi(\boldsymbol{q})\, \rho_\pi^k(\boldsymbol{q}) \quad \text{and} \quad \epsilon(\pi) \equiv \sum_{\boldsymbol{q}} P_\pi(\boldsymbol{q})\, \epsilon_\pi(\boldsymbol{q}). \tag{49}$$

With $\rho^k(\pi)$ and $\epsilon(\pi)$, we form the Lagrangian $j(\pi) = \epsilon(\pi) + \sum_k \lambda^k \rho^k(\pi)$ and seek the optimal request policy by minimizing $j(\pi)$ for the given rate constraints.

4.3.3 Scheduling Algorithm for Multiple Sets of Descriptions

Let us consider a group of frames, denoted by $\mathcal{G}$, to be scheduled by the client. Suppose that the client adopts a request policy π_l for the description set $l \in \mathcal{G}$, and let $\boldsymbol{\pi} = (\pi_1, \pi_2, ..., \pi_L)$ represent the group request policy for group $\mathcal{G}$. Following a similar analysis presented in 4.2.3, we compute the expected cost for server $\mathcal{S}_k$ from

$$R^k(\boldsymbol{\pi}) = \sum_{l \in \mathcal{G}} B_l^k \rho^k(\pi_l), \tag{50}$$

and the expected distortion from

$$\begin{aligned} D(\boldsymbol{\pi}) \ = \ & \sum_{l \in \mathcal{G}} P_d(l^1) P_d(l^2) D_{1,1}(l) \\ & + \sum_{l \in \mathcal{G}} \left(1 - P_d(l^1)\right) P_d(l^2) D_{0,1}(l) \\ & + \sum_{l \in \mathcal{G}} P_d(l^1) \left(1 - P_d(l^2)\right) D_{1,0}(l) \\ & + \sum_{l \in \mathcal{G}} \left(1 - P_d(l^1)\right) \left(1 - P_d(l^2)\right) D_{0,0}(l), \end{aligned} \tag{51}$$

where $P_d(l^k)$ represents the decodability probability of description l^k. This probability is given by

$$P_d(l^k) = \left(1 - \theta^k(l)\right) \prod_{l' \in \mathcal{A}_l} \left(1 - \prod_k \theta^k(l')\right). \tag{52}$$

Finally, the optimal group request policy is found by minimizing the Lagrangian

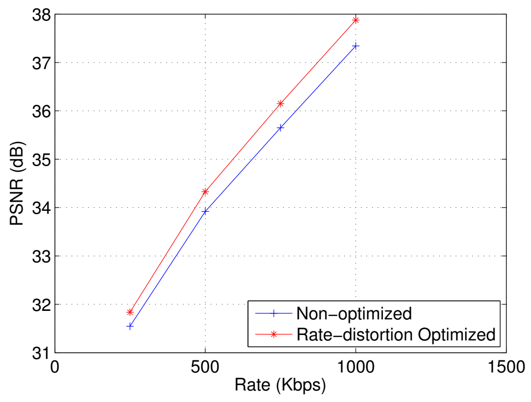
$$J(\boldsymbol{\pi}) = D(\boldsymbol{\pi}) + \sum_{k} \lambda^k R^k(\boldsymbol{\pi}). \tag{53}$$

4.3.4 Simulations and Results

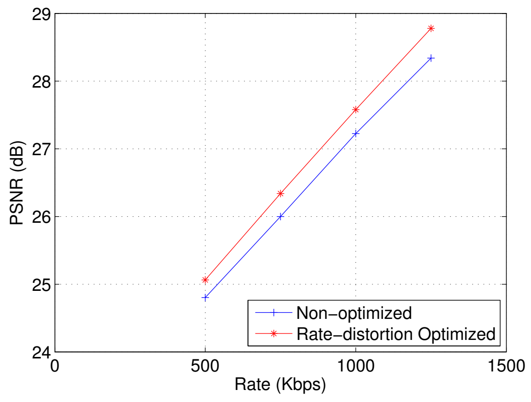
In this section, we present simulation results to evaluate the performance of the rate-distortion optimized packet scheduling algorithm. We compare optimized MD streaming with non-optimized MD streaming, where the descriptions on different servers are requested sequentially. That is, at each request opportunity the client requests the subsequent packet for the corresponding description from each server. To this end, we conducted our simulations on two different setups: (*i*) when the forward paths were totally link-disjoint, hence, the packet loss events for different descriptions were uncorrelated, and (*ii*) when the forward paths shared the point-of-congestion (PoC), hence, the packet loss events were correlated. This analysis was important for understanding the potential impact of a shared congestion on the performance of multi-server streaming.

We used two standard test sequences FOREMAN (352×288) and FOOTBALL (352×240) in our simulations. The descriptions were produced by the time-domain partitioning method with a standard H.264 encoder [120], and a GOP structure consisting of an I-frame and nine P-frames at 30 frames per second. The rest of the simulation settings were adopted from Section 4.2.4 except that the interval between the request opportunities was set to 33 ms for the simulations discussed below.

First, consider the PSNR plots given in Figure 34. Naturally, the video quality improves as the rate increases. However, the rate-distortion optimized MD streaming scheme achieves up to 0.53 and 0.44 dB superior performance compared to non-optimized MD streaming for the FOREMAN and FOOTBALL sequences, respectively. When we examine the packet traces, we deduce that the PSNR gain largely stems from the fact that the rate-distortion optimized packet scheduling algorithm does not request the descriptions belonging to the same video frame at the same opportunity, rather it requests the ones belonging to different video frames. This helps the client maintain a good video quality even in the event of a concurrent description loss.



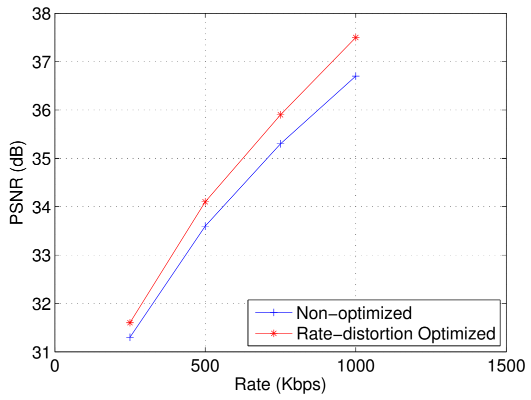
(a)



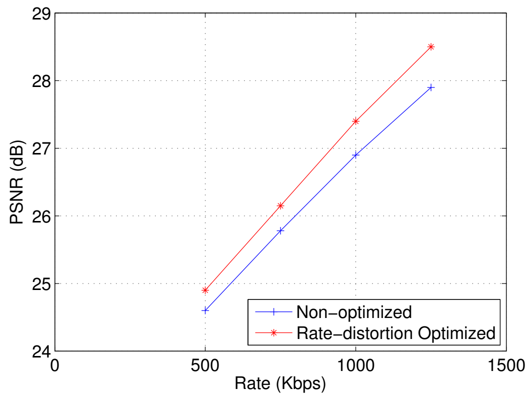
(b)

Figure 34: Comparison of the non-optimized and rate-distortion optimized MD streaming systems for the FOREMAN (a) and FOOTBALL (b) sequences when the forward paths do not share the PoC.

Next, consider the plots in Figure 35 that correspond to the case when the forward paths share the PoC. As shown in [43] and in Chapter 2, MD streaming becomes less error-resilient when the description losses become more correlated. That is, if both descriptions are simultaneously lost, the video quality degrades because of the poor picture reconstruction and the subsequent error propagation. This is also clearly seen when the plots in Figures 34 and 35 are compared. However, it is important that the performance gap between the optimized and non-optimized streaming schemes increased in Figure 35. Specifically, rate-distortion optimized MD streaming incurs a relatively smaller performance degradation (ranging from 0.2 to 0.3 dB) compared to non-optimized MD streaming (ranging from 0.3 to 0.6 dB). The reason is two-fold: First, as discussed above, rate-distortion optimization refrains from requesting correlated descriptions at the same opportunity. This naturally reduces the probability of simultaneous description losses. Second, when a description belonging to an important frame is inferred to be lost, the rate-distortion optimized packet scheduling algorithm requests it again instead of requesting a description that belongs to a less important frame. This way, the number of decodable frames is still kept as large as possible in spite of the shared PoC. On the other hand, the non-optimized MD streaming scheme is bound to lose correlated descriptions every time a congestion occurs at the PoC. This inevitably causes a larger quality degradation at the client.



(a)



(b)

Figure 35: Comparison of the non-optimized and rate-distortion optimized MD streaming systems for the FOREMAN (a) and FOOTBALL (b) sequences when the forward paths share the PoC.

4.4 Conclusions

In this chapter, we presented a generalized packet scheduling algorithm for streaming on-demand video from multiple servers. This algorithm achieved rate-distortion optimal streaming at the client by evaluating the video characteristics, network conditions and application requirements. Our simulations showed that the clients experienced a more continuous and higher streaming quality by exploiting the error-resiliency features of server diversity. We also demonstrated how multiple description coding and multi-server streaming could be combined together and used to deliver a reliable video even when the network paths leading to the client had their bottleneck links in common.

The algorithms developed in this chapter can also be adapted for the cases where the client can connect to a single server over multiple networks. For example, a multi-homed client may have two or more network interfaces connected to different networks in order to increase its bandwidth and improve the connection reliability. By exploiting multiple simultaneous connections (each through a different network interface) to a single video server, this multi-homed client may enjoy the benefits of multi-server streaming and enhance its video-on-demand experience.

CHAPTER V

IN-NETWORK SOLUTIONS FOR INTERACTIVE VIDEO SERVICES

While emerging broadband access technologies such as DSL and cable are making multimedia services feasible and economically attractive for end-users, there still exist several hurdles in terms of service sustainability and reliability. Unfortunately, without the desired QoS support, tackling these hurdles with traditional solutions is an insuperably difficult task. Yet, novel designs that are proven to be useful in various scenarios may easily fail when the underlying network experiences severe packet loss or delay. Such circumstances are unavoidable in today's best-effort Internet and will likely prevail in the near future as well. A promising approach in satisfying the stringent requirements of delay-intolerant video applications is to benefit from configurable proxies. In this study, we introduce a versatile proxy-based solution to enhance the performance of such applications running over networks with large delays. We first propose a methodology that accurately identifies lost packets in real time. This methodology is then used by the proxy and end-users to improve the error-control/protection capability of the video applications. By Internet experiments between the U.S. and Europe, we demonstrate the effectiveness and potential benefits of the proposed approach.

5.1 *Introduction*[†]

The primary role of ubiquitous networking, in particular of the Internet, is to disseminate information in a timely manner and provide an inexpensive communication platform to its users. As the access technologies provide high bandwidths at economically-attractive prices and by the help of the advances in audiovisual signal processing, a larger number of people are communicating interactively through networks every day. However, despite all the advances, these technologies have so far not been able to provide the desired reliability. The

[†]Parts of this chapter were previously published in [121–125].

reason behind this shortfall is that the service requirements of the emerging conversational interactive applications differ significantly from those of the conventional data-oriented applications. Recent studies focused on designing new solutions that were specifically tailored for the unique features of real-time media. However, without the essential QoS provisioning, even these media-aware approaches may struggle, if the network becomes physically incapable or performs poorly for a prolonged amount of time. In other words, clever design has its limitations. Such circumstances are unavoidable in today's best-effort Internet and will likely prevail in the near future as well.

A particularly useful method to improve the reliability of delay-intolerant video applications is to utilize proxies. Depending on their types and functionalities, proxies can be used in different contexts. In this work, we investigate the potential benefits of proxies in an interactive application for which packet delivery deadlines are stringent. In this class of applications, dealing with large delay variations for the end-users that are geographically distant from each other is arduous. Large round-trip delays hinder feedback-based error-control/protection methods since the validity of the feedback messages are mostly depreciated, if not totally useless, by the time they are received. As a result, end-users can be ineffective in taking the necessary actions against missing packets. In addition, due to the late feedback reports, end-users will also fail to adapt to the network conditions in a timely manner, if the network conditions change rapidly.

The network delay between two end-users is determined by the underlying IP routing mechanisms, over which the end-users do not have any control. If the Internet path between two end-users has many hops, the end-to-end delay as well as the delay jitter will likely be large on this path. Consequently, conversational applications may incur interruptions and perform poorly. A practical way to alleviate these problems is to divide the large network between the end-users into two or more smaller sub-networks by introducing *intermediate proxies* (I-Proxy), as sketched in Figure 36. This partitioning provides two main advantages: First, each sub-network now includes fewer hops, and hence, has a smaller end-to-end delay and jitter. This enables the end-users to receive more recent and accurate feedback about the network conditions, and perform better error control and protection. Second, we can benefit from the error-control capabilities of the I-Proxy. Depending on the network configuration, the potential improvement in the end-to-end video quality can be substantial.

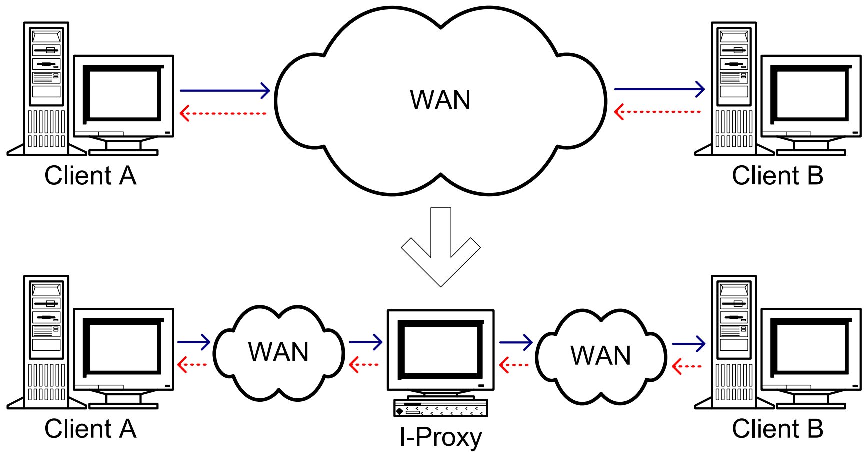


Figure 36: Illustration of the I-Proxy approach.

Proxies have been around since the early days of the Internet. The main purpose behind their deployment has been to cache popular files, data or multimedia, at locations close to the end-users, and reduce the load on the network and content servers. In this context, several content caching and distribution, server scheduling, and proxy-placement algorithms have been proposed [126–132]. In their work [130], Hartanto *et al.* studied proxy-caching strategies for continuous media in interactive streaming applications. In particular, they developed caching strategies based on the request patterns for the proxies connected to the clients via LAN. In [131], the authors studied the problem of minimizing the bandwidth consumption by jointly optimizing server scheduling and caching strategies. All these studies utilized proxies for caching and reducing the network load. However, none of these studies considered using proxies for providing better error control for time-critical media content. On the other hand, some of the existing messenger services employ systems similar to the I-Proxy approach. Next, we summarize the similarities and differences[1].

Currently, there are several free messenger services offered by different service providers. Among the popular ones, MSN Messenger requires two clients to talk to each other over

[1]As these services are proprietary, the following discussion is based on the information collected from discussion groups and user experiences on the Internet. For more information on these services, visit http://messenger.msn.com, http://messenger.yahoo.com and http://www.skype.com.

a direct connection. In Yahoo Messenger, the clients connect to an audio server before making a call unless a direct connection can be established. Teleconferencing between multiple parties is also available, where a central audio server hosts the session. Likewise, Skype relays the calls between two clients over a super node, which is a Skype client with a public IP address, if the clients cannot connect to each other directly.

In terms of the system configuration, relaying packets over external nodes, *e.g.*, the central audio servers in Yahoo Messenger and the super nodes in Skype, has similarities with relaying packets over an I-Proxy. However, for Yahoo Messenger and Skype, the main motivation in relaying is to enable the voice service for the clients who reside behind non-UPnP-capable firewalls or NAT devices[2,3]. For the clients those are behind UPnP-capable devices, a direct connection can be established through the UDP hole punching technique [133], thus relaying is not required. In contrast, an I-Proxy offers not only a firewall-traversal solution but also customizable error-control capabilities that lead to superior audiovisual quality.

The rest of the chapter is organized as follows: In Section 5.2, we provide an overview of the I-Proxy approach. Sections 5.3 through 5.5 discuss the proposed methods and implementation issues in detail. Experimental results are presented in Section 5.6. Finally, we conclude the chapter in Section 5.7.

5.2 Intermediate-Proxy Approach

In this section, we provide two example cases to demonstrate the potential benefits of using proxies in interactive video applications. Because of its popularity, we choose videotelephony as our target application. However, it is straightforward to generalize the I-Proxy approach to other applications such as videoconferencing and distance learning.

Consider a videotelephony session between two end-users that are network-wise far away from each other, *e.g.*, one resides in U.S. and the other one resides in Europe. Both clients capture and encode video in real time. After packetizing the video, they transmit the

[2]UPnP: Universal plug and play. Refer to 3.1 for a detailed description.

[3]NAT: Network address translation, also known as network masquerading or IP-masquerading. A NAT device rewrites the source and/or destination addresses of IP packets as they pass through. The goal is to enable multiple hosts on a private network to access the Internet using a single public IP address.

video packets to an I-Proxy, and the I-Proxy forwards them to the other end. Meanwhile, the I-Proxy also caches any forwarded packet for a short amount of time. Since the round-trip delay between such a client pair averages a few hundred milliseconds, the error-control/protection capability is limited. For the sake of clarity, in the rest of the chapter, we consider video transmission in one direction, *i.e.*, from the server to the client.

5.2.1 Enabling Retransmission-Based Error-Control Methods

Automatic repeat request (ARQ) is a fundamental error-control method that has been widely used for reliable data transfer protocols such as TCP. Compared to forward error correction (FEC), ARQ has a simpler design and generally achieves higher throughput as long as the channel error rate or packet loss rate is not very high. On the other hand, ARQ-based error control can be impractical for networks where the extra delay introduced by the retransmissions is prohibitively large. However, by the help of an I-Proxy, ARQ may become a feasible approach for error recovery as the application runs over two small sub-networks rather a single large network.

In packet-switched IP networks, packets experience inevitably variable delay due to queuing, route changes, packet reordering, etc. To this effect, a client can never be sure, if a missing packet has been lost or merely delayed. Basically, a packet is assumed to be lost, if it is not received within an expected time. In that case, as shown in Figure 37, the client sends a retransmission request to the server. However, the I-Proxy intercepts this request and immediately retransmits the requested packet to the client, provided that the packet is available in the cache. The explicit advantage of this *early retransmission* is the savings in the recovery time for the missing packet; it avoids unnecessary delays that would be incurred in case of a retransmission by the server. Less explicitly, the I-Proxy also reduces the amount of network resources consumed during the retransmission. This approach is particularly useful when the packets often get lost between the I-Proxy and client, and the round-trip delay between the I-Proxy and client is relatively smaller. On the other hand, if the packet is not available in the cache, the retransmission request is conveyed to the server. The server can do an end-to-end retransmission, if sufficient time exists.

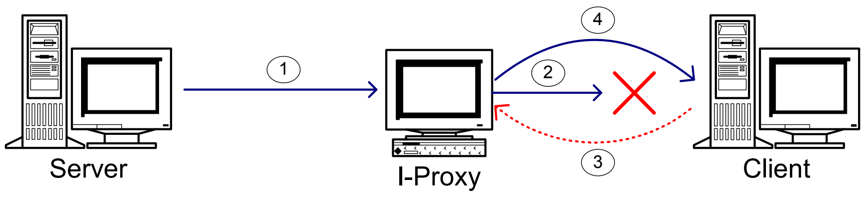


Figure 37: Early retransmission by the I-Proxy.

There might be some packets that get lost between the server and I-Proxy as well. Eventually, the client will observe these lost packets and request a retransmission. However, the I-Proxy can infer any missing packet earlier than the client and subsequently request a retransmission from the server. The server then retransmits the requested packet and the I-Proxy forwards it to the client. This proactive approach, called *fast retransmission*, avoids unnecessary waiting for the client to identify the missing packets (See Figure 38). In other words, by the help of the I-Proxy, the server is informed about the missing packets at an earlier stage, which allows us to take the necessary actions on time. Note that without an I-Proxy we would have to wait for the client to report the missing packet, and by that time, it might have been too late for a retransmission attempt because of the insufficient remaining time to the decoding deadline.

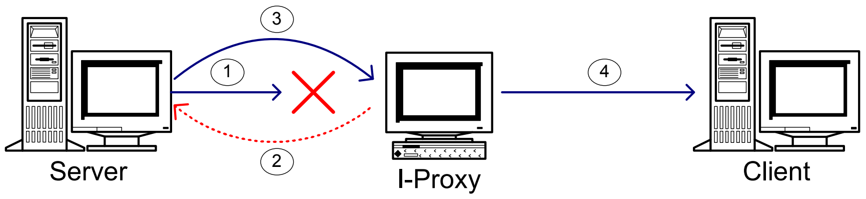


Figure 38: Fast retransmission by the I-Proxy.

It is worth emphasizing that the early/fast retransmission performed by the I-Proxy is a strategy between doing a retransmission on an end-to-end and a hop-by-hop basis. While the I-Proxy cannot deliver the advantages of the latter approach, *e.g.*, faster error recovery and minimal use of network bandwidth, the chances are it will recover the packets faster compared to the end-to-end retransmissions, provided that an intelligent proxy placement/selection is made.

5.2.2 Fast Intra-Frame Updates

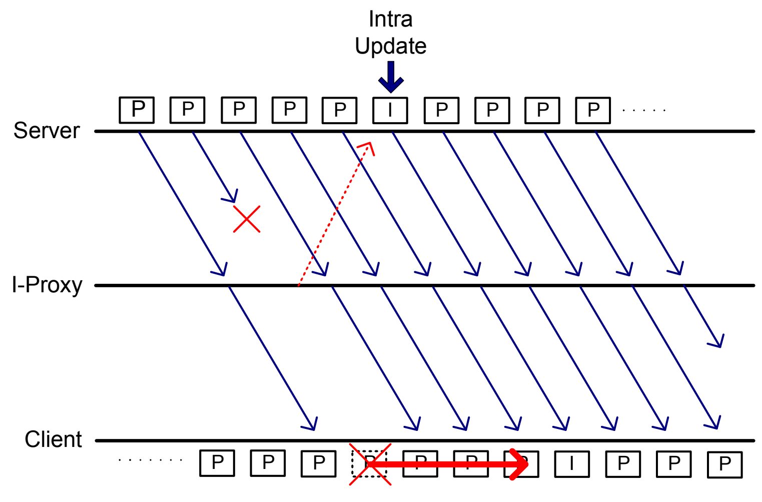
In the research community, there has been a great interest in developing error-resilient video coding techniques that are useful in reducing the impact of the Internet's imperfections. Although such techniques provide visual improvements, the reconstruction quality is generally limited because of the mismatches between the encoder and decoder states. A quick way to suppress the resulting temporal error propagation is to insert an intra-coded picture, which can be decoded independently of other pictures. This is particularly useful in video streams that are encoded without a strict GOP structure. Of course, a swifter insertion of an intra-coded picture upon the detection of a packet loss, terminates the error propagation at an earlier time. As discussed above, the I-Proxy can infer missing packets earlier than the client, and when it does, the server is warned to intra-code the next picture. In doing so, the delay between the detection of the packet loss and insertion of the intra-coded picture shortens, which consequently reduces the amount of error propagation. This is illustrated in Figure 39.

Note that some of the error-control/protection methods that can be employed by the video applications are not necessarily retransmission-based. However, as a common task, any of these methods should identify the lost packets as early as possible in order to improve its performance. In the next section, we investigate this issue and propose a method that accurately identifies the late/lost packets in real time. In two-way video applications, this method runs on the end-users and I-Proxy, as all of these entities function both as a server and client. However, to keep the discussion concrete and focused, we shall refer to one of the end-users as the server and the other one as the client.

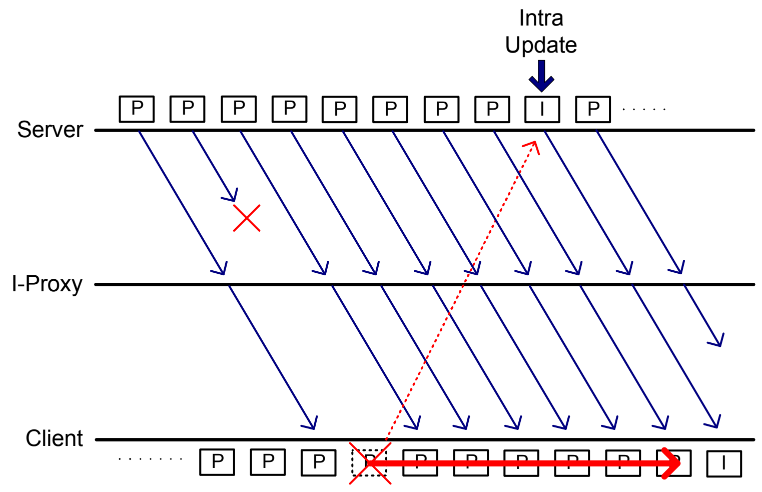
5.3 Detection of Lost Packets

5.3.1 Preliminaries

The Internet is a shared medium; any packet injected into the Internet has to wait for some time before it is serviced. It therefore experiences a random delay. Because of the finite buffering capabilities of the intermediate routers and switching devices, it is safe to assume that a packet is lost if it has not been received or acknowledged within some time after



(a)



(b)

Figure 39: Error propagation is suppressed at an earlier stage when the I-Proxy is enabled (a) compared to when it is disabled (b).

its transmission. In TCP jargon, this duration is referred to as the retransmission timeout (RTO). It is vital that the value of the RTO is chosen large enough so that the packets experiencing long queueing delays do not trigger spurious timeouts. However, adopting an arbitrarily large RTO is impractical for delay-sensitive multimedia applications. A delayed retransmission attempt eventually recovers a missing media packet. Yet, the chances are that the retransmitted packet will be late and useless for decoding at the client side. Therefore, an RTO estimation method that quickly detects lost packets is imperative for such applications. Only then can well-timed actions be taken for error control.

In low-delay video applications, it is a common practice to transmit the video frames as soon as they are packetized in order to avoid unnecessary delays. However, because of the efficient predictive-coding techniques that are used in popular video coding standards, encoded video frames vary in size and potentially produce a different number of packets. Furthermore, at high bitrates even the smallest video frame may not fit into one packet. If all the packets belonging to a single frame are transmitted back-to-back, the video traffic inevitably becomes bursty. Combined with the delay jitter experienced along the path, the varying nature of the transmission times causes the video packets arrive at the client at less-predictable times. In this section, we address this problem and propose a *burst-aware* RTO estimation method.

The transmission times of the video packets depend on several factors. Namely, the number of frames in a GOP, the GOP structure, encoding bitrate, video frame rate and IP packet size are the main parameters that vary the transmission times of the packets in a GOP[4]. As an illustrative example, consider Figure 40, where we denote the transmission time at the server of packet n by $t_T[n]$. Suppose that the GOP consists of one I-frame (producing seven IP packets) and nine P-frames (each producing two IP packets), and the frame rate is 20 frames per second. Let $\Delta t_T[n]$ represent the intertransmission time of packet n, which is defined as

$$\Delta t_T[n] = t_T[n] - t_T[n-1]. \tag{54}$$

The mean intertransmission time can be computed from

$$\Delta T = \frac{\text{GOP duration}}{\text{\# of packets in a GOP}}, \tag{55}$$

which is 20 ms in our example. However, as shown in Figure 40, Δt_T can be as small as a few milliseconds, and as large as 50 ms. It is clear that ΔT is largely inadequate to define the transmission regime at the server.

[4]Note that some video applications encode the video without a strict GOP structure. Instead, a small percentage of the macroblocks are intra-coded in every frame, producing almost equal-sized frames, and hence, less-bursty video traffic. In such cases, (55) can be computed over packet groups whose transmission regime exhibits a periodicity.

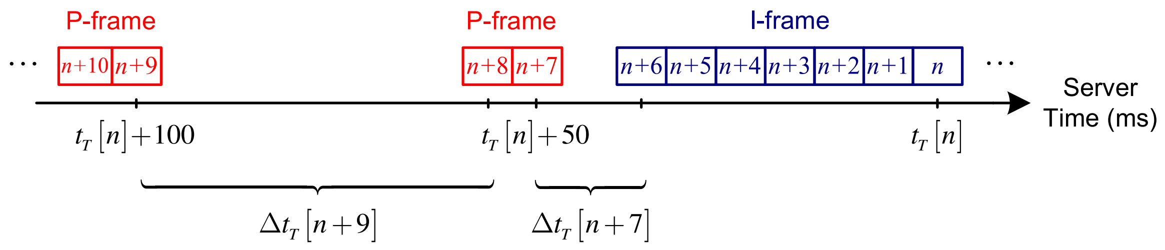


Figure 40: The variation of the intertransmission times at the server side.

As mentioned above, the variation in the intertransmission times depends on many factors. Fortunately, our tests with a standard H.264 video codec [120] show that this variation follows a similar pattern for successive GOPs, provided that the encoder and video-specific parameters are kept the same. However, in some configurations, the pattern breaks because of a sudden scene change. Nevertheless, without loss of generality we can safely assume that the pattern for the first GOP is also valid for the subsequent GOPs, and the server conveys any new pattern information to the client when there is a change in the encoding/packetization process.

5.3.2 Timeout Mechanism

Similar to intertransmission times, we define interarrival times at the client. The interarrival time of packet n is equal to the amount of the time passed since the last arrival. The interarrival time for packet n is given by

$$\Delta t[n] = t_A[n] - t_A[n^*], \tag{56}$$

where $t_A[n]$ denotes the arrival time for packet n, and n^* is the last successfully-received packet before packet n. The main idea behind our RTO estimation method is to estimate the subsequent interarrival time and project the corresponding arrival time after each packet arrival.

We use the notation of $\widetilde{\Delta t}[.]$ and $\widetilde{t_A}[.]$ to denote the estimated interarrival and projected arrival times, respectively. The estimation is based on the last-observed interarrival time and Δt_T. That is,

$$\widetilde{\Delta t}[n] = \mathbf{F}\left(\Delta t[n-1], \Delta t_T[n]\right), \tag{57}$$

for some function $\mathbf{F}$. For example, in Figure 41 packet n is expected to arrive by $\widetilde{t_A}[n]$, which is given by

$$\widetilde{t_A}[n] = t_A[n-1] + \widetilde{\Delta t}[n]. \tag{58}$$

In case of packet n does not arrive within the estimated time frame, the client presumes that this packet is lost and times out.

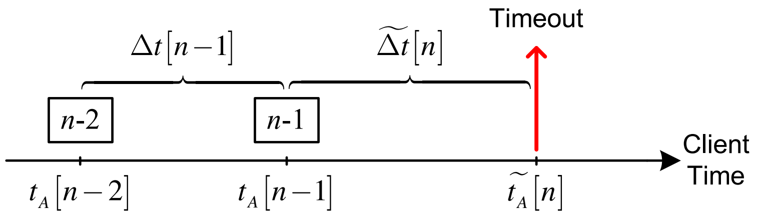


Figure 41: Timeout mechanism.

We benchmark the performance of our RTO estimator with two metrics. First, we define p_f as the estimation failure probability, *i.e.*, the probability of identifying a non-lost packet as it is lost, which can be computed from

$$p_f = P\left\{\widetilde{t_A} < t_A \text{ and } t_A < \infty\right\}. \tag{59}$$

The second metric is the average overwaiting time. For a non-lost packet n, the overwaiting time is defined as

$$o[n] = \begin{cases} \widetilde{t_A}[n] - t_A[n], & \text{if } t_A[n] < \widetilde{t_A}[n] < \infty; \\ 0, & \text{if } \widetilde{t_A}[n] \leq t_A[n] < \infty. \end{cases} \tag{60}$$

On the other hand, the overwaiting time spent for a lost packet can be computed by using a hypothetical arrival time extrapolated from the last packet arrival time. That is,

$$o[n] = \widetilde{t_A}[n] - \left(t_A[n^*] + \sum_{n^* < n' \leq n} \Delta t_T[n'] \right), \text{ if } t_A[n] = \infty, \tag{61}$$

where n^* is the last successfully-received packet. Naturally, there is a trade-off between the estimation failure probability and the average overwaiting time. Estimation failures can be largely avoided, if the client can tolerate a prolonged amount of time before timing out. As this ability diminishes, the client starts giving wrong decisions and may identify late packets as they are lost.

5.3.3 Methodology

Let us start our discussion with the analysis of two packet traces. We produced these traces by simulating a moderate-sized Internet topology [104] in *ns*-2 network simulator [134]. The first trace corresponds to a video sequence (176×144) encoded with a standard H.264 codec [120] at 300 Kbps and a frame rate of 25 frames per second. The GOP structure was one I-frame plus nine P-frames. With an IP packet size of 1500 bytes, each video frame fitted exactly into one IP packet. This resulted in a regular, *i.e.*, non-bursty, video traffic with equal intertransmission times of 40 ms. On the other hand, the second trace corresponds to a bursty video traffic. Specifically, the second video sequence (352×288) was encoded at 600 Kbps and a frame rate of 20 frames per second. The GOP structure was one I-frame plus nine P-frames, where each I-frame and P-frame was packed into seven and two IP packets, respectively.

The forward-trip times and interarrival times extracted from a small segment of the traces are presented in Figures 42 and 43 for both traces. The corresponding distributions are also given in Figure 44. Note that for the lost packets, arrival times and interarrival times are equal to infinity and are not shown. Although an identical network topology was used in producing both traces, the loss rates experienced by the non-bursty and bursty video traffics were not equal, and measured as 2.2% and 3.9%, respectively.

A first look on the distributions shows that forward-trip times have larger variations compared to the interarrival times. The amount of variation further increases with the burstiness of the video traffic. Yet, the autocorrelation plots for the forward-trip times in Figure 45 suggest that the consecutive samples are highly correlated. Despite that the bursty video traffic has a variable intertransmission time, the forward-trip time samples corresponding to this trace have a higher correlation. This is mainly because the bursty video traffic has a smaller mean intertransmission time ($\Delta T = 20$ ms), and hence, more samples are collected in a unit time period compared to the non-bursty video traffic, which has a mean intertransmission time of 40 ms. On the other hand, the interarrival times exhibit a different behavior. Although interarrival time samples are mostly confined within a small region, their correlation is relatively smaller due to the noisy behavior caused by the other background TCP/UDP flows.

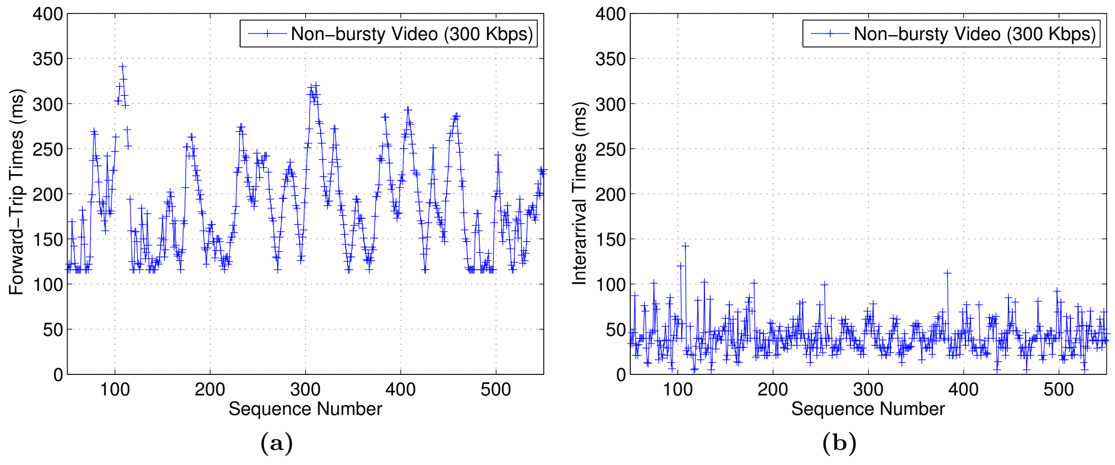


Figure 42: Variation of the forward-trip times (a) and interarrival times (b) for the non-bursty video traffic.

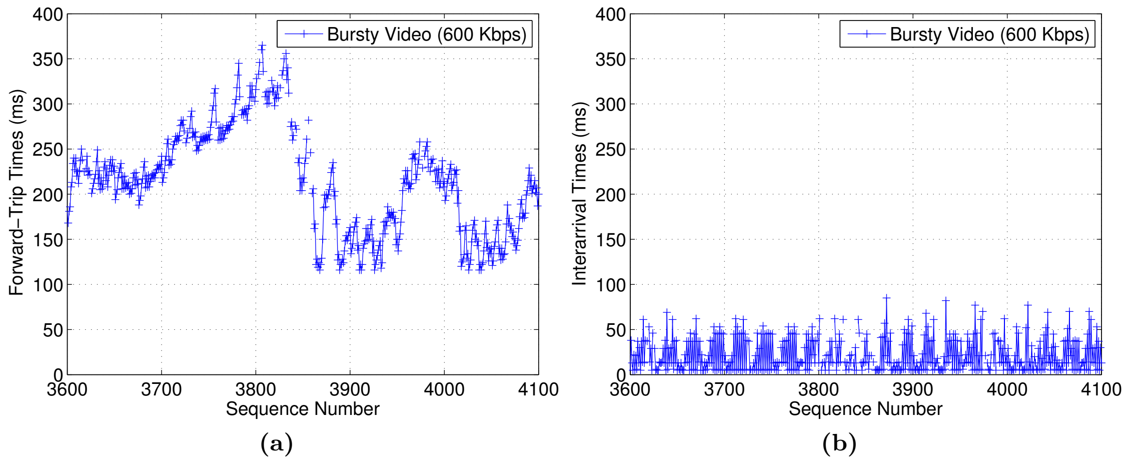
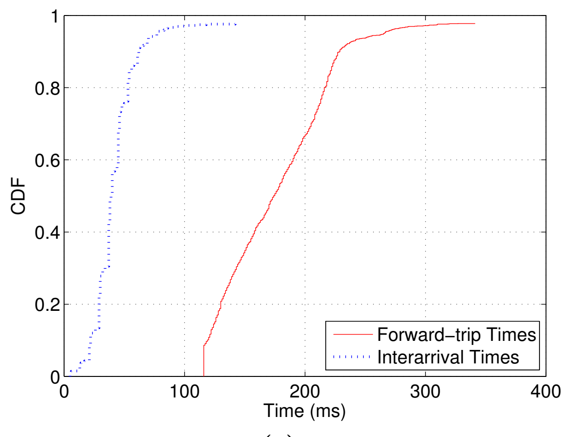


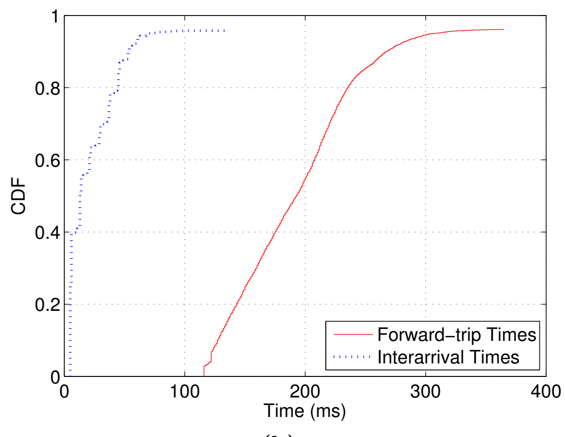
Figure 43: Variation of the forward-trip times (a) and interarrival times (b) for the bursty video traffic.

In continuous-media applications, the client successively receives packets. Upon receiving a packet, the client can estimate the interarrival time of the subsequent packet and compute its projected arrival time by using (58). In this study, we propose the following estimator for the interarrival times:

$$\widetilde{\Delta t}[n] = \max\ \left(\Delta t_T[n], \alpha \times \Delta t[n-1] + \beta \times \Delta t_T[n]\right), \tag{62}$$



(a)



(b)

Figure 44: Distributions of the forward-trip and interarrival times for the non-bursty video traffic (a) and bursty video traffic (b).

where α and β are some constants that determine the responsiveness of the estimator[5]. For example, with $\alpha = 0$ and $\beta = 1$ we get an extremely aggressive estimator, which achieves a small average overwaiting time but potentially a very high failure rate. Using a larger β value may lower the estimation failure probability, however, inevitably increases the average overwaiting time. As shown in Figures 42 and 43, the interarrival time samples are bounded except some impulsive points. If we ignore these impulsive points for the time being, a particularly successful estimator can be achieved with $\alpha = 7/8$ and $\beta = 3/8$ for the non-bursty video traffic and with $\alpha = \beta = 7/8$ for the bursty video traffic. We observe that (62) can closely track the actual arrival times and achieve a small average overwaiting time with these parameters. However, it is highly susceptible to sudden delay increases. In particular, for the non-bursty and bursty video simulations, the observed failure probability is 20.8% and 24.0%, respectively. We address this problem next.

Because of the bursty behavior of the TCP flows in the background, it is possible to observe some abrupt increases in the packet interarrival times, *e.g.*, see packets #108 and #383 in Figure 42, and packets #3872 and #3935 in Figure 43. For a successful RTO

[5]Note that the estimator in (62) requires a packet arrival to compute the subsequent arrival time. Hence, in case of a bursty packet loss or an excessively-delayed packet, this estimator may halt. To avoid such interruptions, the interarrival time estimate for packet n is initialized to $\Delta t_T[n]$. When a new packet arrives, the estimate for the subsequent packet is updated accordingly.

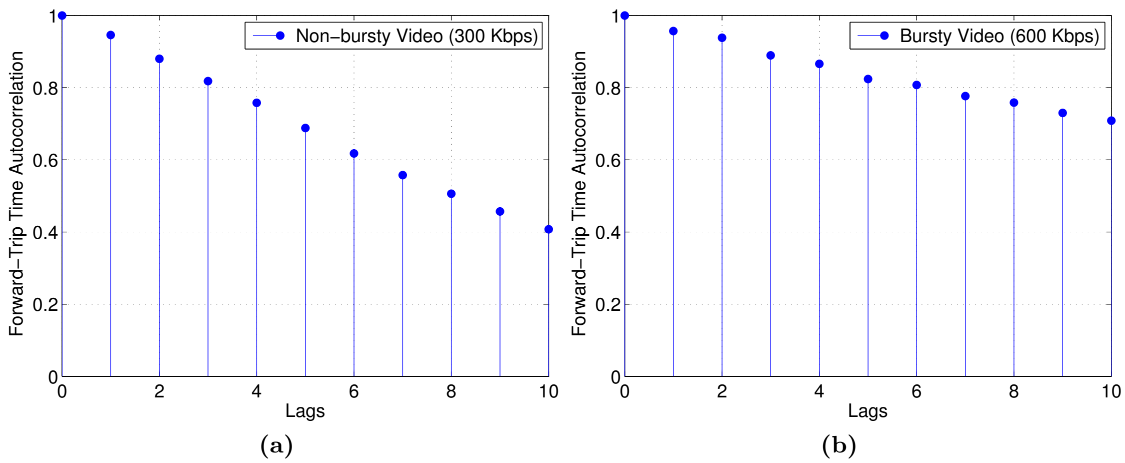


Figure 45: Correlation of the forward-trip times for the non-bursty video traffic (a) and bursty video traffic (b).

estimation, these impulsive points must be caught. Since these sudden increases occur rarely in the interarrival times, rather than modifying (62), *e.g.*, increasing the value of α and/or β, we introduce a new concept, called Timer$_{late-packet}$. As depicted in Figure 46, this is basically a supplementary timer that absorbs large delay spikes. The client starts Timer$_{late-packet}$ when the expected packet does not arrive by the initial projected arrival time. If the expected packet does not arrive until Timer$_{late-packet}$ expires, the client registers the packet as lost and times out. When a new packet (either the expected one or another packet) is received, the client updates the duration of Timer$_{late-packet}$, denoted by δ, as well as the projected arrival times of the subsequent packets, if necessary. It is important to note that since the proposed RTO estimator operates on the interarrival times rather than the arrival times, Timer$_{late-packet}$ defers the estimated arrival time not only for the expected packet but also for all subsequent packets. Also note that once Timer$_{late-packet}$ is started, the client does not start a second one until a new packet arrives. As opposed to exponentially backing off [135], this strategy allows us to keep overwaiting time considerably shorter without sacrificing the accuracy.

With the introduction of Timer$_{late-packet}$, the RTO estimation failure probability for the non-bursty and bursty video simulations reduces to 0.1% and 0.3%, respectively. The respective incurred costs are a small increase from 18 to 26 ms and from 19 to 26 ms in the

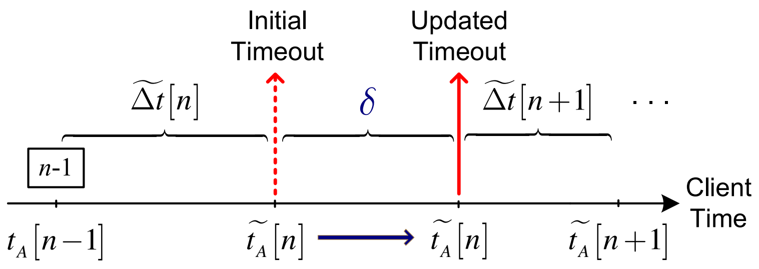


Figure 46: Illustration of Timer$_{late-packet}$.

average overwaiting time. The substantial gain at the expense of a negligible cost clearly proves the benefit of employing Timer$_{late-packet}$ in the RTO estimation.

In the implementation of Timer$_{late-packet}$, the initial value of δ can be set to an arbitrary value since it is revised every time the timer is used. Its value is updated only if the received packet is not the expected packet, or it is the expected packet but it arrived after its estimated arrival time. Let n_i and n_j be the last packet received before Timer$_{late-packet}$ is started and the first packet received after Timer$_{late-packet}$ is started, respectively. We set the value of δ as follows:

$$\delta = \begin{cases} \min\left(\delta_{max}, t_A[n_j] - t_A[n_i]\right), & \text{if } n_j > n_i + 1 \text{ or } t_A[n_j] > \widetilde{t}_A[n_j]; \\ \delta, & \text{ow}, \end{cases} \tag{63}$$

where δ_{max} represents the upper limit for δ. In this study, we observed that $3 \times \Delta T$ was a good choice for δ_{max}.

We examine the performance of the RTO estimator given in (62) by varying the values of α and β. In Tables 5 and 6, we tabulate the results for several $\{\alpha, \beta\}$ pairs as well as the performance of an enhanced TCP-like RTO estimator[6]. A subset of these estimators are also compared in Figures 47 and 48. The results show that the RTO estimation performance varies substantially depending on the values of α and β. Evidently, poorly selected parameters result in a large number of redundant timeouts, which may potentially lead to a significant performance degradation. Another important observation is that RTO estimation becomes more difficult as the burstiness of the video traffic increases. It is clear that depending on the intertransmission time variation, a fine tuning may be required for α

[6]We study this RTO estimator in detail in Chapter 6.

and β in order to get the best RTO performance. However, finding the optimal $\{\alpha, \beta\}$ pair is beyond the scope of this chapter. For the remainder of this chapter, we will use a good pair of α and β based on the results given in Tables 5 and 6. In Chapter 6, we will further address this issue and study an adaptive RTO estimation method.

Table 5: Comparison of different RTO estimators for the non-bursty video traffic.

	Enhanced TCP-like	$\alpha = 0$ $\beta = 1$	$\alpha = 0$ $\beta = 2$	$\alpha = 1$ $\beta = 0$	$\alpha = 2$ $\beta = 0$	$\alpha = 7/8$ $\beta = 3/8$	$\alpha = 7/8$, $\beta = 3/8$ w/ Timer$_{late-packet}$
p_f	0.9%	40.7%	1.1%	31.6%	5.6%	20.8%	0.1%
Mean o	109 ms	9 ms	41 ms	14 ms	46 ms	18 ms	26 ms

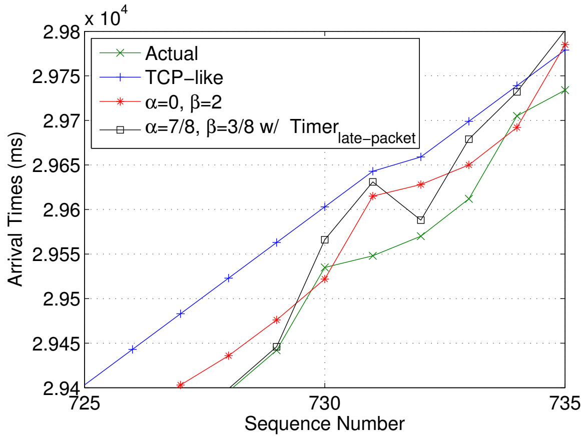


Figure 47: Actual and estimated arrival times for the non-bursty video traffic.

Table 6: Comparison of different RTO estimators for the bursty video traffic.

	Enhanced TCP-like	$\alpha = 0$ $\beta = 2$	$\alpha = 0$ $\beta = 4$	$\alpha = 2$ $\beta = 0$	$\alpha = 4$ $\beta = 0$	$\alpha = 7/8$ $\beta = 7/8$	$\alpha = 7/8$, $\beta = 7/8$ w/ Timer$_{late-packet}$
p_f	1.1%	18.6%	3.0%	24.2%	16.1%	24.0%	0.3%
Mean o	118 ms	27 ms	64 ms	35 ms	69 ms	19 ms	26 ms

Having introduced our RTO estimator, we next combine it with a retransmission-based error-control method. With simulations, we investigate the impact of using an I-Proxy on the performance of error control. Later in Section 5.6, we conduct a similar analysis with Internet experiments.

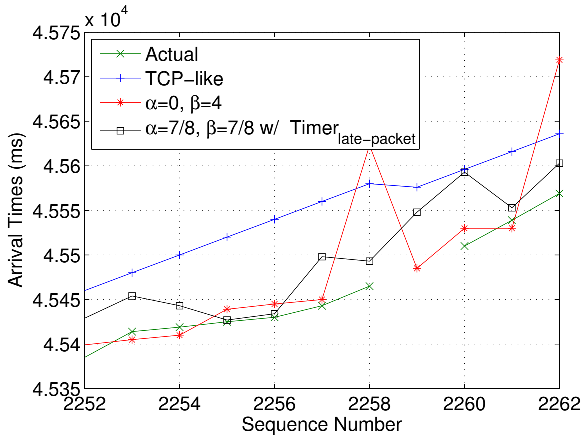


Figure 48: Actual and estimated arrival times for the bursty video traffic.

5.4 *Impact of the I-Proxy on the ARQ Performance*

In this section, we analyze the ARQ performance for the non-bursty video traffic when an I-Proxy is setup between the server and client, and when there is no I-Proxy available, and retransmissions are done on an end-to-end basis. In both scenarios, retransmissions are decided by the RTO estimator given in (62). For RTO estimation, we use parameters $\alpha = 0.875$ and $\beta = 0.375$, and enable Timer$_{late-packet}$. First, we provide the delay distribution of the packets that failed in the first transmission attempt, (i) when no I-Proxy is used, and retransmissions decisions are solely given by the client (end-to-end retransmission), (ii) when an I-Proxy is used, and retransmission decisions are solely given by the I-Proxy (fast retransmission), (iii) when an I-Proxy is used, and retransmission decisions are solely given by the client (early retransmission), and (iv) when an I-Proxy is used, and retransmission decisions are given by both the client and I-Proxy (both fast and early retransmission). In doing so, our goal is to quantify the amount of individual improvements that are contributed by fast and early retransmissions. Second, we compare the distribution of all packet delays observed when an I-Proxy is available, and performs both early and fast retransmission with the case when an I-Proxy is not available.

As the effective packet loss rate after one retransmission opportunity is negligibly small, we limit the number of retransmissions per packet to one. Figure 49 shows the delay distributions in different scenarios for the packets that could not make it in their first transmission attempts. Not surprisingly, packets retransmitted on an end-to-end basis experience considerably larger delays than the ones recovered by the help of the I-Proxy. We also observe that fast retransmission delivers the retransmitted packets slightly faster than early retransmission, although fast retransmission can only recover almost half of the lost packets[7]. This is mainly because that in case of fast retransmission, retransmission decisions are given by the I-Proxy, which naturally observes a smaller amount of jitter, and hence, does a more accurate RTO estimation compared to its client-based counterpart. However, fast retransmission cannot recover the packets that get lost between the I-Proxy and client. In contrast, any lost packet can be detected by the client, and the ones that get lost between the I-Proxy and client can be recovered by early retransmission. For the remaining packets, however, early retransmission performs exactly as an end-to-end retransmission, which inevitably increases the total recovery time.

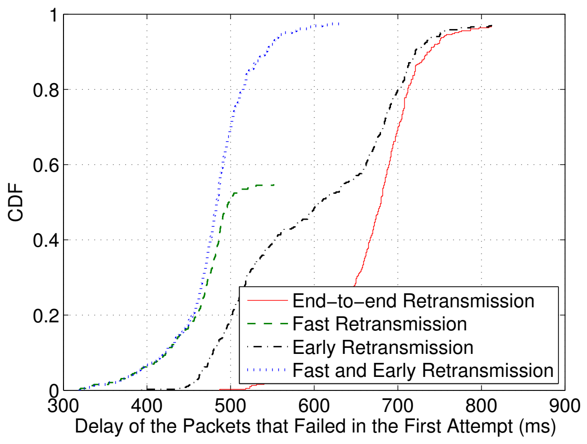


Figure 49: Delay distributions of the packets that failed in the first transmission attempt.

Now, we enable both early and fast retransmission on the I-Proxy. In Figure 50, we compare the delay distribution of all packets that correspond to the cases when an I-Proxy is used and when one is not used. In Figure 50, our focus is particularly on the delay performance of the retransmitted packets since the packets that go through in the first transmission attempt experience the same amount of delay in both cases[8]. The main result

[7]Note that the comparison between fast and early retransmission is dependent on the location of the I-Proxy, and the path characteristics between the I-Proxy and other ends. In our simulations, the location of the I-Proxy was selected in a way such that each sub-network had similar delay and packet loss characteristics.

[8]In reality, transmission through an I-Proxy can cause extra delays compared to a direct transmission

here is that early and fast retransmission are complementary of each other, and we can achieve large reductions in the error-recovery time by using an I-Proxy. For example, if the playout delay is set to 500 ms for a real-time application, the I-Proxy delivers 99% of the packets on time, whereas only 97% of the packets can be delivered on time without the I-Proxy. Likewise, to achieve a packet success rate of 99%, the playout delay should be set to 700 ms if no I-Proxy is used, which is 200 ms larger than the sufficient amount of the buffer required by the I-Proxy approach. The impact of delivering 2% more packets or the ability of reducing the playout delay without sacrificing the quality can be substantial in video applications. Our simulations also show that the amount of the performance gap between the cases when an I-Proxy is used and not used increases further as the network conditions deteriorate.

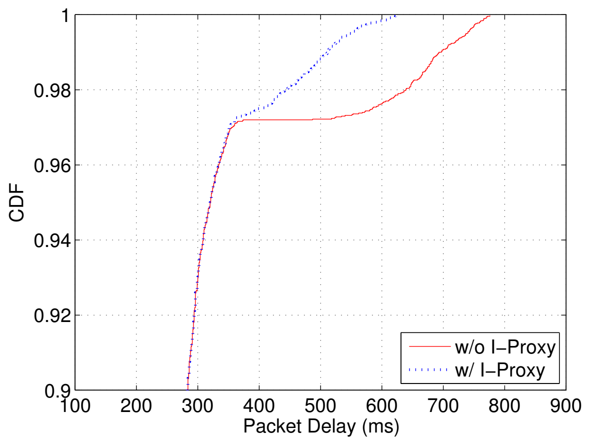


Figure 50: Delay distributions of all packets.

Another important observation from our simulations is that the location of the I-Proxy plays an important role on the ARQ performance of the I-Proxy system. In Appendix C, we investigate this issue further, and study mathematical and practical I-Proxy selection methods.

5.5 Implementation Details

In order to conduct Internet experiments, we developed a prototype streaming system, which consisted of a server, a client and an I-Proxy. Our RTO estimation method was

from the server to the client. See Section 5.5 for details.

implemented as an application-layer protocol and ran on top of UDP. In this section, we briefly summarize the system components and discuss basic design issues.

Our system has three main components:

- Video Server: This is a Win32 application programmed in Java. It captures live video from a source such as a webcam or camcorder. After encoding and packetizing the video, it transmits the packets to the I-Proxy.

- Proxy: This is a simple console application programmed in Java, which runs the RTO estimator studied in Section 5.3. It sends feedback to the server application, and has the functionalities of forwarding, caching, early and fast retransmission.

- Streaming Client: This is a Win32 application programmed in Java. It receives video packets, decodes and then displays them. The RTO estimation method is also implemented at the client.

During the development, the following two issues were important in accomplishing the desired functionality out of our proxy server:

- Caching Strategies: To perform a retransmission-based error-control method, the I-Proxy needs to cache any received packet for a possible retransmission (See Section 5.2.1). However, if we consider the practicability of the system, it is clear that caching each received packet severely limits the scalability of the system. To overcome this problem, the cached packets should be replaced by the new ones without disturbing the functionality of the I-Proxy. Fortunately, our target applications are interactive video applications, in which packets have short lifetimes. Any packet becomes essentially obsolete and can be removed from the cache when the display time of the last frame that depends on this packet (usually the last frame in a GOP) passes[9]. For example, if the transmitted video has a GOP size of 10 frames and the frame rate is 20 frames per second, the critical time to hold the packets in the cache is at most

[9]The GOP structure or packet deadline information is conveyed to the I-Proxy either before the session starts or during the session with packet header updates.

500 ms. Thus, the storage requirement for one-way transmission can be computed by $0.5 \times R$ (Kbits), where R is the video rate in Kbps. Considering that the typical rate for a videotelephony application varies between 40 - 512 Kbps, we can calculate the storage cost for a single session (two-way video transmission) as 5 - 64 KB. Since these storage requirements are far smaller compared to the storage capacity of a commercial (or even a personal) proxy server, several sessions can be run concurrently through a single I-Proxy.

- Switching Overhead: Another issue regarding the scalability of the I-Proxy is the increased path delay and switching overhead. As the video packets are routed to the other end-user via the I-Proxy, an extra delay is inevitably introduced. However, with a proper I-Proxy selection, this overhead can be reduced to an acceptable level. On the other hand, switching overhead is directly related to the processing power and connectivity of the I-Proxy. For example, the I-Proxy we used in our experiments was a Linux-based system with an Intel Pentium-III 1.0 GHz processor, and had a connection of 100 Mbps. We observed that even for a non-optimized implementation, the switching delay per video flow was less than 2 ms. Provided that a commercial proxy with more processing power and higher connection speed is used for caching and forwarding purposes, this overhead may get smaller. Based on these measurements, we can safely assume that the I-Proxy approach scales well to a medium number of simultaneous video sessions without disturbing the individual flows.

5.6 Internet Experiments

In this section, we present experimental results to evaluate the performance improvements gained in a delay-sensitive application by employing an I-Proxy. For this purpose, we setup a videotelephony session of 30 minutes between a client connected to Georgia Tech network and a broadband client connected to an ISP in Ankara, Turkey. The I-Proxy was located at Bilkent University in Ankara, Turkey. The corresponding configuration is depicted in Figure 51. In the rest of the section, we refer to the clients as C_{US} and C_{TR}, respectively.

Before running the sessions, we first conducted initial experiments to measure the path

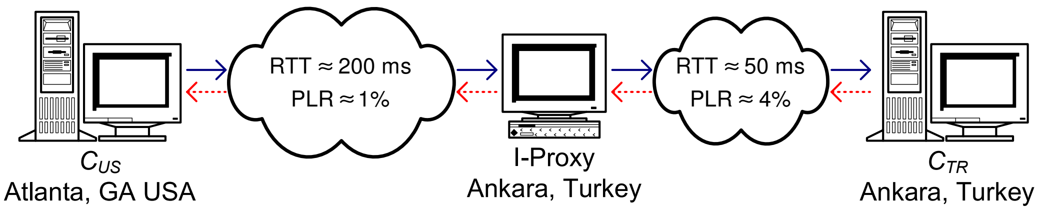


Figure 51: Experimental setup.

characteristics such as packet loss rate, delay and jitter. These experiments were run between each client and the I-Proxy. We measured a low one-way packet loss rate (around 1%) and stable round-trip time (RTT), averaging 200 ms, between C_{US} and the I-Proxy, whereas we observed a higher one-way packet loss rate (around 4%) and more variable RTT between C_{TR} and the I-Proxy, although C_{TR} was both physically and network-wise closer to the I-Proxy (mean RTT was 50 ms). We suspect that this result is mainly because of the better connectivity of both the Georgia Tech and Bilkent campuses.

In the experiments, the video encoding rate was set to 120 Kbps. We used a standard H.264 codec [120] to encode the test sequence Foreman (176×144) at a frame rate of 10 frames per second. As each frame fitted into one packet, the clients transmitted 1500-byte video packets with a fixed intertransmission time of 100 ms. During the transmission, the packets that could not be delivered in 200 ms were not displayed and counted as unsuccessful packets. However, the packets arriving after their decoding deadlines were still used to decode the subsequent predictively-coded frames and reduce the error propagation. In order to quantify our findings, we present our results in terms of both the percentage of successful packets and average video quality. For the latter, we use the peak signal-to-noise ratio (PSNR) measure on the luminance (Y) channel.

5.6.1 Direct Video Transmission between C_{US} and C_{TR}

In this experiment, both clients transmitted the video packets directly to each other. Although this method avoids the extra delays incurred when relaying over the I-Proxy, the observed delay statistics did not differ much on the end-to-end basis; we measured the mean RTT as 240 ms. Since this high RTT value rendered any end-to-end retransmission attempt impractical, each packet had only one transmission opportunity. By the end of the

30-minute videotelephony session, we determined the percentage of successful packets as around 91% for both clients, which actually produced a choppy video. 59% of the unsuccessful packets were lost, and remaining 41% were late and missed their decoding deadlines. In terms of video quality, both clients achieved merely around 31 dB, *i.e.*, 7 dB lower than the original quality.

The main result of this experiment is that lost packets are inevitable when there is only one transmission opportunity in the best-effort Internet, and without an error-control/protection method the video quality suffers. However, note that some packets will be excessively delayed without being lost due to provisioning of the underlying physical network. In such cases, doing a retransmission, either from the server or I-Proxy, is not a solution as it is highly unlikely that a retransmission will be received earlier than the previous transmission(s).

5.6.2 Video Transmission from C_{US} to C_{TR} via I-Proxy

In our initial measurements, we observed more favorable characteristics between C_{US} and the I-Proxy compared to the those between C_{TR} and the I-Proxy, *e.g.*, packets mostly got lost between the I-Proxy and C_{TR}. Considering the short mean RTT (≈ 50 ms) in this sub-network, the lost packets can be recovered by early retransmissions from the I-Proxy before their decoding deadlines pass. To do so, we enable RTO estimation at C_{TR} by adopting the estimator given in (62) with $\alpha = 7/8$, $\beta = 3/8$ and $\text{Timer}_{late-packet}$. With these parameters, the successful packet rate increases from 91.1% to 95.1%, which improves the average video quality rendered at C_{TR} from 31.2 dB to 33.9 dB.

Because of the potential failure of the RTO estimation, there may be some redundant retransmissions that increase the rate consumed by the application. A retransmission request is counted as redundant, if any previous transmission attempt becomes a success. In this experiment, we observed that 5.6% of the requests caused spurious retransmissions. In other words, our estimator failed in estimating the arrival time for 0.3% of the total transmitted packets.

5.6.3 Video Transmission from C_{TR} to C_{US} via I-Proxy

Next, let us focus on the video transmission from C_{TR} to C_{US}. As mentioned above, because of the uneven path characteristics, the majority of the packet losses are experienced between C_{TR} and the I-Proxy. If the I-Proxy does not monitor the packets coming from C_{TR} and does not take the necessary actions for the lost ones, C_{US} will eventually have to detect these lost packets and ask for a retransmission. However, because of the large delay between C_{US} and C_{TR}, these retransmission attempts will be essentially useless. To overcome this problem, the I-Proxy should identify the lost packets and send a retransmission request to C_{TR}. As a result of enabling fast retransmission on the I-Proxy, the successful packet rate increases from 91.3% to 95.5%, which improves the average video quality rendered at C_{US} from 31.4 dB to 34.4 dB. Overall, 2.4% of the retransmission requests made by the I-Proxy were redundant, *i.e.*, RTO estimation failed in 0.1% of the total transmitted packets.

We summarize our results in Table 7. The results clearly demonstrate that we can maintain a more stable video experience by the help of the improved error control provided by the I-Proxy.

Table 7: Experimental results for the Foreman sequence.

	% of Successful Packets	Average Quality	% of Redundant Ret. Requests	Mean Total Transmission Rate
Without I-Proxy				
$C_{US} \rightarrow C_{TR}$	91.1%	31.2 dB	N/A	120 Kbps
$C_{TR} \rightarrow C_{US}$	91.3%	31.4 dB	N/A	120 Kbps
With I-Proxy				
$C_{US} \rightarrow C_{TR}$	95.1%	33.9 dB	5.6%	125.1 Kbps
$C_{TR} \rightarrow C_{US}$	95.5%	34.4 dB	2.4%	124.9 Kbps

5.7 Conclusions

In this chapter, we presented a practical use of proxies in delay-sensitive video applications. In particular, by proposing an efficient and accurate RTO estimation method, we showed the potential benefits of using proxies in providing improved error-control/protection capabilities to the clients communicating with each other over long distances. As we leverage the existing IP infrastructure in transmitting and relaying the video packets, our approach does not require any native network support. This makes our approach advantageous for the worldwide service and application providers that seek cost-effective and efficient solutions for delivering interactive video.

CHAPTER VI

DELAY AND DELAY-BOUNDARY PREDICTION FOR PACKET VIDEO

Time-constrained error recovery is an integral component of reliable low-delay video applications. Regardless of the error-control/protection method adopted by the application, unacknowledged or missing packets must be quickly identified as lost or delayed, so that necessary actions can be taken by the server/client on time. Historically, this problem has been referred to as retransmission timeout (RTO) estimation. Earlier studies show that existing RTO estimators suffer from either long loss detection times or a large number of spurious timeouts. The goal of this study is to address these problems by devising a two-step RTO estimation method that is specifically tailored for low-delay video applications. These two steps are (*i*) delay prediction, and (*ii*) delay-boundary prediction. For delay prediction, we develop an adaptive linear delay predictor that produces the best estimate in terms of the mean-squared error criterion by exploiting the temporal dependence among the packet delay samples. For delay-boundary prediction, on the other hand, we develop a controller that optimally manages the trade-off between the amount of overwaiting and rate of spurious timeouts. As opposed to existing methods, our approach is completely adaptive to the source video transmission rate and time-varying network conditions, and does not use any preset parameters.

6.1 Introduction[†]

In packet-switched IP networks, packets are transmitted from a source node to a destination via several intermediate network devices whose resources such as bandwidth and processing power are shared among the connections flowing through them. While this architecture facilitates a good utilization of the network capacity, it lacks the ability of providing any

[†]Parts of this chapter were previously published in [136, 137].

guarantee on the delays that individual packet flows experience. Depending on the characteristics of the cross traffic at each intermediate router/switching device on their path, the packets are variably delayed. Often, the video applications are forced to pre-buffer some content before they start playing out the streamed video to avoid potential interruptions that may result from excessively-delayed packets. However, adopting a large playout buffer is not viable for several applications such as videotelephony, videoconferencing and distance learning, or not desirable for applications such as on-demand video and IP television (IPTV).

Understanding the nature of packet delay is important for service providers, and protocol and application developers in appropriate network provisioning, designing routing and transport protocols, and developing congestion and flow algorithms. While conventional queuing theory can be used to lay out an analytical framework based on the Markov models, the assumptions of independent arrival and service times are rarely valid in today's Internet. Therefore, such theoretical derivations are of a little use in practice. The study by Li [138] shows that a stable queuing system has a multistructure, where packet delays can be modeled by a stationary process with non-stationary sub-processes. The stationarity perception on measured packet delays depends on the sampling interval. For example, if we measure the delay of the packets that are transmitted at intervals smaller than the average congestion duration, we probably capture the non-stationary behavior. However, when the packets are transmitted at large intervals, our measurements likely reveal the stationary behavior.

The ability of accurate delay and delay-boundary prediction in real time is an integral component for many of the layers in the video communication protocol stack, *e.g.*, rate control, error control and network adaptation. A delay-sensitive video application has to take well-timed actions against unacknowledged and missing packets. It is also as critical that the application employs network-adaptive error-recovery methods and packet scheduling algorithms so that the application can sustain an acceptable level of quality in the event of deteriorated network conditions. Given the highly-dynamic nature of the Internet, delay-boundary prediction for the incoming packets, or, in other words, determining when a missing packet can be considered *lost* as opposed to *late* without exceeding the desired spurious timeout rate is not an easy task. Let us illustrate this point on an example.

Consider a video-on-demand session running over UDP between a server and a client, where the server continuously transmits video packets to the client, and the client reports missing packets to the server with negative acknowledgements (NACKs); a NACK message is generated for a packet when the client decides that the packet is lost. If the NACKs are received by the server early enough, missing packets can be retransmitted successfully before their decoding deadlines pass[1]. As a rule of thumb, the client should not time out pre-maturely for the excessively-delayed packets, since under normal circumstances it is highly unlikely that a retransmitted packet will arrive earlier than the initially-transmitted packet.

Needless to say, the primary challenge is that the client has to decide on timeouts merely by observing the packet arrivals in the course of a streaming session. It is never a clear-cut decision whether a missing packet has been lost or delayed. Naturally, a trade-off between overwaiting and spurious timeouts is present. To address this problem, in Chapter 5, we introduced a client-driven method that used packet interarrival times for retransmission timeout (RTO) estimation. The proposed approach was computationally efficient, and substantially outperformed an enhanced TCP-like RTO estimator by reducing both the amount of overwaiting and rate of redundant retransmissions. In Chapter 5, however, we did not provide a formal way to compute the parameters used in the RTO estimation. Our experiments with several video streams encoded at different bitrates later showed that the best-performing set of parameters varied for each stream, and there was not a global optimal solution that would work for every video traffic. This motivated us to develop an adaptive RTO estimation method that would configure itself based on the source video transmission rate and time-varying network conditions.

[1]Naturally, retransmission-based error-control methods are unsuitable for multimedia applications where the extra delay introduced by the retransmissions is prohibitively large. However, due to emerging broadband technologies, end-to-end delays experienced by Internet users today are comparably smaller. Consequently, retransmission-based error-control methods can still be accommodated by many of today's low-delay multimedia applications.

In this study, we devise a novel RTO estimation method that involves two main steps. In the first step, an adaptive linear delay predictor produces the best estimate in terms of the mean-squared error criterion by exploiting the temporal dependence among the packet delay samples. In the second step, on the other hand, a controller optimally manages the trade-off between the amount of overwaiting and redundant retransmission rate by regulating the bias to be added to the estimate produced in the previous step. This controller has two different modes of operation: (*i*) *media-unaware*, and (*ii*) *media-aware*. In the media-unaware mode, the controller ignores the unequal importance of the video packets and treats each of them equally. In the media-aware mode, however, the controller prioritizes the packets that carry a more important payload and the packets whose decoding deadlines are sooner, over the less important and non-urgent packets. This way, a higher rendering quality is attained at the client side without any additional increase in the rate. Our approach has three main contributions:

- We develop an adaptive delay predictor for high-bitrate video applications. A large number of multimedia protocols such as packet scheduling algorithms and adaptive buffer management techniques can potentially benefit from this predictor [52, 139].
- We derive an optimal media-unaware redundancy-controllable timeout estimator. This estimator allows applications to recover as many packets as possible under a given redundant rate budget.
- We formulate an optimal media-aware timeout estimator that jointly considers the interdependency relations among the video packets as well as their decoding deadlines in computing the timeout estimates while still conforming to the redundant rate constraint dictated by the application or the network.

To the best of our knowledge, neither a redundancy-controllable nor a media-aware RTO estimation method has been previously proposed. To keep the discussion concrete and focused, this chapter explores the adaptive delay prediction and media-unaware timeout estimation only. We defer the discussion of the media-aware timeout estimation to Chapter 7.

It is important to note that not all of the applications necessarily use retransmissions to recover from errors. With our proposed RTO estimation method, one can also employ different types of error-control/protection methods. For example, based on the delay/loss predictions, the amount of redundancy in channel coding or the amount of error resiliency in video coding can be optimally adjusted to minimize the impact of packet erasures.

One of the earliest RTO estimation methods is the Jacobson's algorithm [140], which uses an exponentially-weighted moving average (EWMA) approach. Currently, TCP employs this algorithm with some modifications [141]. This class of RTO estimators have been thoroughly examined by Loguinov and Radha in the context of a video streaming application [142]. Their empirical study concluded that EWMA-based RTO estimation was not quick enough to detect lost packets. The authors also suggested using jitter samples for fine-tuning the estimations. Although [142] presents important findings, its scope is rather limited, since the study primarily focuses on a low-bitrate video streaming application with a large playout buffer. On the TCP end, other proposals for replacing [140] are [143–146]. However, these approaches are not suitable for low-delay video applications either, due to their conservative estimates and slow adaptation to time-varying network conditions.

In a recent study [147], Sinha and Papadopoulos proposed a timerless retransmission protocol that eliminated the pitfalls of round-trip time (RTT) estimation and timer-triggered timeouts. In this protocol, a lost packet can only be identified upon detection of a gap in the received packets. Hence, when a batch of packets are lost or excessively delayed, this protocol has to wait indefinitely until a new packet is received, which may impede the timely recovery of the bursty losses. Previously, Papadopoulos and Parulkar used an algorithm similar to [140] for real-time streaming [21]. A different approach was later proposed by Rhee [148], where retransmission decisions were based on multiples of frame durations. However, these approaches are not adaptive and may not perform well when streaming high-bitrate video under low-delay requirements.

In the rest of the chapter, we continue with an overview of different RTO estimation methods in Section 6.2. In Section 6.3, we study autoregressive models for packet delay prediction. Section 6.4 discusses the media-unaware RTO estimation method. Simulation results and a detailed analysis of different RTO estimation methods are presented in Section 6.5. Finally, Section 6.6 concludes the chapter with a summary of our main findings.

6.2 Overview of RTO Estimators

In this section, we briefly summarize three different classes of previously proposed RTO estimators. Later, in Section 6.5, we compare our approach with these estimators in terms of their performances.

6.2.1 TCP-Like RTO Estimators

The RTO estimation algorithm used in current TCP implementations is based on Jacobson's algorithm [140], which was later modified in [141]. In TCP, the TCP sender records a new RTT measurement when it receives an unambiguous acknowledgement packet. Let $\mathbf{r}[n]$ denote the RTT observation corresponding to packet n. Jacobson's algorithm predicts the RTT of the subsequent packet, denoted by $\tilde{\mathbf{r}}[n+1]$, by computing the following moving average:

$$\tilde{\mathbf{r}}[n+1] = \frac{7}{8}\tilde{\mathbf{r}}[n] + \frac{1}{8}\mathbf{r}[n]. \tag{64}$$

The TCP sender also keeps track of the variation in the observed RTT values, which is computed by

$$\sigma_{RTT}[n+1] = \frac{3}{4}\sigma_{RTT}[n] + \frac{1}{4} \times |\mathbf{r}[n] - \tilde{\mathbf{r}}[n+1]|. \tag{65}$$

Subsequently, the value of RTO is set by using

$$\text{RTO} = \max\Big(RTO_{min}, \tilde{\mathbf{r}}[n+1] + \max\big(G, k \times \sigma_{RTT}[n+1]\big)\Big), \tag{66}$$

where $k = 4$ and G is the clock granularity (The default value for G is 500 ms in the BSD implementation). In practice, RTO_{min} is set to one second [141] to reduce spurious timeouts. In addition, current TCP variants implement Karn's algorithm [135], which suggests doubling the RTO value when a timeout occurs. Employing exponential timer backoff as well as adopting a large RTO_{min} are essential for TCP's congestion control algorithm and network-friendliness. However, such measures are naturally too costly for delay-sensitive applications. Therefore, in our comparisons, we use an *enhanced TCP-like*

RTO estimator, where RTO_{min} is set to zero, and the exponential timer backoff is disabled. In addition, in our simulations and experiments, we use a clock granularity of 10 ms to further improve the RTO estimation performance [141].

6.2.2 Recursive Weighted Median Filtering

Recursive weighted median (RWM) filtering was recently proposed by Ma *et al.* to improve the TCP's RTO estimation method [146]. Due to its nature, a weighted median filter is less susceptible to rapid fluctuations in the observed data and can provide a better model for the signals showing impulsive statistics as compared to (64). The basic idea in RWM filtering is to compute the RTT estimate by taking the weighted median of the last K estimates and last J observations. That is,

$$\tilde{\mathbf{r}}[n] = \mathrm{WM}\left(\left[\tilde{\mathbf{r}}[n-k]|_{k=1}^{K}, \mathbf{r}[n-j]|_{j=1}^{J}\right], \mathbf{W}\right), \tag{67}$$

where $\mathbf{W}$ is the weight vector. The study suggests the following values: $K = 1$, $J = 5$ and $\mathbf{W} = [\frac{1}{2}, (\frac{7}{8})^0, (\frac{7}{8})^1, (\frac{7}{8})^2, (\frac{7}{8})^3, (\frac{7}{8})^4]$. Once an RTT estimate is computed, the value of RTO is determined by scaling the RTT estimate, where the scale factor depends on the mean absolute deviation among the RTT samples. Improvements over Jacobson's algorithm are reported through Internet experiments [146].

6.2.3 Percentile-Based RTO Estimators

Empirical studies usually try to fit a well-known distribution to packet delays experienced in IP networks [7, 99, 101]. Such statistical models are particularly useful in laying out a mathematical framework for building application-layer protocols [52]. For example, [101] suggests that RTT values follow a shifted Gamma distribution, whereas [7, 99] show that packet delay distributions are heavy-tailed and can be characterized by a Pareto distribution. The pertaining distribution model parameters can be estimated from the collected samples by using maximum likelihood estimation. Alternatively, one can collect the delay samples and generate a distribution on the fly. Having a history of the delay samples, the delay of the next packet can be predicted by computing the p^{th}-percentile of the delay histogram, where p is selected depending on the maximum redundant retransmission rate tolerable by the application. The flowchart for the percentile-based RTO estimation is given in Figure 52.

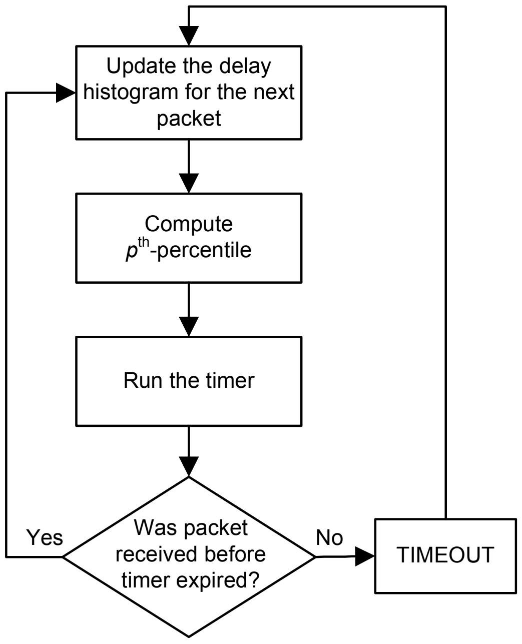

Update the delay histogram for the next packet
Compute p^{th}-percentile
Run the timer
Was packet received before timer expired?
Yes
No
TIMEOUT

Figure 52: Flowchart for the percentile-based RTO estimators.

6.3 Autoregressive Models for Packet Delay

Statistical models, briefly discussed in Section 6.2.3, are only useful in characterizing the general properties of packet delays, and fall short in describing the temporal dependence among packet delays. Previously, Jiang and Schulzrinne investigated the conditional delay distributions [99], and found a significant correlation between the adjacent delay samples. Temporal dependence in packet delay was also reported by Kalman and Girod [149]. In this section, we exploit the packet delay correlation through autoregressive models and introduce the concept of model selection. Before we start our discussion, we first provide an insight for the client-driven RTO estimation.

6.3.1 System Overview

Consider a low-delay video application where the client runs an RTO estimation method to determine the best time to request a retransmission for a missing packet. Upon receiving a packet, the client measures its delay and predicts the delay for the subsequent packet.

The client then computes the amount of additional waiting to take into account the delay variability[2]. In the implementation, we identify each packet with a unique number, which can be associated with the Sequence Number field in the RTP header [102]. Since a retransmission cannot be distinguished from the initial transmission, the delay measurements for the retransmitted packets are ignored to avoid any ambiguity.

Our Internet experiments suggest that the majority of the lost packets can be recovered with a single retransmission, and two or more retransmissions are rarely necessary. Furthermore, the low-delay requirement of our target applications severely limits the possibility of multiple retransmissions. Thus, our discussion focuses only on the single-retransmission case.

6.3.2 Adaptive Linear Delay Prediction

Let us consider a stochastic process $\mathbf{s}$ and let $\mathbf{s}[n-k]$, $k \geq 1$ denote the past samples of this process. The operation of linear prediction expresses the value of $\mathbf{s}[n]$ as the linear combination of the samples $\mathbf{s}[n-k]$. The estimate based on the N most recent values is given by

$$\tilde{\mathbf{s}}_N[n] = E\left\{\mathbf{s}[n] | \mathbf{s}[n-k], 1 \leq k \leq N\right\} = \sum_{k=1}^{N} \alpha_{k,N}\mathbf{s}[n-k]. \tag{68}$$

This estimate is called the one-step forward predictor of order N. The process $\tilde{\mathbf{s}}_N[n]$ is the response of the forward predictor filter

$$\mathbf{H}_N(z) = \sum_{k=1}^{N} \alpha_{k,N} z^{-k} \tag{69}$$

to the input $\mathbf{s}[n]$. Our objective in prediction is to determine the constants $\alpha_{k,N}$ so as to minimize the mean square value

$$P_N = E\left\{\boldsymbol{\epsilon}_N^2[n]\right\} \tag{70}$$

of the forward prediction error $\boldsymbol{\epsilon}_N[n] = \mathbf{s}[n] - \tilde{\mathbf{s}}_N[n]$. From the orthogonality principle, we know that the prediction error, *i.e.*, $\boldsymbol{\epsilon}_N[n]$, is orthogonal to all data used to generate the

[2]Note that the methods summarized in Section 6.2 were originally designed for server-side RTO estimation. In our comparisons (Sections 6.3.5 and 6.5), we adapt them for use at the client side.

prediction, *i.e.*, $\mathbf{s}[n-m]$, where $1 \leq m \leq N$. Mathematically, we have

$$E\left\{\left(\mathbf{s}[n]-\sum_{k=1}^{N}\alpha_{k,N}\mathbf{s}[n-k]\right)\mathbf{s}[n-m]\right\}=0 \quad 1 \leq m \leq N, \tag{71}$$

which yields a set of linear equations known as the Yule-Walker equations. The coefficients $\alpha_{k,N}$ of the predictor filter $\mathbf{H}_N(z)$ can be computed from

$$R[m]-\sum_{k=1}^{N}\alpha_{k,N}R[m-k]=0 \quad 1 \leq m \leq N, \tag{72}$$

where $R[q]$ represents the lag-q autocorrelation of $\mathbf{s}$. The resulting mean-squared prediction error equals

$$P_N = R[0]-\sum_{k=1}^{N}\alpha_{k,N}R[k]. \tag{73}$$

Solving the linear system given in (72) is difficult for large N when the predictor filter $\mathbf{H}_N(z)$ is realized in a ladder, *i.e.*, transversal, structure. An alternative realization is to use a lattice structure. In addition to simplifying the solution of (72), the lattice realization also has the following advantage: Suppose that we have a predictor of order N, and we want to find the predictor of order $N+1$. In the ladder realization, a new set of $N+1$ filter coefficients ($\alpha_{k,N+1}$) has to be computed from scratch. In contrast, in the lattice realization, only the new partial autocorrelation coefficient (ϕ_{N+1}) has to be computed; the first N partial autocorrelation coefficients (ϕ_k) remain unchanged.

In the lattice realization, the predictor filter coefficients can be easily computed by the Durbin-Levinson recursion [150]. Starting with $\alpha_{1,1}=\phi_1$, we recursively express the filter coefficients in terms of the partial autocorrelation coefficients. Initially, for $N=1$, we have

$$\alpha_{1,1}=\phi_1=\frac{R[1]}{R[0]} \quad \text{and} \quad P_1=(1-\phi_1^2)R[0]. \tag{74}$$

By using the relation

$$P_{N-1}\phi_N = R[N]-\sum_{k=1}^{N-1}\alpha_{k,N-1}R[N-k], \tag{75}$$

we compute ϕ_N in terms of the known parameters $\alpha_{k,N-1}$, $R[m]$ and P_{N-1}, and then plug ϕ_N into

$$P_N=(1-\phi_N^2)P_{N-1} \tag{76}$$

to compute P_N. The predictor filter coefficients are recursively computed from

$$\alpha_{N,N} = \phi_N \tag{77}$$

and

$$\alpha_{k,N} = \alpha_{k,N-1} - \phi_N \alpha_{N-k,N-1} \quad 1 \leq k \leq N-1. \tag{78}$$

As the order of prediction increases, the value of the mean prediction-error power decreases or else remains the same. Since prediction-error power is always positive, we have

$$P_1 \geq P_2 \geq \ldots \geq P_N \xrightarrow[N\to\infty]{} P \geq 0. \tag{79}$$

The implication of (79) is that as we increase the order of the predictor filter $\mathbf{H}_N(z)$, we successively reduce the correlation between the adjacent samples of the input process until we ultimately reach a point at which increasing the order of prediction any further does not reduce the prediction-error power. At this point, the error is a white noise process and consists of purely uncorrelated samples[3].

Suppose that $P_{M-1} > P_M$ and $P_M = P_{M+1} = \ldots = P$. By definition, the process $\mathbf{s}$ is called an M^{th}-order autoregressive, denoted by AR(M), process or a wide-sense Markoff process of order M. For this process, the M^{th}-order predictor, $\tilde{\mathbf{s}}_M[n]$, is equivalent to its Wiener predictor:

$$E\left\{\mathbf{s}[n] | \mathbf{s}[n-k], 1 \leq k \leq M\right\} = E\left\{\mathbf{s}[n] | \mathbf{s}[n-k], k \geq 1\right\}. \tag{80}$$

Wiener predictors produce the best fit to the observed data by exploiting the existing correlation completely. However, due to their high complexity and low predictive accuracy, Wiener predictors are usually not used in practice. In the next section, we discuss the issue of prediction model selection by examining different delay traces.

6.3.3 Model Selection

Generally, it is desirable to have the values predicted by a model to be close to the actual data values. As pointed out by (79), increasing the order of prediction naturally produces

[3]Linear models cannot be used to predict a process with uncorrelated samples. However, one can use non-linear models such as Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models to exploit any remaining dependency until the error samples are independent of each other.

better estimates and a lower prediction-error power. However, an overfitted model may not distinguish the systematic effects of the data from its random effects. For practical purposes, we seek a model that yields a high predictive accuracy with the smallest number of parameters.

A popular model selection method is the Akaike's Information Corrected Criterion (AICC) [150, 151]. The AICC score quantifies the relative goodness-of-fit of a statistical model for the given data. A penalty factor is added to the negative log-likelihood of the fitted model for each parameter in order to prevent overfitting. For an AR(M) process, the AICC score is given by

$$\text{AICC} = -2 \log L(\tilde{\mathbf{s}}) + \frac{2n \times (M+1)}{n - M - 2}, \tag{81}$$

where n is the sample size and $L(\tilde{\mathbf{s}})$ is the likelihood of the n observations. The absolute AICC scores include an arbitrary constant, thus, are of no direct use. However, a lower AICC score indicates a better prediction model.

Let us illustrate the importance of model selection on three packet delay traces. To generate these traces, we simulated a moderate-sized Internet topology [104] in *ns*-2 network simulator [134] and used video streams that were encoded by a standard H.264 codec [120] at 300 Kbps, 600 Kbps and 1.2 Mbps. We refer to these delay traces with the notation of $\Delta T = 40$ ms, $\Delta T = 20$ ms and $\Delta T = 10$ ms, respectively, where ΔT denotes the average transmission interval at the server.

First, we examine the relation between the mean prediction-error power and the order of prediction in Figure 53. Based on the definition given in (80), the order of Wiener prediction for the $\Delta T = 10$ ms, $\Delta T = 20$ ms and $\Delta T = 40$ ms traces is found to be 12, 32 and 60, respectively. Clearly, we require a higher order of prediction for larger ΔT. This is not surprising since the correlation between the adjacent delay samples reduces with ΔT. This result can also be explained by the partial autocorrelation functions (PACF) plotted in Figure 54, where we note that it takes a larger number of lags for the PACF to die off as the value of ΔT increases.

Figure 53 shows that the mean prediction-error power gradually decreases with the order of prediction. However, the AICC scores first show a decreasing and then an increasing trend (See Figure 55). In other words, the predictive accuracy improves with increasing N until a point and then starts degrading. Specifically, the AICC scores for the $\Delta T = 10$ ms, $\Delta T = 20$ ms and $\Delta T = 40$ ms traces reach their global minima at $N = 3$, $N = 9$ and $N = 12$, respectively. These values are comparably smaller than the ones corresponding to the Wiener prediction, signifying that Wiener predictors are indeed overfitted and have sub-optimal predictive accuracy.

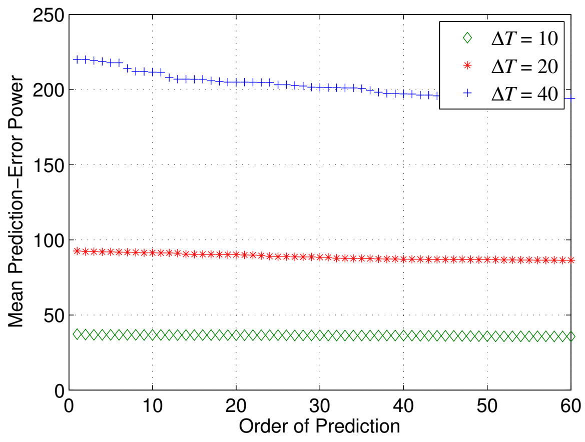


Figure 53: Variation of the mean prediction-error power with the order of prediction.

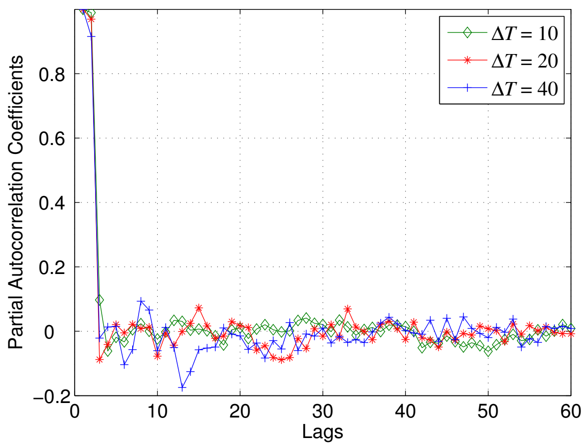


Figure 54: PACFs for all three delay traces.

6.3.4 Practical Considerations

Generally speaking, the AICC method suggests good models that provide sufficient insight into the process being analyzed, while leaving out the random effects. In non-time-critical

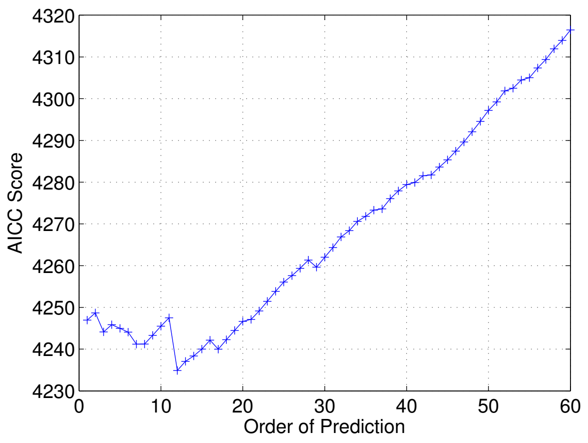


Figure 55: Variation of the AICC score with the order of prediction for $\Delta T = 40$ ms.

tasks, the computational complexity is of a less important issue. Thus, the models suggested by the AICC approach can be facilely employed without hampering the performance of the system. However, if the prediction is carried out in real time, low-complexity models have to be used to sustain the system feasibility. The main objective is, thus, to select a computationally-efficient yet intuitively-plausible prediction model that adequately captures the dynamics in the packet delay process.

A naive approach is the AR(1) model, where the next delay estimate is solely determined by the last observation, *i.e.*, $\tilde{\mathbf{s}}_1[n] = \mathbf{s}[n-1]$. The phase diagrams plotted in Figure 56 clearly indicate the existence of a significant lag-1 correlation among the delay samples and support the AR(1) prediction model. However, this predictor is not capable of distinguishing whether packet delays are increasing, decreasing, or remaining the same, and therefore, does not serve our goal well.

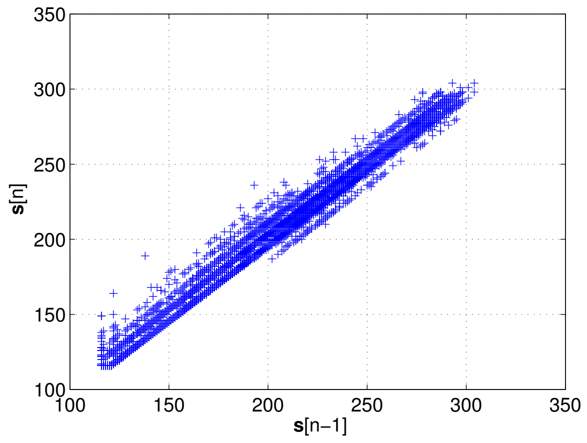
A more elaborate model is the AR(2) model. AR(2) model bases its estimation on the last two observations. By definition, we have

$$\tilde{\mathbf{s}}_2[n] = \alpha_{1,2}\mathbf{s}[n-1] + \alpha_{2,2}\mathbf{s}[n-2], \tag{82}$$

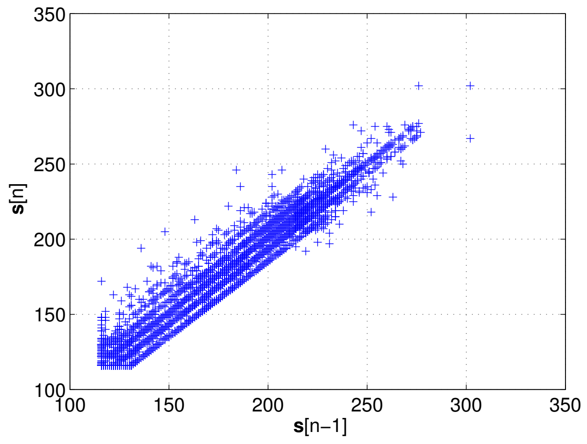
which can be rewritten as

$$\tilde{\mathbf{s}}_2[n] = (\alpha_{1,2} + \alpha_{2,2})\mathbf{s}[n-1] + \alpha_{2,2}\left(\Delta T - \Delta t[n-1]\right), \tag{83}$$

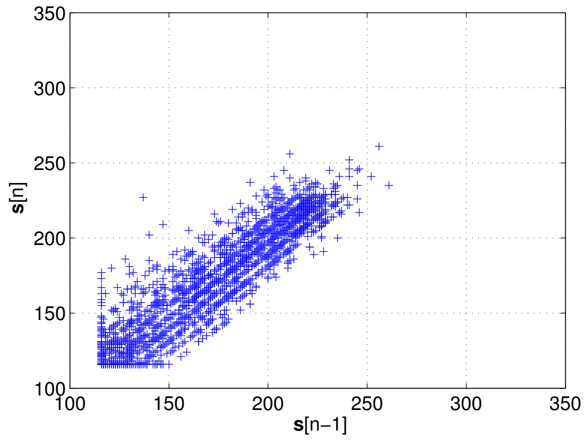
where $\Delta t[n]$ denotes the interarrival time for packet n, *i.e.*, the time difference between the arrivals of packets n and $n-1$. The interpretation of (83) is that the AR(2) model takes



(a)



(b)



(c)

Figure 56: Phase diagrams for $\Delta T = 10$ ms (a), $\Delta T = 20$ ms (b) and $\Delta T = 40$ ms (c).

into consideration not only the last delay sample but also its deviation from the previous sample.

To understand how well an AR(2) predictor compares to its Wiener counterpart, we plot the prediction-error autocorrelation functions (ACF) for both predictors. Since Wiener predictors completely model the data, the resulting error samples are guaranteed to be uncorrelated, which is, however, not necessarily true for AR predictors of lower orders. Nevertheless, Figure 57 shows that the correlation left out by the AR(2) predictors is rather insignificant, implying that AR(2) predictors have sufficient predictive accuracy for practical purposes.

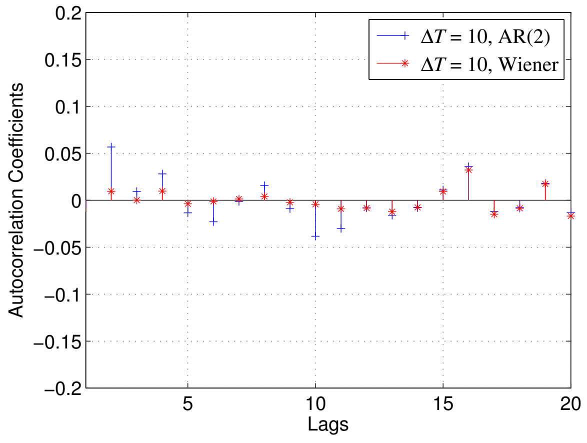
6.3.5 Performance Analysis

In this section, we evaluate the performance of different delay predictors. We analyze (*i*) the exponentially-weighted moving average (EWMA) approach, defined in (64), (*ii*) recursive weighed median (RWM) filtering, defined in (67), (*iii*) Wiener prediction, (*iv*) AICC-based prediction, and (*v*) AR(2) prediction. In Table 8, we observe that the prediction-error standard deviations produced by the AR predictors are substantially smaller than those produced by the EWMA approach and RWM filtering. In particular, the AR(2) predictors can achieve up to 50% and 39% reduction in the prediction-error standard deviation compared to the EWMA approach and RWM filtering, respectively.

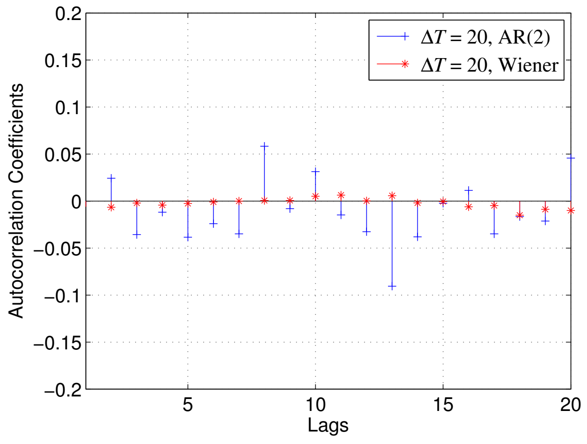
Table 8: Comparison of the prediction-error standard deviations produced by different delay predictors.

	EWMA Approach	RWM Filtering	Wiener Prediction	AICC-based Prediction	AR(2) Prediction
$\Delta T = 10$ ms	11.6 ms	9.0 ms	6.0 ms	6.0 ms	6.0 ms
$\Delta T = 20$ ms	19.4 ms	15.7 ms	9.3 ms	9.5 ms	9.6 ms
$\Delta T = 40$ ms	26.1 ms	21.8 ms	13.9 ms	14.4 ms	14.8 ms

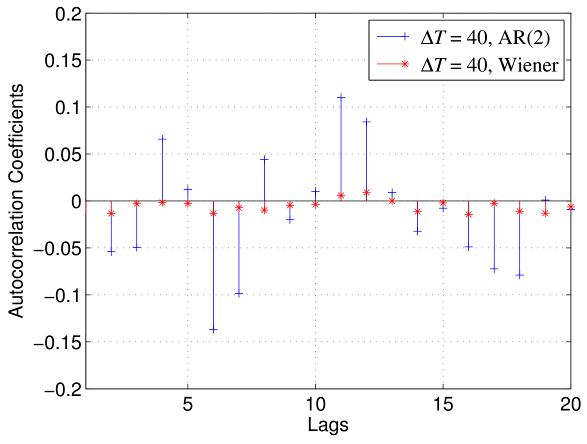
In Figure 58, we compare three different delay predictors in time domain for the $\Delta T =$ 40 ms trace. We observe that the AR(2) predictor outperforms its rivals in terms of tracking the actual delay trace more closely. We also observe that the EWMA approach largely fails in detecting the delay spikes, mainly because EWMA naturally smooths out the outliers.



(a)



(b)



(c)

Figure 57: ACFs of the prediction errors produced by Wiener and AR(2) predictors for $\Delta T = 10$ ms (a), $\Delta T = 20$ ms (b) and $\Delta T = 40$ ms (c).

In Figure 58, we did not include the AR(60) and AR(12) predictors for comparison, since according to Table 8, the AR(2) predictors perform closely to their Wiener counterparts as well as the predictors suggested by the AICC method.

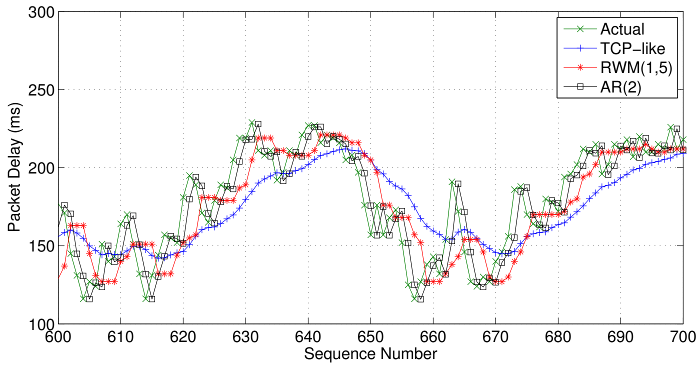


Figure 58: Time-domain comparison of different predictors for $\Delta T = 40$ ms.

6.4 Media-Unaware Timeout Estimation

From the point of view of (70), an underestimate that is marginally smaller than the actual value is as good as an overestimate that is marginally larger than the actual value. However, in the context of RTO estimation, underestimations trigger pre-mature timeouts whereas overestimations eliminate them. In this section, we formulate a computationally-efficient way to compute the minimum amount of additional waiting that is required to keep the probability of timing out pre-maturely below a desired value. In this media-unaware approach, the timeout decisions are given purely based on the observed delay samples. Later, in Chapter 7, we will study a media-aware approach where the timeout computation is carried out based on packet-specific information.

In Figure 59, we plot the prediction-error distributions for the $\Delta T = 10$ ms and $\Delta T = 40$ ms traces. We notice that each of these distributions (particularly, the tail parts) can be approximated by a Gaussian distribution whose mean and standard deviation are equal to those of the corresponding prediction-error distribution. Statistically, Gaussian-distributed samples of a white noise process are independent of each other. In the light of Figure 57, we infer that AR(2) predictors produce error samples that are independent. This result has two important implications: First, a sequence of independent random variables is not predictable by linear or non-linear models. Thus, if packet delay sampling is sufficiently dense, the delay process can be almost completely characterized by an AR(2) model. Second, Gaussian-distributed processes are easy to work with, and a rich set of mathematical tools is available.

Let τ denote the additional amount of waiting to be added to the initial delay predicted by (83), and let $\Phi(\tau)$ denote the delay underestimation probability. By definition,

$$\Phi(\tau) = P\left\{\tilde{\mathbf{s}}_2[n] + \tau < \mathbf{s}[n]\right\}, \tag{84}$$

which is a non-increasing function of τ. We seek the minimum value for τ that satisfies

$$\Phi(\tau) \le p_f, \tag{85}$$

where p_f is the desired probability of timing out pre-maturely. By rewriting $\Phi(\tau)$ as $P\{\tau < \boldsymbol{\epsilon}_2\}$, we compute τ from

$$\tau = F_{\boldsymbol{\epsilon}_2}^{-1}(1 - p_f), \tag{86}$$

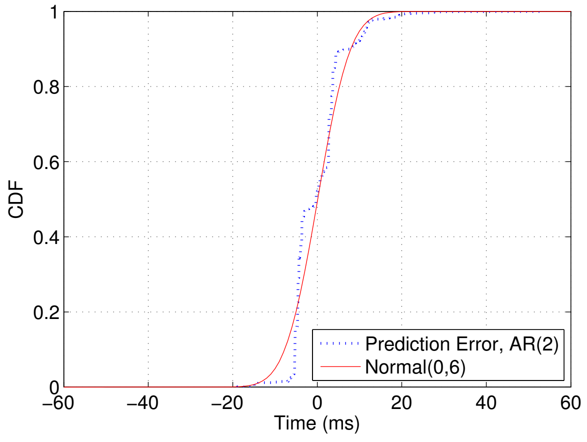
where $F_{\boldsymbol{\epsilon}_2}$ is the cumulative density function of $\boldsymbol{\epsilon}_2$. A nice feature of the Gaussian distribution is that its inverse cumulative function can be directly calculated from the first and second-order moments[4]. For example, to limit the rate of pre-mature timeouts to 5%, τ should be set to 1.65 times the standard deviation, which is $1.65 \times 14.8 = 25$ ms for the $\Delta T = 40$ ms trace. While 25 ms may seem insignificant, τ quickly increases for lower p_f values, *e.g.*, for $p_f = 0.1\%$, the required amount increases to 46 ms.

The adverse impact of a large τ is the increase in the time required to detect lost packets. To quantify the detection time of a lost packet, we use the delay of the last successfully-received packet as the hypothetical delay for the lost packet. The loss detection time is then given by the difference between the predicted and hypothetical delays. That is,

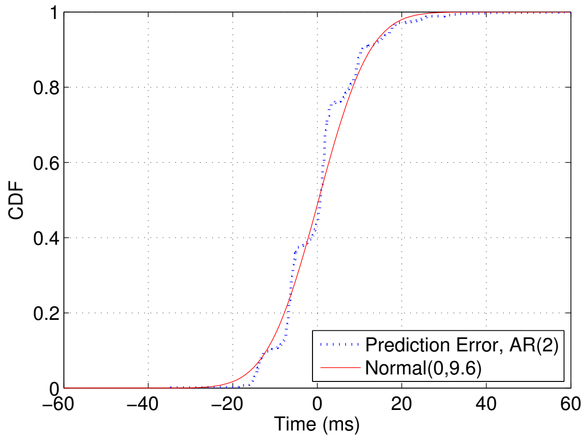
$$w[n] = \tilde{\mathbf{s}}_2[n] + \tau - \mathbf{s}[n^*], \quad \forall n : \mathbf{s}[n] = \infty, \tag{87}$$

where n^* is the last successfully-received packet. The average loss detection time and the

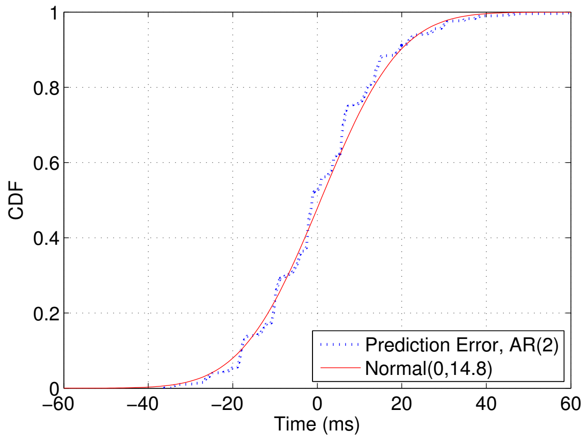
[4]Note that the mean of the prediction error produced by the AR(2) predictor is observed to be close to zero in all traces. Thus, we merely require the sample standard variation to compute the optimal timeout duration from (86).



(a)



(b)



(c)

Figure 59: Prediction-error distributions for $\Delta T = 10$ ms (a), $\Delta T = 20$ ms (b) and $\Delta T = 40$ ms (c). Plots do not include the lost packets.

pre-mature timeout probability are the benchmarks that characterize the performance of an RTO estimator.

6.5 Simulation Results

In this section, we present several *ns*-2 simulation results and evaluate the performances of four different RTO estimators: (*i*) the enhanced TCP-like RTO estimator, denoted by $\text{RTO}_{\text{E-TCP}}$, (*ii*) recursive weighted median filtering, denoted by $\text{RTO}_{\text{RWM}(1,5)}$, (*iii*) a percentile-based RTO estimator that predicts the forward-trip time (FTT) of the next expected packet by computing the p^{th}-percentile of the FTT histogram (excluding the lost packets), denoted by RTO_{PRC}, and (*iv*) the media-unaware RTO estimator, denoted by $\text{RTO}_{\text{AR}(2)}$. Before comparing all four RTO estimators, we first examine them individually.

We start our analysis with the enhanced TCP-like RTO estimator. $\text{RTO}_{\text{E-TCP}}$ runs (64)-(66) on the FTT samples to compute the timeout value[5]. Recall that we set RTO_{min} to zero and disable the exponential timer backoff to avoid excessive overwaiting. Here, by changing the value of k in (66) we characterize the behavior of $\text{RTO}_{\text{E-TCP}}(k)$. The left plot in Figure 60 shows that $\text{RTO}_{\text{E-TCP}}(k)$ suffers from frequent spurious timeouts when $k \leq 3$, mainly because of the slow convergence of (66). As k increases, $k = 4$ is the suggested value for TCP [141], (66) becomes less susceptible to delay variations, and hence, more successful in RTO estimation. However, the right plot in Figure 60 shows that while the average loss detection time increases linearly with k, the 95^{th}-percentile of the loss detection times increases faster. This implies that some of the lost packets are identified in significantly longer times when $k > 3$.

The second RTO estimator is the recursive weighted median (RWM) filtering. Due to the large number of parameters associated with this estimator (K, J and $\mathbf{W}$ in (67)), we omit a detailed analysis and simply adopt the values suggested by [146]. Table 9 shows that $\text{RTO}_{\text{RWM}(1,5)}$ is successful in keeping the estimation failure rate small, particularly for the

[5]Generally speaking, RTT samples show a larger variation due to the variations experienced in both forward and backward-trip times. Hence, using RTT samples in (65) results in overinflated timeout estimates. In order to remove this adverse effect, $\text{RTO}_{\text{E-TCP}}$ uses FTT samples instead of RTT samples for RTO estimation.

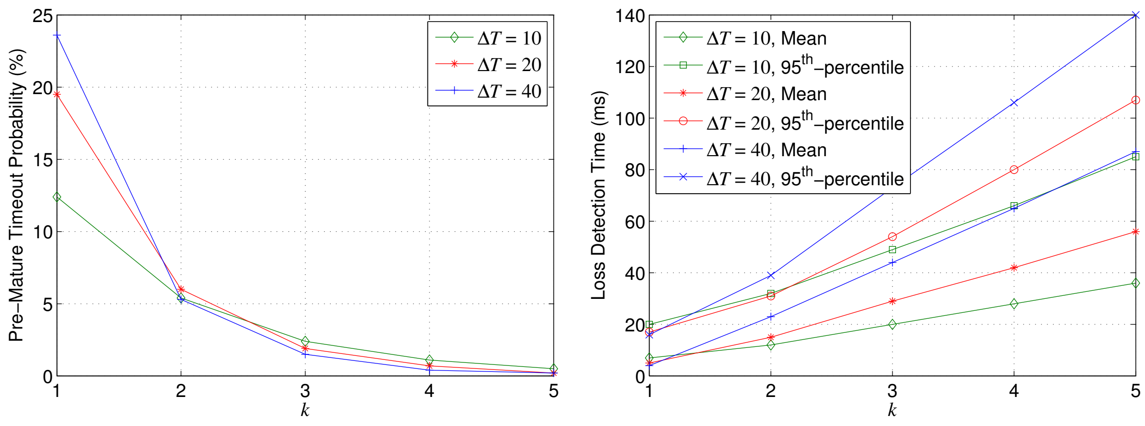


Figure 60: Performance analysis of $\mathrm{RTO}_{\mathrm{E-TCP}}(k)$.

$\Delta T = 10$ ms trace. In their study [146], the authors reported that RWM filtering performed better than Jacobson's algorithm with TCP flows. However, our simulations show that $\mathrm{RTO}_{\mathrm{RWM}(1,5)}$ detects lost packets in a substantially longer amount of time compared to $\mathrm{RTO}_{\mathrm{E-TCP}}$. While this seems to be a contradictory result, we must emphasize that RWM filtering approach was designed for TCP, and hence, does not necessarily have to perform well with non-TCP flows.

Table 9: Performance analysis of $\mathrm{RTO}_{\mathrm{RWM}(1,5)}$.

	p_f	Mean w	95th-percentile of w
$\Delta T = 10$ ms	0.2%	158 ms	199 ms
$\Delta T = 20$ ms	0.3%	156 ms	199 ms
$\Delta T = 40$ ms	0.6%	140 ms	185 ms

Third, we present the results for the percentile-based RTO estimator. In Figure 61, we observe that the estimation failure probability of $\mathrm{RTO}_{\mathrm{PRC}}(p)$ decreases linearly with parameter p. However, the incurred cost, *i.e.*, the increase in the loss detection time, escalates rather quickly with p. Moreover, unlike other RTO estimators, $\mathrm{RTO}_{\mathrm{PRC}}$ performs better with lower-bitrate streams. We conjecture that this is mainly due to the fact that at low bitrates the temporal dependence in packet delay is smaller, and the assumption of uncorrelated delay samples holds more often, making $\mathrm{RTO}_{\mathrm{PRC}}$ more robust and successful in RTO estimation.

Finally, we test the media-unaware RTO estimator with the same three delay traces.

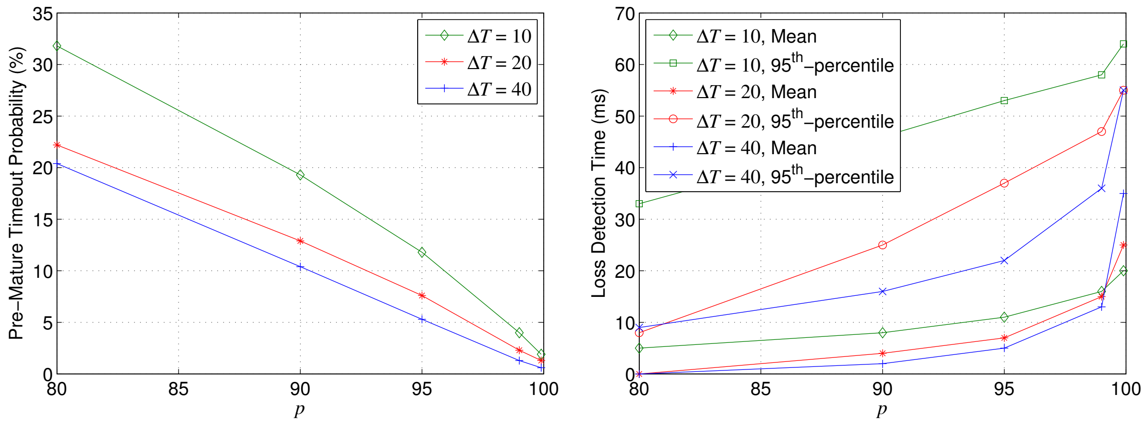


Figure 61: Performance analysis of $\mathrm{RTO}_{\mathrm{PRC}}(p)$.

Figure 62 reveals two important findings: First, we notice that the client detects lost packets faster when streaming at higher bitrates. This result is consistent with Figures 53 and 57 where we showed that the AR(2) prediction is more accurate for smaller ΔT values. Second, we observe that the mean and the 95th-percentile of the loss detection times are close to each other in each trace, suggesting that the loss detection time does not vary much over time.

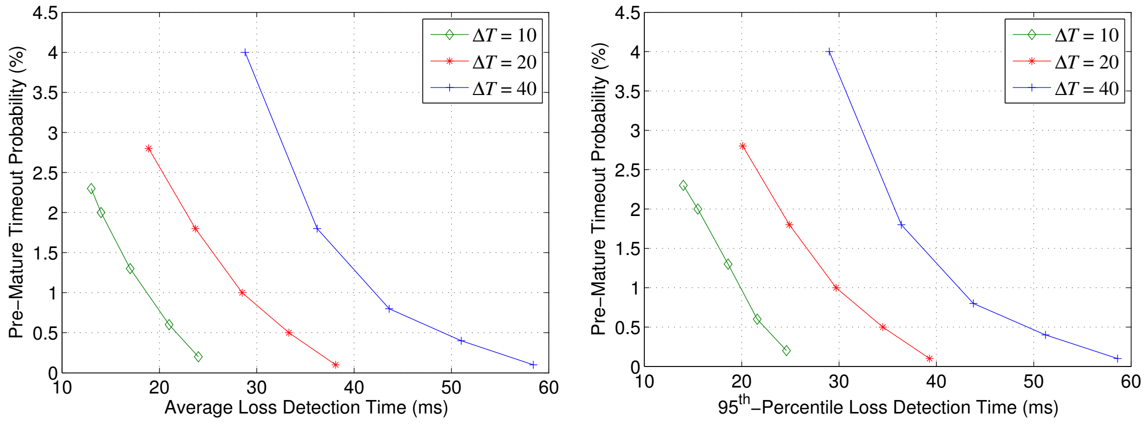


Figure 62: Performance analysis of $\mathrm{RTO}_{\mathrm{AR}(2)}$.

Now we compare $\mathrm{RTO}_{\mathrm{E-TCP}}$, $\mathrm{RTO}_{\mathrm{PRC}}$ and $\mathrm{RTO}_{\mathrm{AR}(2)}$ on the $p_f - w$ plane. Since the loss-detection performance of $\mathrm{RTO}_{\mathrm{RWM}(1,5)}$ is the worst by a large margin, we omit it from this comparison. Here, we are interested in determining which RTO estimator detects the lost packets in the shortest amount of time without exceeding a given pre-mature timeout probability. Figure 63 shows that $\mathrm{RTO}_{\mathrm{AR}(2)}$ substantially outperforms $\mathrm{RTO}_{\mathrm{E-TCP}}$ in all cases. For the $\Delta T = 10$ ms and $\Delta T = 20$ ms traces, $\text{RTO}_{\text{AR(2)}}$ also achieves a better performance than RTO_{PRC}. However, in the $\Delta T = 40$ ms trace, RTO_{PRC} detects the lost packets 8 - 20 ms faster than $\text{RTO}_{\text{AR(2)}}$ at regions where $p_f > 0.6\%$. Nevertheless, at the expense of a 20 ms increase in the average loss detection time, $\text{RTO}_{\text{AR(2)}}$ is able to diminish the pre-mature timeout rate to 0.1%.

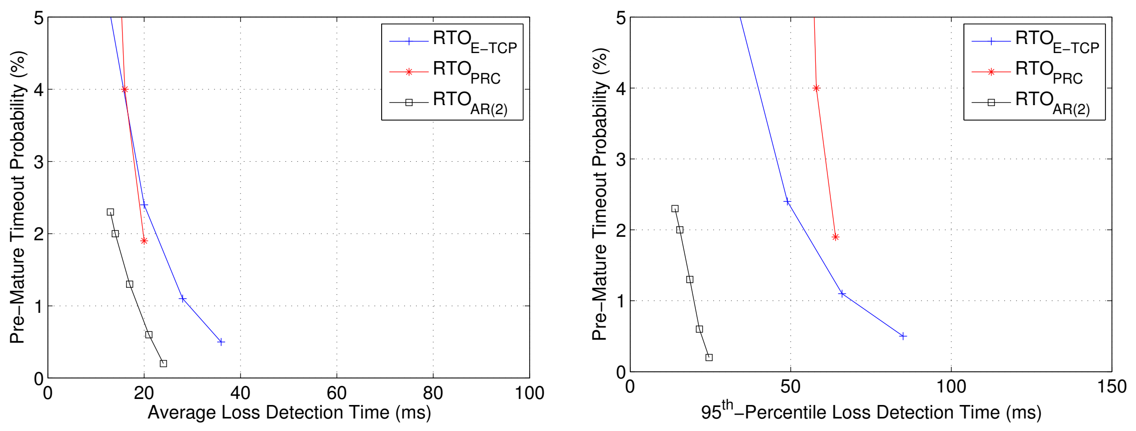
One important issue in RTO estimation is the rapid convergence of the timeout estimates. Based on our simulations, we found that $\text{RTO}_{\text{E}-\text{TCP}}$ required at least 15 - 20 samples to produce good estimates. Thus, when the network conditions changed rapidly, $\text{RTO}_{\text{E}-\text{TCP}}$ largely failed. This problem was solved to some extent by $\text{RTO}_{\text{RWM(1,5)}}$, which only required five samples to produce an estimate. On the other hand, RTO_{PRC} initially required several samples to be able to work properly. In contrast, $\text{RTO}_{\text{AR(2)}}$ required only the last two samples for RTO estimation. This fast-convergence feature provides $\text{RTO}_{\text{AR(2)}}$ robustness when the packets continuously experience a large amount of jitter, or when only a small number of delay samples are available for RTO estimation.

6.6 Conclusions

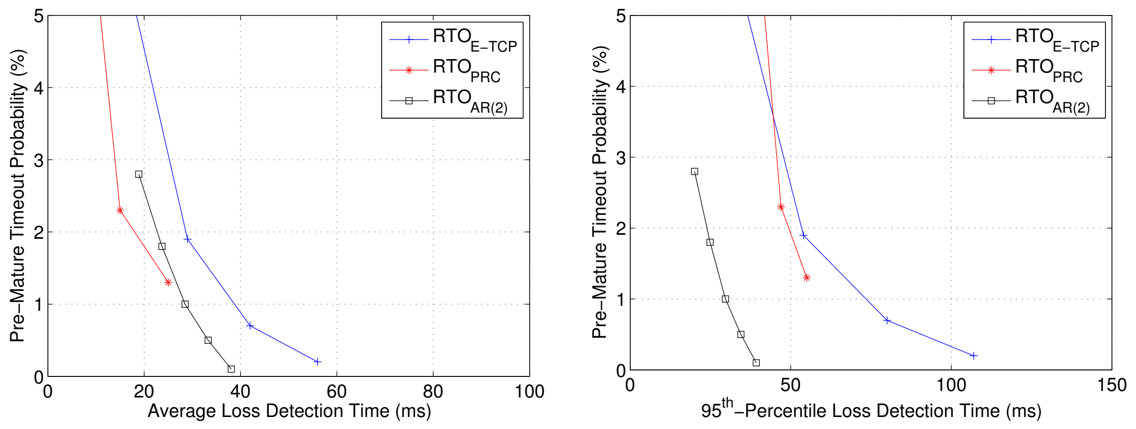
In this study, we developed an adaptive RTO estimation method for low-delay video applications. By exploiting the temporal dependence in packet delay, we achieved the optimal trade-off between the amount of overwaiting and redundant retransmission rate. With simulation results, we showed that our approach detected lost packets faster and more accurately compared to other RTO estimators. Our findings can be summarized as follows:

- RTO estimation is an essential component for any error-control/protection method. A good RTO estimator should be able to quickly identify lost packets under rigid delay requirements.

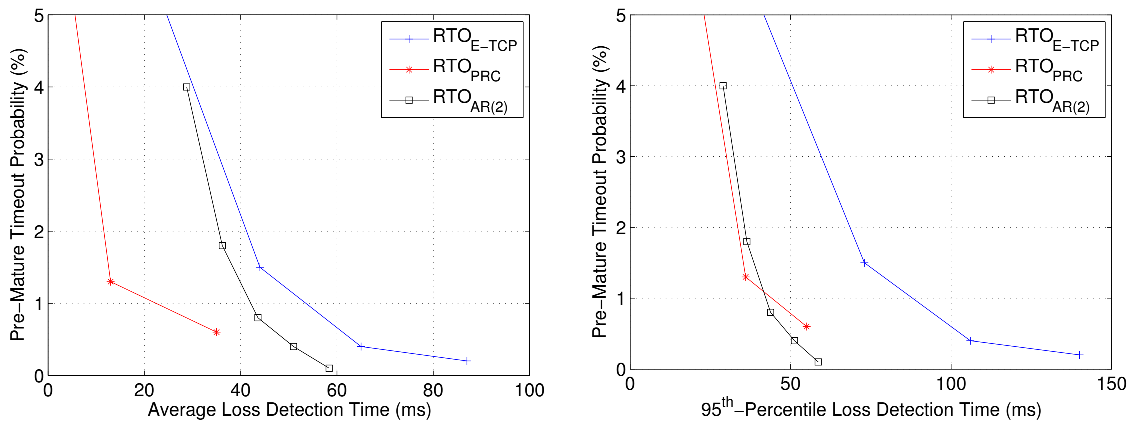
- The RTO estimators that are developed for TCP severely suffer from long loss detection times. We were able to reduce the detection times by tweaking the estimation parameters, *e.g.*, k in (66) and $\mathbf{W}$ in (67). However, the resulting rate of spurious timeouts was unacceptably high. Percentile-based RTO estimators deliver higher



(a)



(b)



(c)

Figure 63: Comparison of different RTO estimators for $\Delta T = 10$ ms (a), $\Delta T = 20$ ms (b) and $\Delta T = 40$ ms (c).

quality of video compared to the TCP-oriented RTO estimators at the expense of increased redundant retransmissions.

- Provided that the packets are transmitted at sufficiently short intervals, consecutive delay samples show a strong correlation. Wiener predictors can be used to fully exploit this correlation and produce uncorrelated prediction-error samples. We showed that these uncorrelated error samples could be modeled by a Gaussian distribution, implying that the error samples were indeed independent. Thus, Wiener prediction models can completely characterize the packet delay process. We also showed that AR(2) predictors could be safely used in practice instead of their Wiener counterparts.
- Adaptivity to time-varying network conditions is the key in successful RTO estimation. Slow adaptation potentially leads to a significant performance degradation in terms of redundant and late retransmissions.

CHAPTER VII

MEDIA-AWARE RETRANSMISSION TIMEOUT ESTIMATION FOR LOW-DELAY VIDEO APPLICATIONS

Developing error-control and error-resiliency methods for transmitting delay-sensitive media content over the best-effort networks poses several challenges. Due to the lack of QoS guarantees in the conventional Internet as well as in emerging wireless networks, these methods must continuously monitor the characteristics of the underlying network and try to infer the incipient network conditions so that they can take the necessary actions on time. This is utmost important for enhancing the end-user quality, particularly in low-delay multimedia applications. In this study, we tackle this problem from an error-control method perspective and develop an innovative framework that optimizes the retransmission decisions based on the urgency and importance of the media packets.

7.1 Introduction

In Chapter 6, we developed an adaptive retransmission timeout (RTO) estimation method for low-delay Internet video applications. This method consisted of two main steps: (*i*) delay prediction, and (*ii*) delay-boundary prediction. In the first step, we exploited the temporal dependence among the packet delay samples and used an adaptive linear delay predictor to produce the best estimate in terms of the mean-squared error criterion. This predictor computed the required predictor filter coefficients on the fly, and did not use any fixed coefficients. This way, we were able to carry out the delay prediction in an optimal fashion regardless of the source video transmission rate and time-varying network conditions. In the second step, on the other hand, we used a controller that optimally managed the trade-off between the amount of overwaiting and spurious retransmission rate by adjusting the bias to be added to the estimate produced in the first step. The goal was to compute the shortest timeout duration, and hence, to maximize the chance of on-time error recovery such that the redundant retransmission rate did not exceed a desired threshold. Our overall approach merely used the delay samples observed at the client side for timeout estimation.

In packetized video applications, however, timely delivery of a packet does not guarantee successful decoding. This is because many video coding standards, *e.g.*, MPEG-x and H.26x, use motion-compensated prediction to gain in coding efficiency at the expense of inducing a dependency structure among the encoded video frames. This dependency structure renders video frames unequally important. For example, a predicted frame can only be decoded after all the frames to which this particular predicted frame is referenced (called *ancestor* frames), are received and decoded. This implies that a frame missing during its decoding not only causes errors or a freeze during its display time, but also impedes the successful decoding of all frames that are dependent on it (called *descendant* frames). The resulting error propagation continues through all dependent frames and usually decays slowly. It is therefore essential to optimize the error control for each video packet/frame based on its importance.

In this study, we develop a *media-aware* RTO estimation method that computes the timeout estimates by jointly considering the interdependency relations and the decoding deadlines of video frames. Naturally, we should select a shorter timeout duration for packets belonging to more important and urgent frames than it is for packets belonging to less important and non-urgent frames. If retransmission capability is severely limited due to scarce bandwidth, we may even opt not to request a retransmission for less important packets and save the retransmission opportunities for more important packets. This prescient discrimination helps us achieve a higher rendering quality of video at the client side without any additional increase in the total transmission rate.

The architecture of the client-driven media-aware RTO estimation is sketched in Figure 64. Upon receiving a packet, the client measures its delay, and updates the necessary media-specific information and prediction-model parameters. The client then predicts the delay for the subsequent packet. This prediction does not use any media-specific information since the packet delay is totally independent of the payload. In contrast, the timeout duration for a packet is computed based on the importance of its payload, the time remaining to its decoding deadline as well as the redundant rate budget.

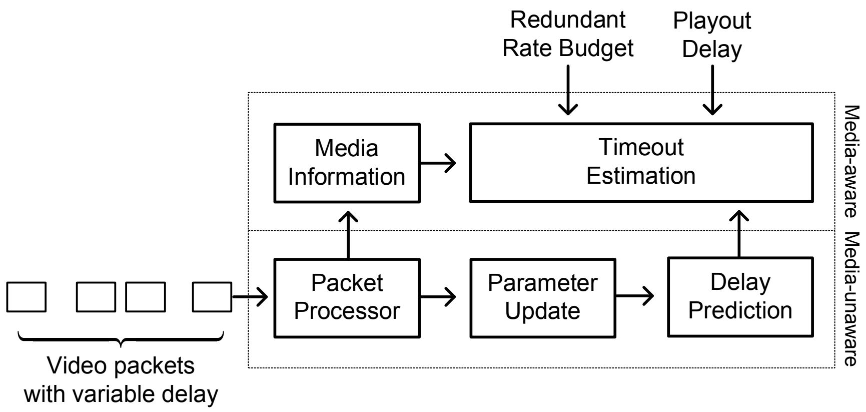


Figure 64: Architecture of the client-driven media-aware RTO estimation.

Recently, a large number of studies explored the problem of rate-distortion optimized media transmission in various contexts (See the references in [57]). Inspired by the work of Chou and Miao [52], these studies offered solutions to compute the optimal transmission and/or error-control policies by solving a Markov decision process (MDP) framework. However, in the interest of obtaining a manageable solution, the original MDP framework ignored the correlation between consecutive packet delay samples. Furthermore, the MDP framework also adopted the assumption of no dependency between the packet loss events and packet delays[1]. One can justify these assumptions for low-bitrate video transmission where the packets are transmitted at large intervals, and the delay/loss correlation between the packets is rather insignificant. However, these assumptions may not hold and substantially hinder the performance of the optimal policies when transmitting high-bitrate video[2].

The rest of the chapter is organized as follows: In Section 7.2, we formulate our optimization problem and derive a set of equations for media-aware RTO estimation. We discuss the solution approach and implementation issues in detail in Section 7.3. Results produced from simulations and Internet experiments are presented in Sections 7.4 and 7.5,

[1]We attempted to address this issue in Chapter 4 to some degree. In this chapter, we further investigate this issue and propose models that fully capture the delay-loss correlation.

[2]Note that some of the low-bitrate applications such as VoIP and online gaming also transmit packets frequently. In such applications, the delay/loss correlation between the consecutive packets can be significant and should not be ignored.

respectively. Finally, Section 7.6 concludes the chapter.

7.2 Problem Formulation

We solve the problem of media-aware RTO estimation within a finite-horizon optimization framework; at each decision epoch a set of frames are considered, and the optimal timeout durations are computed for each packet/frame. Let $\mathcal{G}$ denote a set of frames and assume that the frames within this set have well-defined interdependency relations that are known by the client. The critical step in media-aware RTO estimation is to develop an expression for evaluating the expected video quality of set $\mathcal{G}$ in terms of the packet decodability probabilities. As it will be clear shortly, the decodability of a packet depends on its on-time delivery probability, therefore, on the amount of its timeout duration, as well as the decodability of the packet(s) to which this packet is referenced.

In Chapter 4, we adopted a sophisticated video quality metric. This metric required the knowledge of per-packet distortion information, which could only be extracted during the encoding process. For our derivations in this chapter, we prefer a more practical and easy-to-work-with metric. To this effect, we quantify the video quality by the average rendered frame rate. By definition, the achieved frame rate for set $\mathcal{G}$ is computed by

$$\mathcal{Q}_{\mathcal{G}} = f_0 \times \frac{\eta_{\mathcal{G}}^{+}}{\eta_{\mathcal{G}}}, \tag{88}$$

where f_0, $\eta_{\mathcal{G}}^{+}$ and $\eta_{\mathcal{G}}$ are the original frame rate, the number of decodable frames and the total number of frames in set $\mathcal{G}$, respectively. Generally, a video frame is packetized into one or more equal-sized packets. Thus, without loss of generality, we assume that the decodable fraction of frame $\mathcal{F}_u$ is given by the ratio of the number of decodable packets in frame $\mathcal{F}_u$, denoted by υ_u^{+}, to the total number of packets in frame $\mathcal{F}_u$, denoted by υ_u. Hence, we have

$$\eta_{\mathcal{G}}^{+} = \sum_{u=1}^{\eta_{\mathcal{G}}} \frac{\upsilon_u^{+}}{\upsilon_u}. \tag{89}$$

Since f_0 and $\eta_{\mathcal{G}}$ are constants in (88), our goal is essentially to compute the optimal timeout duration for each packet in set $\mathcal{G}$ such that the expression in (89) is maximized while the expected redundant retransmission probability does not exceed the desired limit.

Our optimization problem can be formalized as follows:

Given: A set of frames, $\mathcal{G}$.

Objective: Find the optimal timeout for each packet in set $\mathcal{G}$.

$$\boldsymbol{\tau}_{opt} = \arg\max_{\boldsymbol{\tau}} \eta_{\mathcal{G}}^{+} \tag{90}$$

Subject to: Expected redundant retransmission probability stays within the required limit.

$$\frac{\sum_{u=1}^{\eta_{\mathcal{G}}} \sum_{n=1}^{\upsilon_u} \boldsymbol{p}_f[n]}{\sum_{u=1}^{\eta_{\mathcal{G}}} \upsilon_u} \leq p_f \tag{91}$$

Given a set of frames, the optimization problem defined in (90) and (91) requires the delay prediction for R future packets, where $R = \sum_{u=1}^{\eta_{\mathcal{G}}} \upsilon_u$. That is, if n^* denotes the last-successfully received packet, we need to predict the delays for packets n^*+1, n^*+2, ..., n^*+R. For this purpose, we use the multi-step version of the AR(2) predictor given in (82). The r-step AR(2) predictor is defined as follows:

$$\tilde{\mathbf{s}}_2^r[n] = E\left\{\mathbf{s}[n] | \mathbf{s}[n-k], r \leq k \leq r+1\right\} \quad 1 \leq r \leq R. \tag{92}$$

The r-step predictor filter coefficients, $\alpha_{1,2}^r$ and $\alpha_{2,2}^r$, are computed by solving the corresponding Yule-Walker equations as described in Section 6.4.

In the following, we develop the mathematical framework for media-aware RTO estimation, and illustrate the relation of the observed delay samples, timeout estimates, playout buffer size and retransmission round-trip times to the video quality. The following equations are provided in a generalized form, however, it should be noted that for packet n, the corresponding r-step predictor filter and error statistics are used, where $r = n - n^*$. We list our notation in Table 10.

Let p_n denote the probability of packet n being received by its decoding deadline, $t_D[n]$[3]. In our problem scenario, each packet has two transmission opportunities (one initial transmission and one retransmission). We first examine these cases separately, and then combine them together to compute p_n.

[3]Here, the decoding deadline represents the difference between the transmission time at the server and the decoding time at the client. Note that t_D is common for all packets belonging to a particular video frame.

Table 10: List of the notation for the optimization problem.

$\mathcal{A}_n$	Set of the ancestor packets for packet n
$\boldsymbol{\epsilon}_2^r[n]$	Prediction error for packet n
$F_{\boldsymbol{\epsilon}_2^r}$	Cumulative density function of $\boldsymbol{\epsilon}_2^r$
$\mathcal{F}_u$	Frame u
f_0	Frame rate of the original video
$\mathbf{I}[n]$	Retransmission indicator function for packet n
$\eta_{\mathcal{G}}$	Total number of frames in set $\mathcal{G}$
$\eta_{\mathcal{G}}^+$	Number of decodable frames in set $\mathcal{G}$
$\boldsymbol{p}_f[n]$	Pre-mature timeout probability for packet n
p_f	Desired probability of timing out pre-maturely
p_n	Probability of on-time delivery for packet n
p_n^1	Probability of on-time initial transmission for packet n
p_n^2	Probability of on-time retransmission for packet n
P_n	Decodability probability for packet n
$\mathcal{Q}_{\mathcal{G}}$	Video quality of set $\mathcal{G}$
$\mathbf{r}[n]$	Retransmission round-trip time for packet n
$\mathcal{G}$	Set of frames considered in the optimization
$\mathbf{s}[n]$	Observed delay for packet n
$\tilde{\mathbf{s}}_2^r[n]$	Predicted delay for packet n
$\boldsymbol{\tau}[n]$	Additional amount of waiting for packet n
$t_D[n]$	Decoding deadline for packet n
υ_u	Total number of packets in frame $\mathcal{F}_u$
υ_u^+	Number of decodable packets in frame $\mathcal{F}_u$

The first step is to calculate the probability of on-time initial transmission for packet n. Due to the correlation between the delay samples, this probability is given as follows:

$$p_n^1 = P\left\{\mathbf{s}[n] \le t_D[n] | \mathbf{s}[n^*], \mathbf{s}[n^* - 1]\right\}. \tag{93}$$

Expressing this conditional probability in closed form, however, is difficult. Instead, we can avoid the conditions by substituting $\mathbf{s}[n]$ with $\tilde{\mathbf{s}}_2^r[n] + \boldsymbol{\epsilon}_2^r[n]$. The conditional in (93) can be now expressed as an unconditional probability of the random variable $\boldsymbol{\epsilon}_2^r$:

$$p_n^1 = \begin{cases} F_{\boldsymbol{\epsilon}_2^r}\left(t_D[n] - \tilde{\mathbf{s}}_2^r[n]\right), & \text{if } \ \mathbf{s}[n] < \infty; \\ 0, & \text{if } \ \mathbf{s}[n] = \infty. \end{cases} \tag{94}$$

Note that p_n^1 is still conditioned on whether packet n is lost or not since the prediction error is only defined for non-lost packets. Thus, we also need to compute the loss probability of packet n.

In order to understand the relation between the packet loss and delay, we plot the loss probability of packet $n^* + r$ as a function of the delay of packet n^* for the $\Delta T = 40$ ms trace. Figure 65 shows that the loss probability for packet $n^* + 1$ is negligible if packet n^* experienced a delay smaller than 220 ms. However, if packet n^* experienced a delay between 220 and 260 ms, the chance of being lost for packet $n^* + 1$ increases up to 50%. Clearly, there is a strong dependence between the loss probability of packet $n^* + 1$ and the delay of packet n^*. More importantly, a noticeable dependence also exists between the loss probability of packet $n^* + r$ and the delay of packet n^* for $r \leq 10$. Ignoring this dependence and merely using the average packet loss rate (2.2% for this particular trace) would result in either an overrated or underrated packet loss probability. Therefore, it is important that we express the loss probability of packet n as a conditional on the last observed delay sample.

$$P\left\{\mathbf{s}[n] = \infty\right\} = P\left\{\mathbf{s}[n] = \infty | \mathbf{s}[n^*]\right\} \tag{95}$$

In practice, the conditional loss probability distribution can be generated on the fly. A closed form expression is not essential for media-aware RTO estimation.

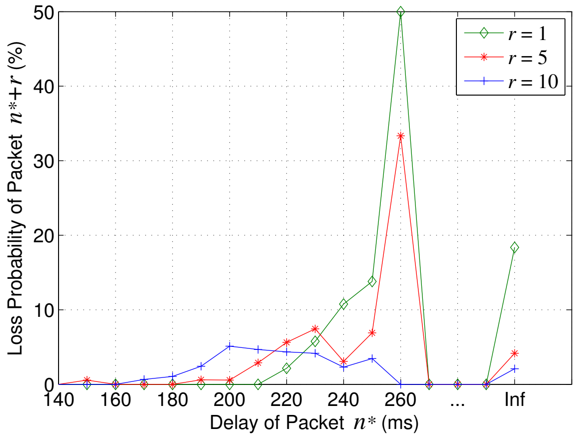


Figure 65: Relation between the packet delay and packet loss probability for $\Delta T = 40$ ms.

Having computed the first step of the on-time delivery probability, we now compute the probability of the retransmission for packet n being received before the decoding deadline.

Recall that when a packet is received, the client predicts the delay for the subsequent packet and estimates the timeout duration. If the expected packet is still not received within this time, a retransmission request is sent to the server. Assuming that the request is immediately processed by the server, the probability of an on-time retransmission equals

$$p_n^2 = P\left\{\tilde{\mathbf{s}}_2^r[n] + \boldsymbol{\tau}[n] + \mathbf{r}[n] \leq t_D[n]\right\}, \tag{96}$$

where $\mathbf{r}[n]$ is the round-trip time for the retransmission. It is important to note that in (96) we do not impose any condition on the previous delay samples. The reason is that the correlation between $\mathbf{r}[n]$ and $\mathbf{s}[n^*]$ is usually insignificant. Thus, p_n^2 can be computed from the empirical distribution of $\mathbf{r}$ in a straightforward manner.

Once we have computed p_n^1 and p_n^2, it is easy to express p_n as

$$p_n = p_n^1 + \mathbf{I}[n] \times (1 - p_n^1)p_n^2, \tag{97}$$

where $\mathbf{I}[n]$ is an indicator function: $\mathbf{I}[n] = 1$ if a retransmission is requested for packet n, and 0 otherwise. Considering that the chance of a retransmission arriving earlier than the initial transmission is negligible, (97) reduces to

$$\begin{aligned} p_n \quad = \quad & \Big(1 - P\left\{\mathbf{s}[n] = \infty\right\}\Big) \times F_{\boldsymbol{\epsilon}_2^r}\left(t_D[n] - \tilde{\mathbf{s}}_2^r[n]\right) \\ & + \mathbf{I}[n] \times P\left\{\mathbf{s}[n] = \infty\right\} \times P\left\{\tilde{\mathbf{s}}_2^r[n] + \boldsymbol{\tau}[n] + \mathbf{r}[n] \leq t_D[n]\right\}. \end{aligned} \tag{98}$$

As mentioned previously, packet n can only be decoded if all of its ancestor packets were decoded successfully. Thus, the decodability probability of packet n equals the following product:

$$P_n = p_n \times \prod_{n' \in \mathcal{A}_n} p_{n'}, \tag{99}$$

where $\mathcal{A}_n$ denotes the set of the ancestor packets for packet n. Here, we observe how the dependency structure of the streamed video explicitly factors in the video quality. More implicitly, we also notice that as more of its descendant packets are received by the client, an ancestor packet becomes more important since its successful delivery would enable the decoding of several packets. Note that in (99), we are able to express P_n as the product of individual packet decodability probabilities since any existing delay/loss correlation is already taken into account while computing p_n.

Given the packet decodability probabilities, we compute the expected number of decodable packets in frame $\mathcal{F}_u$ from

$$v_u^+ = \sum_{n=1}^{v_u} P_n. \tag{100}$$

Finally, the expected video quality of set $\mathcal{G}$ can be calculated by substituting (100) in (89).

The last step in our optimization problem is to calculate the redundant retransmission probability for each packet. Similar to our derivation in Section 6.4, we compute the premature timeout probability for packet n as follows:

$$\begin{aligned} \boldsymbol{p}_f[n] &= \mathbf{I}[n] \times P\left\{\tilde{\mathbf{s}}_2^r[n] + \boldsymbol{\tau}[n] < \mathbf{s}[n]\right\} \\ &= \mathbf{I}[n] \times \Big(1 - F_{\boldsymbol{\epsilon}_2^r}(\boldsymbol{\tau}[n])\Big). \end{aligned} \tag{101}$$

A solution to (90) is feasible only if the expectation of the redundant retransmission probability over all packets in set $\mathcal{G}$ satisfies the constraint given in (91).

7.3 Solution Approach and Implementation Issues

In this section, we first formulated an optimization problem and then derived a set of equations pertaining to this optimization problem. Depending on the complexity of the video dependency structure and the horizon of the optimization, the solution can potentially require a large number of multiplications and additions. In practice, however, solving the system given in (90) and (91) is less complicated than it may seem. For example, when network conditions are not severe and packet delays are below a certain threshold, the client can safely skip computing a timeout estimate for the subsequent packets based on the knowledge that the loss probability for those packets is negligible. The delay traces we collected reveal that the majority of the packets usually experience a non-critical delay, implying that the computational load of the RTO estimation on the client is often minimal.

It is, however, critical to solve (90) and (91) for the client when an incipient congestion is inferred. An important issue in this optimization is the selection of the optimization horizon and the granularity of the timeout durations. Suppose that we have R packets and we need to select a timeout duration for each of them from a set of H quantized values. In this case, our solution has a complexity of $O(H^R)$. Due to the exponential relation, the optimization horizon R cannot be chosen arbitrarily large. Furthermore, the predictive accuracy of the multi-step delay predictor degrades with R. Figure 66 shows that the prediction-error standard deviation doubles at step four and triples at step 10 for all delay traces. Since a poor prediction has no practical use, we suggest that the optimization horizon should not exceed 10. On the other hand, the value of H depends on the maximum complexity tolerable by the client. In our simulations and experiments, we selected the timeout durations among seven different values from the set $\mathcal{H} = \{0$ ms, 20 ms, 40 ms, 60 ms, 80 ms, 100 ms, $\infty\}$.

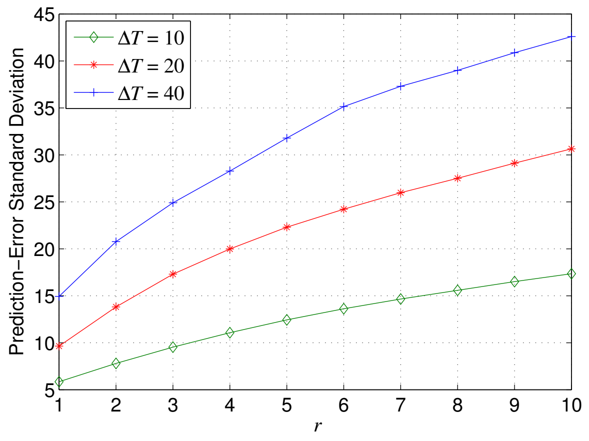


Figure 66: Variation of the prediction-error standard deviation with r.

For R-step prediction, we require R sets of the predictor filter coefficients, $\alpha^r_{1,2}$ and $\alpha^r_{2,2}$. To compute these coefficients, we use a window-based approach. The window size W is chosen short enough to ensure the pseudostationarity of the input data over the length W. Our tests indicate that $W = 20$ is a good choice. Given this window size, the predictor filter coefficients are computed by solving (72). Note that solving (72) requires the knowledge of the sample autocorrelations for the first two lags in our case. When a new sample is observed, the oldest sample is removed from the window and the other samples remain unchanged. Thus, the sample autocorrelations can be updated in an efficient manner. Furthermore, the filter coefficients vary over time, but due to the pseudostationarity of the data, we observe that $\alpha^r_{1,2} + \alpha^r_{2,2} = 1$. Thus, it is sufficient to compute only one of the coefficients for each r.

Here, we have to point out that the proposed media-aware RTO estimator is a considerably more elaborative method compared to Jacobson's algorithm or the RWM approach (See Section 6.2). Hence, it naturally requires more computational power. Yet, due to the simplifications discussed in this section, the computation overhead can be considered modest for today's available communication systems.

7.4 Simulation Results

In this section, we analyze the performance of the media-aware RTO estimator. We denote this estimator with $\mathrm{RTO}_{\mathrm{Media-aware}(R)}$, where R is the optimization horizon. Naturally, $\mathrm{RTO}_{\mathrm{Media-aware}(1)}$ is the same as the media-unaware RTO estimator, denoted by $\mathrm{RTO}_{\mathrm{AR}(2)}$, studied in Chapter 6. To better illustrate the impact of R on the performance of $\mathrm{RTO}_{\mathrm{Media-aware}(R)}$, we compare the on-time arrival rates of the individual frames in a GOP. For this purpose, we encoded a test sequence with a standard H.264 codec [120] at 300 Kbps and 20 frames per second. The adopted GOP structure was one I-frame plus nine P-frames. We streamed this video between two end-points in a moderately-congested Internet topology, where the forward-path packet loss rate averaged 5%.

In Figure 67, we plot the average on-time arrival rates of the individual frames when the playout buffer is 500 ms. Under the adopted simulation settings, we observe that $\mathrm{RTO}_{\mathrm{AR}(2)}$ could deliver approximately 30% of the retransmissions on time, and, as expected, this success rate did not vary much among the frames. In contrast, $\mathrm{RTO}_{\mathrm{Media-aware}}$ was able to deliver as much as 40% of the I-frame retransmissions on time at the expense of least important P-9 frames. In other words, $\mathrm{RTO}_{\mathrm{Media-aware}}$ recovered more of the important video content by not increasing the streaming rate, but by relinquishing the recovery of the less important content. It is important to note that as we increased the optimization horizon, the optimization gain improved.

Next, we plot the average on-time arrival rates when the playout buffer is relaxed to 600 ms. In this case, $\mathrm{RTO}_{\mathrm{AR}(2)}$ delivered 90% of the retransmissions on time. Naturally, the on-time retransmission probability improved with an increase in the playout buffer size. Although $\mathrm{RTO}_{\mathrm{Media-aware}}$ still produced better quality video, the performance gap between

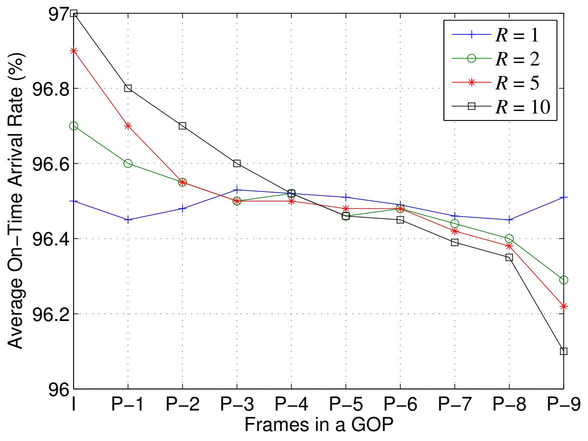


Figure 67: Variation of the frame on-time performance with the optimization horizon when the playout buffer is 500 ms.

the media-aware and media-unaware approaches reduced with respect to the previous case. Thus, we conclude that the media-aware RTO estimation becomes more crucial under low end-to-end delay requirements, and as the delay requirement relaxes, its performance converges to that of the media-unaware RTO estimation.

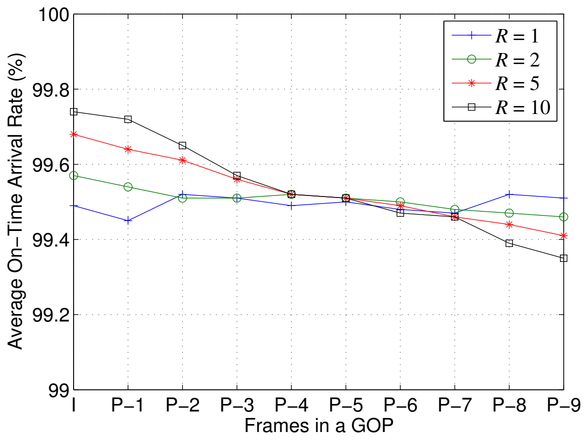


Figure 68: Variation of the frame on-time performance with the optimization horizon when the playout buffer is 600 ms.

7.5 Internet Experiments

In this section, we present a comprehensive set of experimental results and assess the performance of different RTO estimation methods. First, we explain the setup and then discuss the results.

7.5.1 Experimental Setup

We developed and established an experimental platform on the Internet. On this platform, we emulated a real-time video streaming application over UDP between a broadband client in Ankara, Turkey and a broadband server in Atlanta, GA USA[4]. The client simultaneously streamed video packets from the server and carried out the RTO computation. When a packet was identified as lost, a retransmission request was sent to the server. If this request was successfully received, the server immediately retransmitted the requested packet.

In the implementation, we identified each packet with a unique sequence number assigned by the server. Prior to any video transmission, the client synchronized its internal clock with the server [152] and received video-specific information such as encoding bitrate and GOP structure, from the server. This information as well as the delay measurements were used by the client for real-time RTO estimation.

We conducted our experiments in six sessions of 60 minutes, where we tested all five different RTO estimators discussed throughout Chapter 6 and this chapter. After each session, the mean delay and packet loss rate were measured to ensure that similar network characteristics were observed in all sessions. The mean round-trip delay and mean forward-path packet loss rate were measured as approximately 250 ms and 6.0%, respectively. While all lost packets were eventually detected by any of the RTO estimators, the total recovery times varied depending on the agility of the RTO estimators. To illustrate this variation, we examine the delay distributions of the retransmitted packets. Figure 69 shows that we empirically required a playout buffer of 440 ms to merely enable the retransmission capability. Naturally, with a larger playout buffer, more and more retransmissions would reach the client on time. Figure 69 also reveals that playout buffers larger than 600 ms were essentially useless since even the worst-performing RTO estimator completed its retransmissions within 600 ms. Yet, for the sake of demonstrating the impact of accurate RTO estimation, we adopted a playout buffer of half a second in producing the results presented in this section.

[4]The reason for experimenting over an intercontinental network was to observe a wide range of packet loss and delay characteristics.

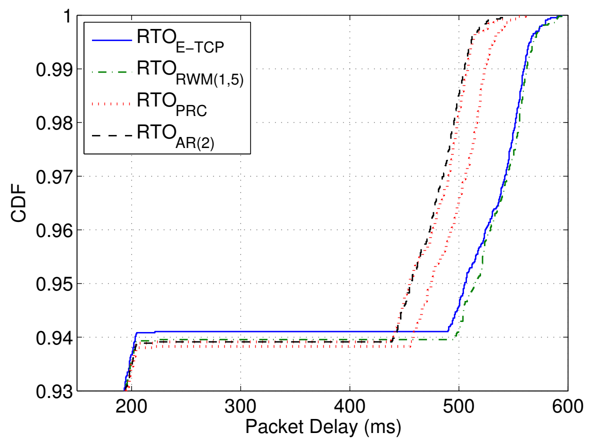


Figure 69: Total-delay distributions for different RTO estimators.

For video quality comparison, we encoded the test sequences FOREMAN (352×288) and SUZIE (176×144) with a standard H.264 codec [120] and a GOP structure of one I-frame plus nine P-frames at 600 Kbps, 20 frames per second and 150 Kbps, 30 frames per second, respectively. We report our results in terms of three metrics: (*i*) percentage of the on-time packets, (*ii*) average video quality, and (*iii*) glitch rate. For the average video quality, we use the peak signal-to-noise ratio (PSNR) measure on the luminance (Y) channel. Glitch rate is defined as the percentage of the frames whose PSNR value is below a certain threshold. The value of this threshold is sequence-specific and should be determined by visually inspecting the individual frames. For the FOREMAN and SUZIE sequences, we selected a threshold of 30 and 34 dB, respectively. During the decoding process, we concealed the missing macroblocks with the ones in the last successfully-decoded frame in order to reduce the amount of severe transitions from the frames in error. In addition, the packets that missed their decoding deadlines, *i.e.*, late packets, were still used in the decoding process to improve the decodability of the predictively-encoded frames. Yet, Figure 70 shows that the late packets severely affected the video quality rendered at the client.

7.5.2 Results

In the following discussion, we first present the results obtained from the FOREMAN sequence. Then, we will provide the results for the SUZIE sequence.

An immediate result of our experiments is that without any error control, the video

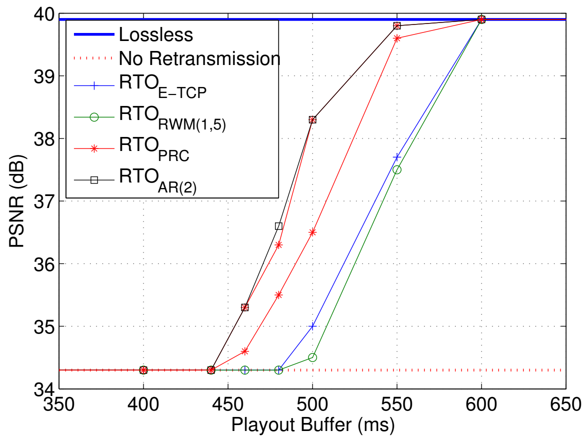


Figure 70: Variation of the video quality with the playout buffer size for the FOREMAN sequence.

quality severely suffers from the lost packets; in the case of the FOREMAN sequence, the average streaming quality barely reached 34.3 dB, which is 5.6 dB lower than the lossless case. When RTO$_{\mathrm{E-TCP}}$ was employed, the video quality improved by 0.7 dB at the expense of an average rate increase of 39 Kbps. However, 7% of this rate increase was redundant due to pre-mature timeouts, and 86% of it was useless since those packets missed their decoding deadlines. The second method, RTO$_{\mathrm{RWM}(1,5)}$, improved the video quality by only 0.2 dB while increasing the average transmission rate by 38 Kbps. We observed that 5% of the retransmissions requested by RTO$_{\mathrm{RWM}(1,5)}$ were redundant and 93% of them were late. These results clearly indicate that the RTO estimators that are primarily designed for TCP are not suitable for low-delay applications. The overinflated estimates inevitably disrupt the timely detection of lost packets, which adversely affects the on-time retransmission performance.

In the third session, we tested RTO$_{\mathrm{AR}(2)}$ and obtained an average video quality of 38.3 dB (4.0 dB improvement over the no-retransmission case) at an average streaming rate of 638 Kbps. The quality gain stemmed from the fact that only 5% and 30% of the total retransmissions were redundant and late, respectively. In the fourth and fifth sessions, we experimented with RTO$_{\mathrm{PRC}}$ with two different p values. These p values were chosen by trial and error such that (*i*) RTO$_{\mathrm{PRC}}$ recovered as many packets on time as RTO$_{\mathrm{AR}(2)}$, and (*ii*) RTO$_{\mathrm{PRC}}$ consumed an average total streaming rate equal to the one consumed by $\text{RTO}_{\text{AR(2)}}$. In the first case, RTO_{PRC} produced almost three times more redundant retransmissions compared to $\text{RTO}_{\text{AR(2)}}$. In the second case, on the other hand, RTO_{PRC} barely delivered a 36.5 dB video.

Finally, in the last session, we experimented with $\text{RTO}_{\text{Media-aware(10)}}$. While the percentage of the late retransmissions slightly increased with respect to $\text{RTO}_{\text{AR(2)}}$, $\text{RTO}_{\text{Media-aware(10)}}$ further improved the video quality by 0.3 dB and delivered the highest quality of video at 38.6 dB. The comparison of $\text{RTO}_{\text{Media-aware(10)}}$ against the other four RTO estimators for the FOREMAN sequence is given in Table 11.

Table 11: Experimental results for the FOREMAN sequence.

	Rate (Kbps)	Rate of Redundant Retransmission	Rate of Late Retransmission	PSNR (dB)	Glitch Rate
Lossless	600	N/A	N/A	39.9	0%
No Retransmission	600	N/A	N/A	34.3	35%
$\text{RTO}_{\text{E-TCP}}$	639	7%	86%	35.0	28%
$\text{RTO}_{\text{RWM(1,5)}}$	638	5%	93%	34.5	30%
RTO_{PRC}	644	18%	29%	38.3	8%
RTO_{PRC}	638	5%	55%	36.5	11%
$\text{RTO}_{\text{AR(2)}}$	638	5%	30%	38.3	8%
$\text{RTO}_{\text{Media-aware(10)}}$	638	5%	35%	38.6	4%

We present the results for the SUZIE sequence in Table 12. Compared to the results of the higher-bitrate FOREMAN sequence, we observe that RTO_{PRC} showed a little performance improvement, whereas the other RTO estimators experienced a small amount of performance degradation. This observation is consistent with the simulation results presented in Section 6.4.

We summarize our experimental results for both test sequences on a rate-quality plane. In Figures 71 and 72, we clearly see that $\text{RTO}_{\text{Media-aware(10)}}$ outperforms all other four RTO estimators by achieving a higher video quality while streaming at an equal or smaller bitrate. In other words, $\text{RTO}_{\text{Media-aware(10)}}$ has the best rate-quality performance.

Table 12: Experimental results for the SUZIE sequence.

	Rate (Kbps)	Rate of Redundant Retransmission	Rate of Late Retransmission	PSNR (dB)	Glitch Rate
Lossless	150	N/A	N/A	38.9	0%
No Retransmission	150	N/A	N/A	36.5	27%
$\text{RTO}_{\text{E-TCP}}$	160	7%	96%	36.6	25%
$\text{RTO}_{\text{RWM}(1,5)}$	160	8%	92%	36.7	23%
RTO_{PRC}	161	15%	40%	37.8	9%
RTO_{PRC}	160	7%	50%	37.5	13%
$\text{RTO}_{\text{AR}(2)}$	160	7%	40%	37.8	9%
$\text{RTO}_{\text{Media-aware}(10)}$	160	7%	44%	38.1	3%

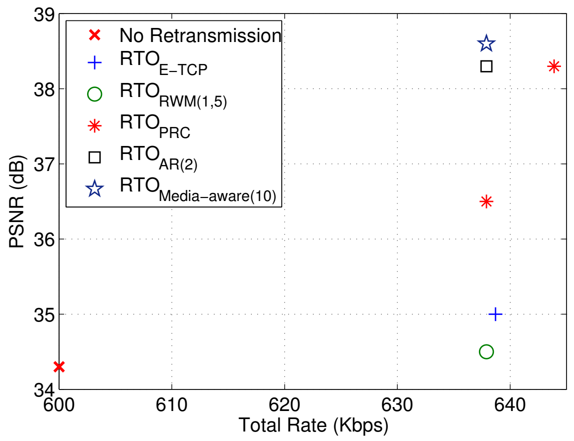


Figure 71: Rate-quality performance of different RTO estimators for the FOREMAN sequence.

7.6 Conclusions

Previously in Chapter 6, we proposed an autoregression-based adaptive RTO estimation method for low-delay Internet video applications. This method substantially outperformed existing estimators such as the enhanced TCP-like RTO estimator and recursive weighted median filtering. In this study, we furthered our approach and developed a media-aware RTO estimator. This RTO estimator computes the optimal timeout duration for each packet such that the video quality rendered at the client is maximized for a given retransmission rate budget. Our simulations show that media-aware RTO estimation provides a significant quality improvement over its media-unaware counterpart, particularly when the application

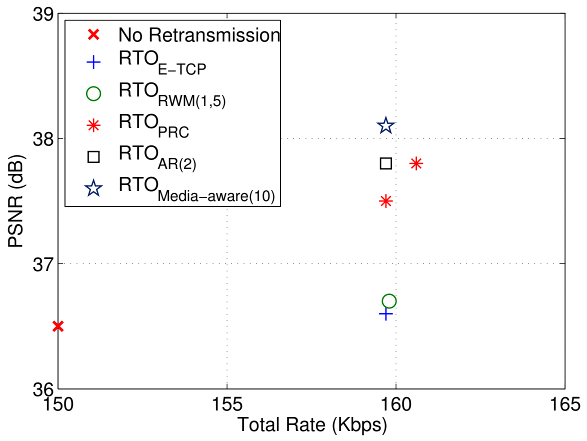


Figure 72: Rate-quality performance of different RTO estimators for the Suzie sequence.

requires a low end-to-end delay.

CHAPTER VIII

CONCLUSIONS AND FUTURE WORK

8.1 Contributions

This thesis develops a collection of end-to-end and system-level protocol-based solutions for a diverse set of multimedia applications, and conducts a comparative performance analysis against the state-of-the-art methods. Contributions of this thesis include the following:

- We develop an optimal multi-path transmission framework for streaming error-resilient videos over overlay networks in low-delay video applications,
- We experiment with and analyze the performance of the single/multi-hop and single/multi-path transmission methods for streaming high-resolution videos in mesh networks,
- We formulate a mathematical framework that models on-demand video delivery from multiple content servers, and develop a packet scheduling algorithm that achieves the rate-distortion optimal performance within a multi-server streaming system,
- We investigate the use of proxies to overcome the problems associated with packet loss, large delay and delay jitter in interactive video applications,
- We characterize the packet dynamics in networked-video applications, propose models for packet delay, develop accurate delay and delay-boundary prediction methods, and interpret the correlation between the packet delay and packet loss events,
- We engineer media-aware and network-adaptive error-control methods, and lay out a framework that computes the optimal actions for error control.

8.2 Future Research Directions

Multimedia networking is an active area of research, and it is continuously evolving. There are several open issues that remain to be thoroughly researched. Below, we present an overview for the ones that fall into the context of this thesis.

Optimized Residential Wireless Mesh Networks:

During our experiments in Chapter 3, we fixed the location of the mesh nodes since most in-home nodes such as TVs, PVRs and DVD players are usually immobile. However, intermediate relay nodes can sometimes be mobile nodes, in which case the problem of low-latency video transmission becomes more complicated. The second simplification we made in that chapter was that we considered a single flow throughout the mesh network and did not allow any other background traffic. Of course, in real life, there will very likely be several simultaneous data flows as well as multiple video flows, each potentially originating and terminating at different nodes. In multi-sender multi-receiver environments, a joint optimization for path selection and resource allocation will be compulsory, which, we believe, will be a good extension to this work.

In-Network Cooperative Media-Aware Error Control:

In Chapter 5, we studied the potential benefits of using proxies in providing improved error-control/protection capabilities to the clients communicating with each other over long distances. In Appendix C, our simulations showed the effectiveness of the I-Proxy approach and that its advantages could be best exploited with a proper I-Proxy selection. An interesting future work is to extend the idea of I-Proxy to Skype-like peer-to-peer networks where multiple client pairs exist, and the I-Proxy functionalities are carried out by actual clients rather than dedicated proxy servers. Two important problems that deserve further attention are the selection of the relay nodes and the error-control functionalities that will be performed at the relay nodes.

Proxy-Based On-Demand Video Distribution in Cable Networks:

A challenging problem for cable TV operators is to deliver the on-demand content from potentially several different content providers throughout the world to its customers in a high-quality fashion. Customers demand, actually will pay only for, an uninterrupted service and error-free audiovisual quality. While the on-demand content can be easily pushed from local caches, head-ends or even central content repositories by the help of intelligent content caching and replication, several problems likely occur when the content is pushed from outside sources. The outsourced video streams are subject to congestion in the public Internet and at the points-of-presence (PoP), where several Internet service providers interconnect to each other.

A practical solution to this problem is to co-locate proxy servers at the ingress points. These proxies partition each end-to-end connection into two sub-connections. This allows us (*i*) faster packet loss detection and recovery, (*ii*) jitter compensation via buffering, and (*iii*) content adaptation via transcoding and transrating. Furthermore, if the network conditions between the outside source and designated proxy deteriorate over time, the incoming video traffic can be rerouted over alternative proxies to deliver a continuous video experience to the customers. Another advantage of this approach is that the cable operators can use their proprietary transport and application-layer protocols for video within their network, while retaining a standard-compliant connection with the external content providers.

Delay Prediction in Wireless and Mixed Networks:

Delay and delay-boundary prediction are two essential components for error-control and protection methods. In Chapter 6, we studied autoregressive models to characterize the nature of the packet delay observed in Internet video applications. Our simulations and experiments mainly focused on the wired networks. Yet, applications that are running between two wireless clients or between a wireless client and a wired client also require accurate models for delay and delay-boundary prediction. To this effect, research needs to be performed to capture the unique features of the wireless channels and develop tailored prediction models.

APPENDIX A

FAST HEURISTICS FOR OPTIMAL MULTI-PATH SELECTION

In Chapter 2, we modeled multi-path streaming for multiple description video, and presented a formal way to evaluate the streaming distortion for each set of paths in a given network. The optimal solution was a straightforward evaluation of the distortion for each possible path pair and choosing the one with the minimum distortion. However, given that (3) is highly nonlinear, solving this optimization problem may become computationally-intractable in large topologies. In particular, optimal solution may not be feasible if the client does not have enough processing power, memory resources and time to find the solution, or if the client resides in a network where the conditions change rapidly. In this appendix, we address this problem and investigate a fast heuristics-based solution.

First, we analyze the brute-force complete enumeration approach. Then, we develop a heuristic that selects a good pair of diverse paths. This heuristic is based on first identifying a small subset of good paths, and then choosing the pair among this subset that minimizes (3). In developing our heuristic, we exploit the routing hierarchy in the Internet. See [104] for further details on this subject.

A.1 Brute-Force Approach

Consider an Internet topology with M transit domains (TD) each of which contains N nodes. An illustrative topology with four TDs is shown in Figure 73. Assume that the multi-path routing capability is enabled only within TDs. In this topology, the brute-force (BF) approach for finding the optimal pair enumerates all possible paths between the server and client. Then, (3) is computed for each pair, and the pair that gives the minimum average distortion is selected.

The complexity analysis of this approach can be summarized as follows: All paths in the given topology can be generated in $O(2^{MN})$ time, and (3) is evaluated for all possible

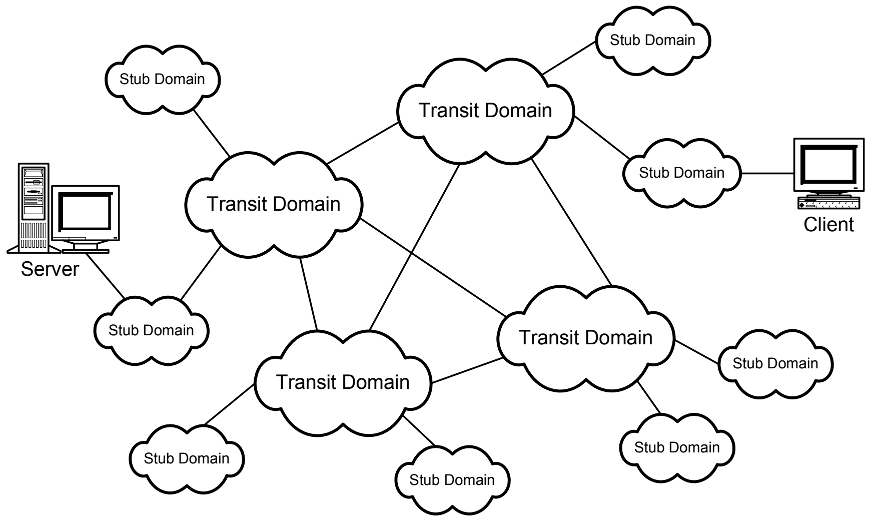


Figure 73: An Internet topology with four transit domains. Reproduced from Section 2.5.3.

pairs in $O(MN \times 2^{2MN})$ time. The identification of the optimal pair requires $O(\log 2^{2MN})$ time. Hence, the BF approach has a complexity of

$$O(2^{MN}) + O(MN \times 2^{2MN}) + O(MN) = O(MN \times 2^{2MN}). \tag{102}$$

In (102), we observe that the BF approach complexity increases exponentially with both M and N. Clearly, this approach does not scale well. Below, we develop a more efficient heuristic, which we refer to as the *rapid path generation* (RPG) approach.

A.2 Rapid Path Generation

An important characteristic of the Internet is that a path connecting two nodes in different stub domains (SD), *e.g.*, the ones containing the server and client, never passes through any other SD [104]. This feature implies that multi-path routing mainly takes place within or between the TDs. Moreover, between two TDs, usually there is only one link through which all traffic from the source domain is carried to the destination domain. This natural decomposition of the Internet leads us to consider the TDs individually in the path-generation process.

We start our algorithm by finding good paths in each TD. During this process, we also exploit the fact that among the links in a TD, only a few of them may be heavily congested at a time while the rest is lightly congested or not congested at all. Eliminating these congested links reduces the topology size. Moreover, after this reduction, the remaining links will have similar (almost zero) packet loss rates. On the other hand, the end-to-end bandwidth is often limited by the SD or inter-domain links. We assume that TD links can support the bandwidth desired by the clients. Hence, the bandwidths of the remaining TD links have a negligible effect in selecting the good paths.

Now, we are left only with the delay and jitter parameters in the path-generation process. To find a subset of good paths in each TD, we solve a k-shortest path algorithm where the cost on each link is the sum of its delay and average jitter. The most straightforward k-shortest path algorithm for M TDs requires $O(kMN^2)$ time. After selecting k good paths in each TD, we combine them to create $O(k^M)$ end-to-end paths. Then, out of these paths we form $O(k^{2M})$ pairs. At this point, we evaluate (3) for each pair. This process takes $O(MN \times k^{2M})$ time. Finally, selecting the pair with the minimum average distortion can be accomplished in $O(\log k^{2M})$ time. Hence, the RPG approach has an overall complexity of

$$O(kMN^2) + O(MN \times k^{2M}) + O(M) = O(MN \times k^{2M}). \tag{103}$$

Generally, we have $M < N$, which is the case in small-to-moderate sized networks. Hence, although the RPG approach runs in exponential time in M, it still provides an important improvement over the BF complete enumeration, which requires exponential time in the total number of nodes, *i.e.*, $M \times N$. In addition, with this approach rather than trying to seek the solution in the whole network, we handle reduced-sized problems in each TD separately. Hence, the demand for the memory space in the RPG implementation is substantially lower than that in the BF implementation. On the other hand, k is an important parameter in (103). We will examine its impact on the performance of the RPG approach in the next section.

A.3 Performance Analysis

In order to evaluate the performance of the RPG approach, we used random Internet topologies generated by GT-ITM [104]. We varied k, the number of TDs and SDs to analyze the effects of the variations in the network connectivity. The link parameters were assigned as described in Section 2.5.3. For each random topology, we first found the path pairs corresponding to the BF and RPG solutions. Then, by using each pair we streamed a standard test sequence from the server to the client with a delay tolerance of 200 ms[1]. We repeated this process 100 times to obtain reliable PSNR results. The results are tabulated in Tables 13 - 15 along with the values of k, M and N. The PSNR differences between the BF and RPG approaches are also given in the last column of each table. Note that each row represents a different random topology. Hence, the results in a row of a table should only be compared with the results in the same row of other tables.

Table 13: Comparison of the BF and RPG approaches when $k = 1$.

k	M	N	BF	RPG	Difference
1	2	5	33.39 dB	31.82 dB	1.57 dB
1	2	8	33.45 dB	31.56 dB	1.89 dB
1	3	5	33.36 dB	31.63 dB	1.73 dB
1	3	8	33.38 dB	31.44 dB	1.94 dB
1	4	8	33.30 dB	31.39 dB	1.91 dB

Table 14: Comparison of the BF and RPG approaches when $k = 2$.

k	M	N	BF	RPG	Difference
2	2	5	33.39 dB	32.93 dB	0.46 dB
2	2	8	33.45 dB	32.85 dB	0.60 dB
2	3	5	33.36 dB	32.84 dB	0.52 dB
2	3	8	33.38 dB	32.66 dB	0.72 dB
2	4	8	33.30 dB	32.43 dB	0.87 dB

The results show that as we identify more good paths in each TD, the performance of the RPG approach gets closer to the optimal in all topologies. Although there is approximately 1.80 dB difference in the average qualities when $k = 1$, this difference reduces to around 0.16 dB when $k = 3$. This is also shown in Figures 74 - 76, where individual frame qualities are plotted. In Figure 76, the RPG plot tracks the BF plot more closely on the average than it does in Figure 74. Even when $k = 3$ (and $M = 4$, $N = 8$), the number of operations

[1]We used the TABLE TENNIS (352×240) sequence. This sequence consisted of 150 frames. The frame rate was 30 frames per second.

Table 15: Comparison of the BF and RPG approaches when $k = 3$.

k	M	N	BF	RPG	Difference
3	2	5	33.39 dB	33.32 dB	0.07 dB
3	2	8	33.45 dB	33.30 dB	0.15 dB
3	3	5	33.36 dB	33.24 dB	0.12 dB
3	3	8	33.38 dB	33.15 dB	0.23 dB
3	4	8	33.30 dB	33.06 dB	0.24 dB

required by the RPG approach is 500K times smaller than the one required by the BF approach, which proves a quite large saving in terms of processing power and computation time.

An interesting issue here is the choice of k. In addition to the reported results, we also tested the same topologies with $k = 4$. However, the improvement over the case when $k = 3$ was not that significant. Hence, we conclude that keeping $k = 3$ is a good choice. It is worthy to note that $k = 1$ does not necessarily imply that the 1^{st} and 2^{nd} shortest paths are selected between the server and client. Although only the shortest path is computed in each TD for $k = 1$, we still evaluate (3) for all possible end-to-end pairs and choose the pair accordingly.

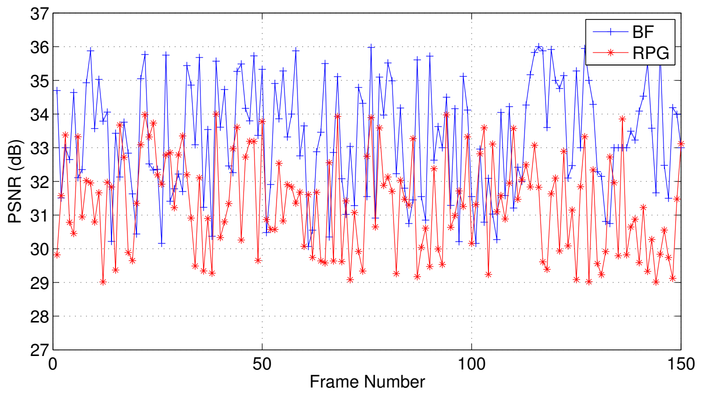


Figure 74: The PSNR comparisons of the BF and RPG approaches when $k = 1$, $M = 4$ and $N = 8$.

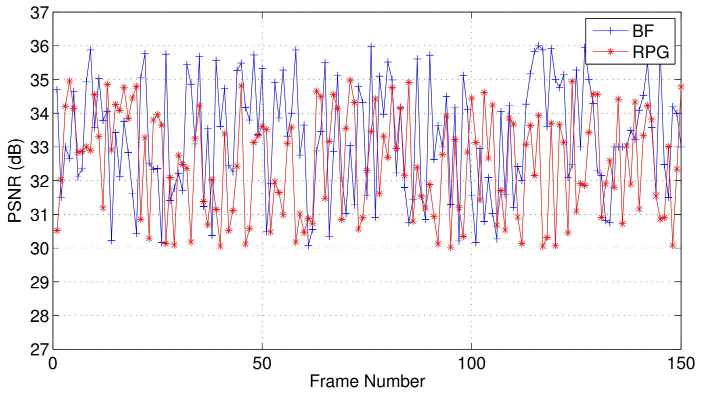


Figure 75: The PSNR comparisons of the BF and RPG approaches when $k = 2$, $M = 4$ and $N = 8$.

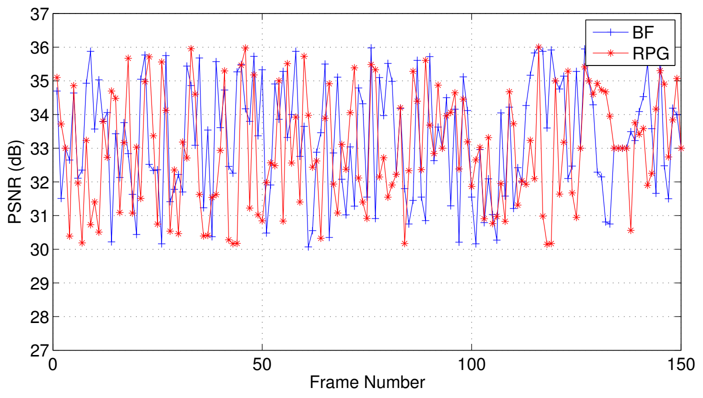


Figure 76: The PSNR comparisons of the BF and RPG approaches when $k = 3$, $M = 4$ and $N = 8$.

A.4 Conclusions

In this appendix, we presented a fast heuristics-based solution for the optimal multi-path selection problem. This approach performs within 0.2 dB of the optimal solution in small-to-moderate sized networks without incurring any runtime or memory-space problems. Particularly, this fast heuristic is best suited to interactive multimedia applications such as videoconferencing and VoIP, where multi-path computation is a time-critical process. In addition, our approach is also suitable for the clients whose processing-power capabilities are limited.

APPENDIX B

MULTIPLE DESCRIPTION VS. SCALABLE VIDEO STREAMING WITH OPTIMAL DIVERSE ROUTING

With the proliferation of multiple description (MD)-based streaming solutions, it naturally became necessary to conduct a performance comparison study for MD and scalable video (SV) streaming. Several researchers investigated this issue to date in different setups and scenarios [44, 72, 75, 153, 154]. A common result out of all these studies was that MD streaming outperformed SV streaming when no error protection was applied on the sub-bitstreams. The main reason was that any erasure in the base layer would cause irrecoverable errors and avoid the decoding of other enhancement layers. In contrast, erasures in a description would only affect that description, leaving other descriptions completely intact. These studies also reported that protecting the base layer against erasures and packet losses was essential to improve the performance of SV streaming. Two main suggested methods were retransmission-based error control and forward error correction. However, these methods were prohibitive in some scenarios due to the extra delays introduced.

While these studies shed light on the specifics of MD vs. SV streaming, they often fell short to provide an analytical comparison approach. In this appendix, we address this problem by extending the multi-path selection framework that we developed in Chapter 2 to SV streaming. We claim that using multiple transmission paths for SV streaming can be an effective way in protecting the base layer, provided that the paths are selected intelligently. A major advantage of this approach is that it achieves high-quality SV streaming without using any error control or protection.

The contribution of this study is two-fold: First, we provide an optimal multi-path selection method for SV streaming. Second, we propose a quality comparison methodology for determining whether MD or SV streaming performs better under the given network conditions. With this methodology, a multi-codec application that supports both MD and

SV codecs can estimate the performance of both streaming approaches prior to and during the video transmission, and make a codec switch, if necessary. In this perspective, the results presented in this study not only provide prominent insights into the capabilities of MD and SV streaming but also complement the MD vs. SV streaming performance comparison studies conducted so far.

B.1 Multi-Path Selection for Scalable Video Streaming

In an MD streaming system, individual descriptions can be decoded independently. However, in an SV streaming system, the base layer has to be received intact to be able to decode any enhancement layer. Suppose that our SV streaming system produces one base layer and one enhancement layer. For SV streaming, we define three distortion terms: (*i*) $D_{1,1}$ is the achieved distortion when both the base and enhancement layers are received successfully on time, (*ii*) $D_{1,0}$ is the achieved distortion when only the base layer is received successfully on time, and (*iii*) $D_{0,x}$ is the achieved distortion when the base layer is lost or late. Thus, we can express the expected distortion for SV streaming at the client as

$$\begin{aligned} E\{D\} &= \overbrace{P\left\{\text{Both received on time}\right\}}^{P_{1,1}} \times D_{1,1} \\ &\quad + \overbrace{P\left\{\text{Base layer received on time \& enhancement layer lost or delayed}\right\}}^{P_{1,0}} \times D_{1,0} \\ &\quad + \underbrace{P\left\{\text{Base layer lost or delayed}\right\}}_{P_{0,x}} \times D_{0,x}. \end{aligned} \tag{104}$$

Note that the arrival of the enhancement layer is a "don't care" in $P_{0,x}$ since the enhancement layer cannot be decoded unless the base layer is successfully received on time.

Due to the fundamental differences in MD and SV coding, MD and SV streams generally produce different average qualities (distortions), even they are encoded at the same total rate. In order to estimate the corresponding distortions accurately by using (2), we need to compute the actual source rates for both of them. Let's explore this on an example.

Consider a source video encoded by an MD/SV encoder at a total rate of R_M. To achieve the same quality, suppose that it is sufficient to encode the same source video at a rate of R_S for a conventional single stream encoder. Since the redundancy introduced into the stream by the single stream encoder is smaller, we have $R_S \leq R_M$. We define the *redundancy level* injected into an MD and SV stream as the proportion of the additional rate needed by its encoder. That is, the redundancy level, denoted by ρ, is given by

$$\rho = \frac{R_M - R_S}{R_M}. \tag{105}$$

We compute the redundancy level of the individual sub-bitstreams, *e.g.*, the descriptions and layers, with the same formula by replacing the total rate with the sub-bitstream rates in (105). Now, we can revise the distortion model given in (2) as

$$D(r_k, \rho) \approx \frac{\kappa}{r_k \times (1 - \rho)}, \tag{106}$$

where r_k is the rate in terms of the number of bits per source sample on the k^{th} channel (defined in (6)), and κ is a model parameter. With this revised distortion model, we apply the same set of equations derived in Section 2.4.1 to compute the distortion terms of $D_{1,1}$, $D_{1,0}$ and $D_{0,x}$. Note that $D_{0,x}$ is computed in a similar fashion as $D_{0,0}$. The success probabilities $P_{1,1}$, $P_{1,0}$ and $P_{0,x}$ are also computed as discussed in Section 2.4.2.

B.2 Simulation Results

In this section, we investigate the performance of MD and SV streaming in a simulation environment. For this purpose, we developed a real-time streaming application that supported both MD and SV codecs. We adopted the simulation settings from Section 2.5. We present several results for the standard test sequences Table Tennis and Flower Garden, both of which have a resolution of 352×240 pixels.

In the simulations, the MD and SV streams were produced by the help of MPEG-2 based MD and SV encoders, respectively. For MD streaming, we preferred the time-domain partitioning method with two descriptions. For SV streaming, we used the SNR-scalability mode with two layers (one base plus one enhancement layer). The redundancy levels of each stream and sub-bitstreams were obtained from a lookup table, which was prepared offline for the encoders used in the simulations. During the decoding process, we concealed the missing frames by repeating the information from the last available frame.

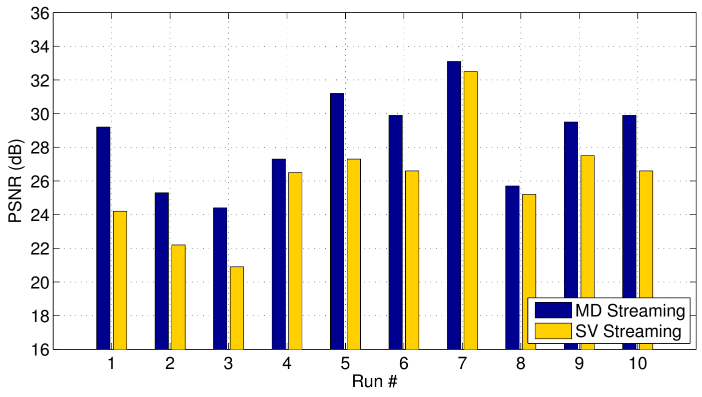
To emulate a realistic simulation environment, we generated a random Internet topology with GT-ITM [104]. This topology consisted of 240 nodes with four mesh-connected transit domains, 10 nodes per transit domain, one stub per transit-domain node and five nodes in a stub domain. We assumed multi-path routing capability at each transit-domain node. The link parameters were assigned random values as described in Section 2.5.3. We compared three different streaming methods: (*i*) shortest-path, (*ii*) maximally link-disjoint path, and (*iii*) optimal multi-path streaming.

Although we held the topology unchanged, we varied the link parameters and chose different congested links in each of the 10 independent simulations. We summarize the results for the Table Tennis and Flower Garden sequences in Figures 77 and 78, respectively. Not surprisingly, maximally link-disjoint path streaming outperforms shortest-path streaming. However, optimal multi-path streaming further achieves a better video quality in all simulations.

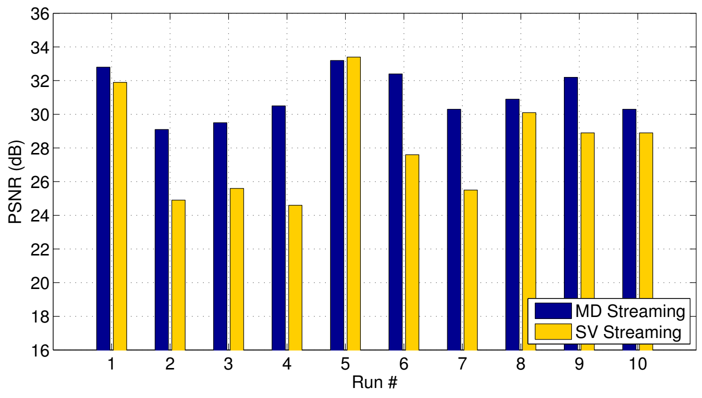
B.3 Discussion

A quick examination of the results reveals the fact that the average video quality is degraded severely when the shortest-path streaming method is employed as the underlying routing method. Moreover, the streaming quality widely fluctuates, which further degrades the perceptual quality. Particularly, this has more devastating effects on SV streaming, which is an expected result since the quality of the SV stream often suffers from the losses incurred in the base layer. On the other hand, MD stream is able to recover from the losses with a slight quality degradation unless both descriptions are concurrently lost. These results agree with those reported in [44, 75].

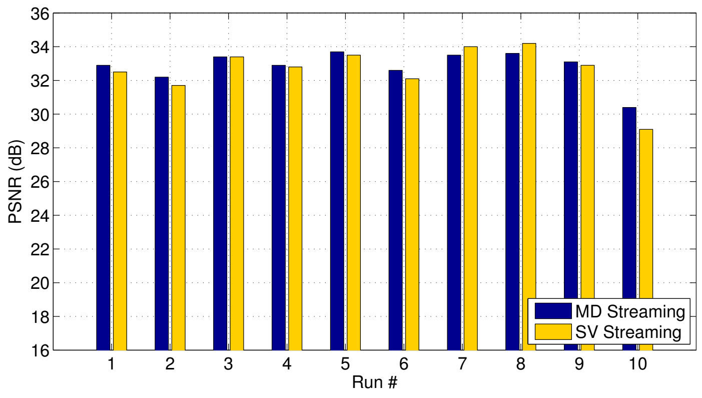
The results also show that maximally link-disjoint path streaming is able to reduce the adverse effect of a congested link by exploiting the path diversity and improves the average quality over shortest-path streaming. However, the improvement in SV streaming is not as much as the one in MD streaming. The reason is that the path assigned to the base layer by the maximally link-disjoint path streaming method need not necessarily be a good path. If this path experiences packet losses and/or long delays, SV streaming still



(a)

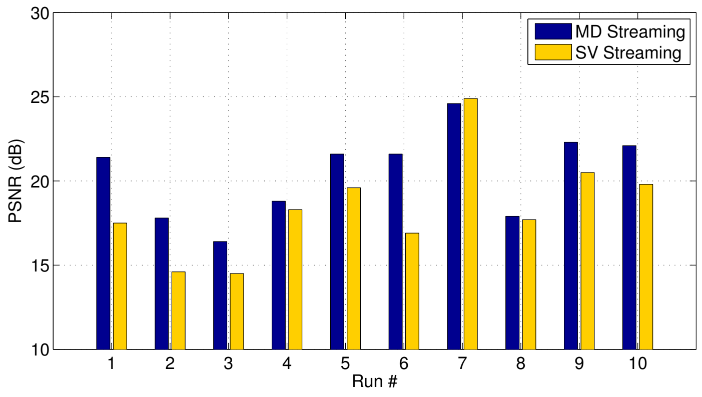


(b)

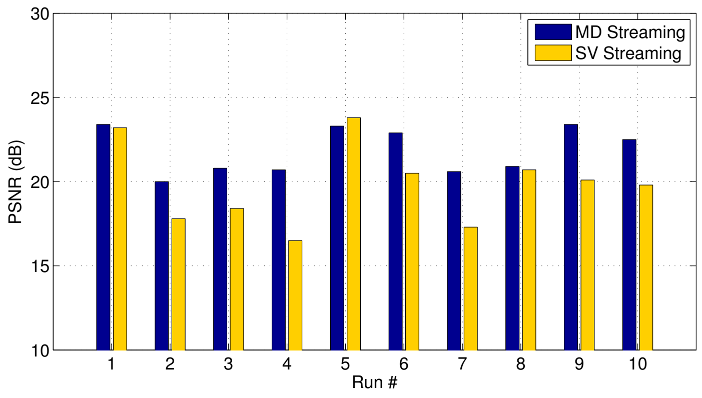


(c)

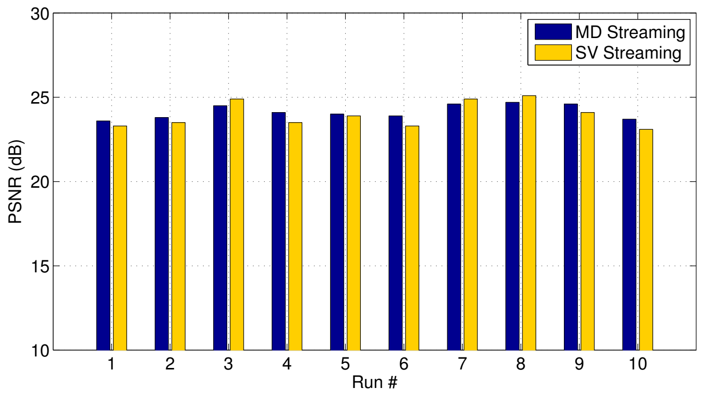
Figure 77: Average PSNRs for different streaming methods for the Table Tennis sequence: shortest-path streaming (a), maximally link-disjoint path streaming (b) and optimal multi-path streaming (c).



(a)



(b)



(c)

Figure 78: Average PSNRs for different streaming methods for the Flower Garden sequence: shortest-path streaming (a), maximally link-disjoint path streaming (b) and optimal multi-path streaming (c).

performs poorly, even though the enhancement layer is transmitted over a link-disjoint path. However, the performance of SV streaming increases drastically with optimal multi-path streaming. This is because optimal multi-path streaming considers the characteristics of the base and enhancement layers, and selects the paths accordingly.

B.4 Conclusions

In this appendix, we provided an optimal multi-path selection method for SV streaming and proposed a quality comparison methodology for determining whether MD or SV streaming performed better under the given network conditions. Our simulations ran over an overlay infrastructure showed that sizeable PSNR improvements could be achieved when the video was streamed over intelligently-selected multiple paths instead of the shortest path or maximally link-disjoint paths.

APPENDIX C

PROXY SELECTION METHODS FOR INTERACTIVE VIDEO

Our simulations and Internet experiments with a single-proxy system between the U.S and Europe, presented in Chapter 5, clearly proved the advantages of the I-Proxy approach. These simulations and experiments also showed that the location of the I-Proxy played an important role on the overall performance of the system. Given that a large number of I-Proxies can be available at different locations throughout the Internet, we clearly require a method for selecting a good I-Proxy. In this appendix, our goal is to model the dynamics involved in the networks with proxies, and investigate mathematical and practical I-Proxy selection methods.

C.1 Effects of the Location of the I-Proxy

The analysis of Figure 49 reveals two important results:

- Fast retransmission may recover packets faster than early retransmission, however, cannot recover all lost packets.
- Early retransmission can recover any lost packet but may not be able to recover all of them on time.

Generally speaking, congestion and packet losses occur on the bottleneck link of the path between two ends. As mentioned previously, the node that is closest to where the packets are dropped can notify the server and request a retransmission before all the other nodes. Let us exemplify this point on the single Internet path sketched in Figure 79. Here, assume that the bottleneck is the link between the first and second nodes. In this case, locating the I-Proxy on the second node minimizes the loss detection time, and consequently, the error-recovery time.

Yet, an opposite scenario is when the congested link is closer to the client than it is to the server. An example is sketched in Figure 80, where the link between the fourth and fifth

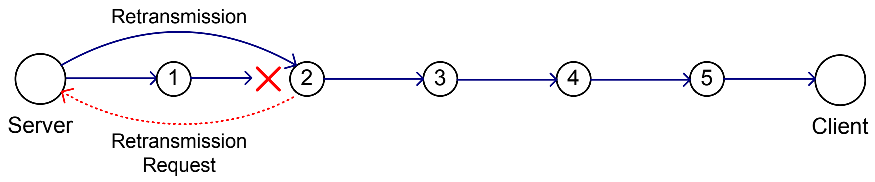


Figure 79: Fast retransmission by the I-Proxy.

nodes is congested and causing the majority of the packet losses. In this case, it is a better practice to place the I-Proxy on the fourth node rather than the fifth one. The reason being that packets will eventually arrive at the fourth node from where a quick recovery can be done for those packets that get lost between the I-Proxy and client. As a matter of fact, locating the I-Proxy on the third node would unnecessarily increase the error-recovery time, whereas locating on the fifth node would render almost all of the advantages of the I-Proxy approach useless. It is interesting to note that the scenario sketched in Figure 79 is the exact replica of the one sketched in Figure 80 with the server and client are switched. In two-way video applications, this implies that the I-Proxy selected for the video transmission in one direction is a good choice for the video transmission in the reverse direction as well.

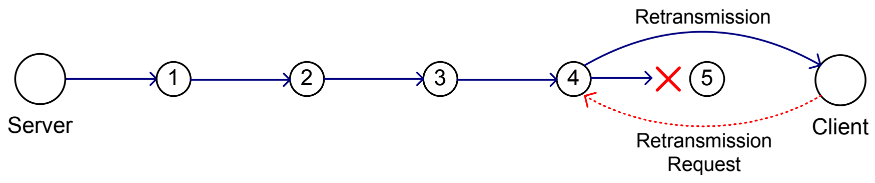


Figure 80: Early retransmission by the I-Proxy.

These two examples clearly show why an intelligent I-Proxy placement/selection is essential to the success of our approach. Unfortunately, there is no immediate solution to this problem. The network conditions should be carefully evaluated, and a proper decision should be given accordingly. In the next section, we look into this problem and provide a solution based on a mathematical model.

C.2 Model-Based I-Proxy Selection

In this section, we derive mathematical expressions to model the dynamics of fast and early retransmission mechanisms. In our formulations, we make certain assumptions regarding the knowledge of path conditions between the end-users and the candidate I-Proxies such as packet delay and loss characteristics. While such information is difficult to collect in practice, we believe that the following discussion provides an insight for the I-Proxy selection problem. Based on our findings, we will propose a practical approach in Section C.3. We start our discussion by introducing the notation.

C.2.1 Definitions

We characterize a communication network by a set of I-Proxy nodes, $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_n$, a server $\mathcal{S}$ and a client $\mathcal{C}$. The delay of a packet transmitted the k^{th} time from node $\mathcal{U}$ destined to node $\mathcal{V}$ via node $\mathcal{X}$ is denoted by $t^k_{\mathcal{U}\rightarrow\mathcal{X}\rightarrow\mathcal{V}}$, which includes the processing, transmission, propagation as well as the queuing delays. A sample network with three I-Proxies is sketched in Figure 81. When a user logs into the system, a list of the available I-Proxies and relevant statistics are transmitted to the user.

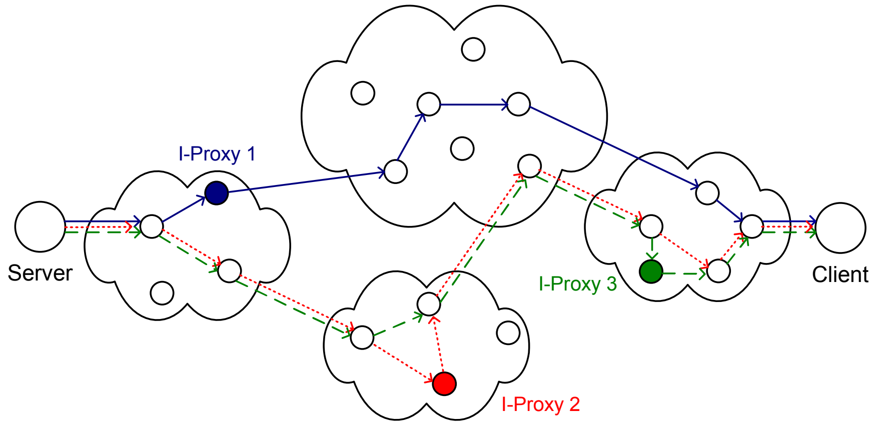


Figure 81: Sample network topology with three I-Proxies.

A critical variable that affects the on-time delivery performance of an application is the delay jitter. Unfortunately, it is not possible to accommodate an excessively-delayed packet in conversational applications due to the strict delay requirements. Hence, the server has to deliver every packet to the client within a short amount of time after the packet is generated. In the sequel, we refer to the maximum tolerable delay by the application as T_{max}.

Without loss of generality, we assume that any packet delivered within T_{max} improves the video quality rendered at the client[1]. Naturally, the path that has the best on-time delivery performance delivers the highest quality of video on the average. Hence, the I-Proxy should be selected such that the path going through it has the highest on-time delivery rate. Explicitly, this selection process requires the evaluation of each possible path between the server and client. For this purpose, we develop a quality metric, *i.e.*, an objective function, that computes the on-time delivery rate based on the network conditions between the server/client and each I-Proxy. The quality of the path going through I-Proxy $\mathcal{N}_k$ is denoted by $\mathcal{Q}_{\mathcal{N}_k}$. To make the analysis tractable, we limit the maximum number of retransmissions to one for each packet, which is at the end a reasonable assumption for interactive video applications.

C.2.2 Objective Function

First, we compute the on-time delivery rate of the direct path between the server and client, *i.e.*, no I-Proxy is used. We express the quality of the direct path as

$$\mathcal{Q}_{\emptyset} = P\left\{t^1_{\mathcal{S}\to\mathcal{C}} \le T_{max}\right\} + P\left\{t^1_{\mathcal{S}\to\mathcal{C}} = \infty\right\} \times P\left\{t_{E2E-ARQ} \le T_{max}\right\}, \tag{107}$$

where $P\{t_{E2E-ARQ} \le T_{max}\}$ represents the probability of an on-time end-to-end retransmission. This probability can be written as follows:

$$P\left\{t_{E2E-ARQ} \le T_{max}\right\} = P\left\{d_{\mathcal{C}} + t^1_{\mathcal{C}\to\mathcal{S}} + t^2_{\mathcal{S}\to\mathcal{C}} \le T_{max} | t^1_{\mathcal{S}\to\mathcal{C}} = \infty\right\}. \tag{108}$$

In (108), $d_{\mathcal{C}}$ is the difference between the time of the initial transmission at the server and the time a retransmission request is sent by the client. The value of $d_{\mathcal{C}}$ depends on the delay and congestion level between the server and client as well as the agility of the RTO estimator employed by the application. Note that in (108), we use a conditional probability to compute the probability of a successful end-to-end retransmission. This is particularly important since packet delays and loss events are correlated in the short term (See Chapters 6 and 7 for details). Avoiding this conditional would lead to overestimating this probability.

[1]Due to the interdependency relations among video packets, the decoder may not be able to decode a video packet even if it is received before its decoding deadline.

Now, we can generalize (107) to the case when an I-Proxy is used between the server and client. Denoting the durations of fast and early retransmissions by t_{F-ARQ} and t_{E-ARQ}, respectively, we can express the quality of the path going through node $\mathcal{N}_k$ as

$$\begin{aligned} \mathcal{Q}_{\mathcal{N}_k} &= P\left\{t^1_{\mathcal{S}\to\mathcal{N}_k\to\mathcal{C}} \le T_{max}\right\} \\ &\quad + P\left\{t^1_{\mathcal{S}\to\mathcal{N}_k} = \infty\right\} \times P\left\{t_{F-ARQ} \le T_{max}\right\} \\ &\quad + P\left\{t^1_{\mathcal{S}\to\mathcal{N}_k} < \infty\right\} \times P\left\{t^1_{\mathcal{N}_k\to\mathcal{C}} = \infty\right\} \times P\left\{t_{E-ARQ} \le T_{max}\right\}. \end{aligned} \tag{109}$$

The probabilities of successful fast and early retransmissions are given by

$$P\left\{t_{F-ARQ} \le T_{max}\right\} = P\left\{d_{\mathcal{N}_k} + t^1_{\mathcal{N}_k\to\mathcal{S}} + t^2_{\mathcal{S}\to\mathcal{N}_k\to\mathcal{C}} \le T_{max} | t^1_{\mathcal{S}\to\mathcal{N}_k} = \infty\right\} \tag{110}$$

and

$$P\left\{t_{E-ARQ} \le T_{max}\right\} = P\left\{d_{\mathcal{C}} + t^1_{\mathcal{C}\to\mathcal{N}_k} + t^2_{\mathcal{N}_k\to\mathcal{C}} \le T_{max} | t^1_{\mathcal{N}_k\to\mathcal{C}} = \infty\right\}, \tag{111}$$

respectively. It is not surprising that as the I-Proxy gets closer the server, the chance of successful fast retransmission increases while the early retransmission capability gets less viable. The opposite is true if the I-Proxy gets closer to the client, as suggested by Figures 79 and 80. Note that (111) assumes that the packet requested by the client is available at the I-Proxy. This assumption is not strictly true. However, it is likely that the I-Proxy will have already received and cached the packet by the time it receives a retransmission request from the client.

Recall that at the beginning of this section, we assumed that packet delay and loss characteristics were known for all paths between the server and client. Armed with this information and by incorporating the statistical models for the specific RTO estimator that is used by the application, one can evaluate each possible path between the server and client. The goal is to select the path that has the highest average quality metric.

$$k^* = \arg\max_k E\left\{\mathcal{Q}_{\mathcal{N}_k}\right\} \quad k = \{\emptyset, 1, 2, \ldots, n\} \tag{112}$$

In selecting the I-Proxy based on (112), we very likely may face two main challenges: First, it is a difficult task to gather the required statistics, particularly when the network conditions change rapidly. Second, service providers may allocate a large number of

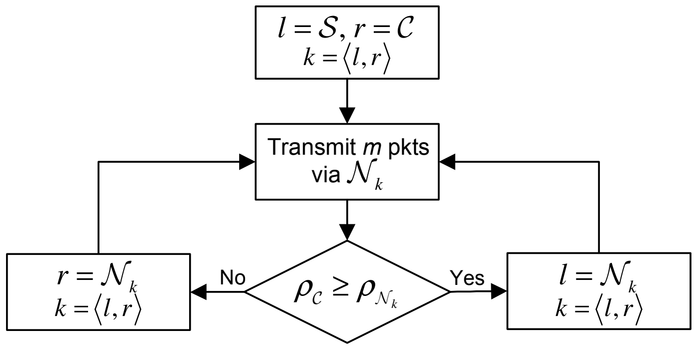
I-Proxies throughout the Internet. In practice, it is neither feasible nor necessary to evaluate (112) for each I-Proxy. In the light of our findings, we next propose a practical approach to address these two concerns.

C.3 Practical I-Proxy Selection

To date, several metaheuristic methods have been developed for solving combinatorial and global optimization problems. A popular and successful one is the variable neighborhood search method, where the basic idea is to start with a feasible solution and try to find a better one through a local search. Provided that a good neighborhood structure is defined, one can obtain near-optimal solutions. However, constructing a good neighborhood structure is particularly difficult in our case, since our problem domain, *i.e.*, the Internet, is a huge diverse environment with time-varying characteristics. As a result, the chances are the solution easily gets trapped in a local optima.

To circumvent this potential drawback, we apply the basic idea of binary search. In this approach, we initially start direct video transmission from the server to the client. After a certain number of packets, say m, are transmitted, we count the number of retransmission requests received, and divide it by m to compute the retransmission request rate of the client, which is denoted by $\rho_{\mathcal{C}}$. Unless the retransmission request rate is negligibly small, the server selects an I-Proxy that is approximately in the middle of the path between the server and client, and starts relaying the video packets over it. Here, selecting a node in the middle of an Internet path stands for finding a node that is network-wise equally close to both ends. If we examine (109) carefully, we infer that selecting the mid node is the best action in practice, if we do not have enough information whereabouts of the congestion. In the flowchart sketched in Figure 82, the selection process is represented by $\langle l, r \rangle$, where l and r are pointers to the boundary nodes.

Once m packets are transmitted over the selected I-Proxy, the server counts the number of individual retransmission requests. If the client reports a larger number of lost packets compared to the I-Proxy, the server infers that majority of packet losses occur between the I-Proxy and client. In this case, the server selects a new I-Proxy that is in the middle of the path between the current I-Proxy and client. However, if the I-Proxy reports more losses, a new I-Proxy that is in the middle of the path between the server and current I-Proxy is selected.



$l = \mathcal{S}, r = \mathcal{C}$ $k = \langle l, r \rangle$
Transmit m pkts via $\mathcal{N}_k$
$\rho_{\mathcal{C}} \geq \rho_{\mathcal{N}_k}$
No
Yes
$r = \mathcal{N}_k$ $k = \langle l, r \rangle$
$l = \mathcal{N}_k$ $k = \langle l, r \rangle$

Figure 82: Flowchart for the binary-search based I-Proxy selection method. $\rho_{\mathcal{N}_k}$ denotes the retransmission request rate of the current I-Proxy.

An important advantage of this approach is that upon a change in the location of the congestion, a new more appropriate I-Proxy is selected through accumulated statistics.

C.4 Simulation Results and Performance Analysis

In this section, we evaluate the performance of three different video transmission schemes: (*i*) direct video transmission between the server and client, (*ii*) video transmission over a randomly selected I-Proxy, and (*iii*) video transmission over an I-Proxy selected based on the method described in Section C.3. To this effect, we generated a moderate-sized Internet topology based on the Georgia Tech Internetwork Topology Models [104] and simulated it with *ns*-2 network simulator [134]. On this topology, we selected a server and a client that were 10-hop away from each other. Data and video flows were attached to all stub-domain nodes to generate background traffic. For RTO estimation, we adopted the method proposed in Section 5.3.

The video sequence we used in the simulations (Foreman, 352×288) was encoded offline by a standard H.264 codec [120] at 600 Kbps and 20 frames per second. To maintain a good interactivity in our application, we limited the buffering at the client side to 200 ms. That is, any packet that could not be delivered within 200 ms was considered late and not displayed. When we tested the direct video transmission, we observed that one-way delay between the server and client averaged over 100 ms, which rendered end-to-end retransmissions impractical. Without any retransmission, only 96.2% of the packets were delivered on time for decoding, which actually produced an unsatisfactory video quality at 36.1 dB (3.8 dB lower than the original quality). As our next step, we introduced a varying number I-Proxies on the path between the server and client to enable the ARQ capability, and evaluated the random and binary-search based I-Proxy selection methods. We present the results in Figure 83.

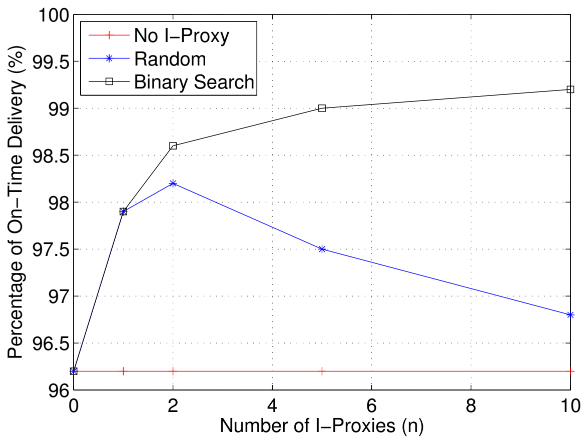


Figure 83: Comparison of different I-Proxy selection methods.

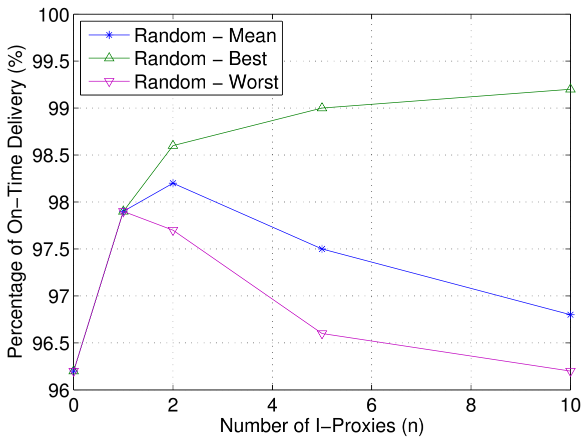


Figure 84: Performance variation of the random I-Proxy selection method.

With the introduction of the first I-Proxy, the percentage of the on-time packets increases by 1.7% for both selection methods. At the client side, this translates to 1.6 dB improvement in the average quality. When the second I-Proxy is introduced into the network, we observe that the binary-search based selection method provides a larger increase in the on-time delivery performance than the random selection method. This is mainly because that binary-search based selection method always selects the best I-Proxy, while random selection has an equal chance of selecting any available I-Proxy. In fact, as the number of available I-Proxies increases, the random selection method starts performing worse on the average. In opposite, each additional I-Proxy continuously improves the performance of the binary-search based selection method. When all the intermediate nodes between the server and client can act as an I-Proxy, *i.e.*, when $n = 10$, we achieve an on-time delivery performance of 99.2%, which produces an average video quality of 39.0 dB at the client[2,3].

As mentioned in Section C.3, the binary-search based I-Proxy selection method is network adaptive. Congestions are detected through observations, and a new selection is made if necessary. In opposite, random selection does not adapt to network changes, and may or may not do a good initial selection. If an inappropriate I-Proxy selection is made, the system performance can be severely impaired. Figure 84 clearly shows that the performance may even deteriorate to a point where the system performance equals the case when no I-Proxy is used.

C.5 Conclusions

In this appendix, we addressed the problem of I-Proxy selection, and studied both mathematical and practical solution approaches. With examples and simulations, we demonstrated the superiority of our network-adaptive I-Proxy selection method over the naive random I-Proxy selection method. The main result of this study is that using I-Proxies is an effective way for providing more robust and better error-control capabilities for delay-sensitive applications, and its advantages can be best exploited with a proper I-Proxy selection.

[2]The remaining 0.8% of the packets are late for decoding mainly due to long-lasting congestions and imperfections in the RTO estimation.

[3]Note that we use only one I-Proxy at a given time regardless of the number of available I-Proxies.

REFERENCES

[1] Transmission control protocol. [Online]. Available: http://www.ietf.org/rfc/rfc793.txt (Accessed Sept., 2006)

[2] A. Goel, C. Krasic, K. Li, and J. Walpole, "Supporting low latency TCP-based media streams," in *10th IEEE Int. Wksp. Quality of Service*, 2002.

[3] T. Kim and M. H. Ammar, "Optimal quality adaptation for scalable encoded video," *IEEE J. Select. Areas Commun.*, vol. 23, no. 2, pp. 344–356, Feb. 2005.

[4] User datagram protocol. [Online]. Available: http://www.ietf.org/rfc/rfc768.txt (Accessed Sept., 2006)

[5] Problem statement for the datagram congestion control protocol (DCCP). [Online]. Available: http://www.ietf.org/rfc/rfc4336.txt (Accessed Sept., 2006)

[6] V. Paxson, "End-to-end Internet packet dynamics," *IEEE/ACM Trans. Networking*, vol. 7, no. 3, pp. 277–292, 1999.

[7] D. Loguinov and H. Radha, "End-to-end Internet video traffic dynamics: Statistical study and analysis," in *IEEE Int. Conf. Computer Communications (INFOCOM)*, 2002.

[8] Y. Wang, J. Ostermann, and Y. Q. Zhang, *Video Processing and Communications.* Prentice Hall, 2002.

[9] J.-C. Bolot, S. Fosse-Parisis, and D. F. Towsley, "Adaptive FEC-based error control for Internet telephony," in *IEEE Int. Conf. Computer Communications (INFOCOM)*, 1999.

[10] B. Girod and N. Farber, *Wireless Video*, M.-T. S. A. Reibman, Ed. Marcel Dekker, 2000.

[11] W. Kumwilaisak, J. Kim, and C.-C. J. Kuo, "Video transmission over wireless fading channels with adaptive FEC," in *Picture Coding Symposium*, 2001.

[12] M. Elaoud and P. Ramanathan, "Adaptive use of error-correcting codes for real-time communication in wireless networks," in *IEEE Int. Conf. Computer Communications (INFOCOM)*, 1998.

[13] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," *IEEE Trans. Inform. Theory*, vol. 42, no. 6, pp. 1737–1744, Nov. 1996.

[14] U. Horn, K. Stuhlmuller, M. Link, and B. Girod, "Robust Internet video transmission based on scalable coding and unequal error protection," *Image Communication*, vol. 15, no. 1-2, pp. 77–94, Sept. 1999.

[15] E. Mohr, E. A. Riskin, and R. E. Ladner, “Unequal loss protection: Graceful degradation of image quality over packet erasure channels through forward error correction,” *IEEE J. Select. Areas Commun.*, vol. 18, no. 6, pp. 819–829, June 2000.

[16] J. Kim, R. M. Mersereau, and Y. Altunbasak, “Error-resilient image and video transmission over the Internet using unequal error protection,” *IEEE Trans. Image Processing*, vol. 12, no. 2, pp. 121–131, 2003.

[17] J. H. Kim, R. M. Mersereau, and Y. Altunbasak, “Bit-plane-wise unequal error protection for Internet video applications,” in *IEEE Int. Conf. Communications (ICC)*, 2002.

[18] G. Al-Regib and Y. Altunbasak, “An unequal error protection method for packet loss resilient 3-D mesh transmission,” in *IEEE Int. Conf. Computer Communications (INFOCOM)*, 2002.

[19] J. Kim, R. M. Mersereau, and Y. Altunbasak, “A multiple-substream unequal error-protection and error-concealment algorithm for SPIHT-coded video bitstreams,” *IEEE Trans. Image Processing*, vol. 13, no. 12, pp. 1547–1553, 2004.

[20] G. Al-Regib, Y. Altunbasak, and J. Rossignac, “An unequal error protection method for progressively transmitted 3-D models,” *IEEE Trans. Multimedia*, vol. 7, no. 4, pp. 766–776, 2005.

[21] C. Papadopoulos and G. M. Parulkar, “Retransmission-based error control for continuous media applications,” *ACM Int. Wksp. Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 1996.

[22] B. J. Dempsey, J. Liebeherr, and A. C. Weaver, “On retransmission-based error control for continuous media traffic in packet-switching networks,” *Computer Networks and ISDN Systems*, vol. 28, no. 5, pp. 719–736, 1996.

[23] B. Mukherjee and T. Brecht, “Time-lined TCP for the TCP-friendly delivery of streaming media,” in *IEEE Int. Conf. Network Protocols (ICNP)*, 2000.

[24] P. Hurley, J. Boudec, P. Thiran, and M. Kara, “ABE: Providing a low-delay service within best-effort,” *IEEE Network*, vol. 15, no. 3, pp. 60–69, 2001.

[25] H. Liu and M. E. Zarki, “Performance of H.263 video transmission over wireless channels using hybrid ARQ,” *IEEE J. Select. Areas Commun.*, vol. 15, no. 9, pp. 1775–1786, 1997.

[26] R. H. Deng and M. L. Lin, “A type-I hybrid ARQ system with adaptive code rates,” *IEEE Trans. Commun.*, vol. 43, no. 2/3/4, pp. 733–737, 1995.

[27] Y. Wang and Q.-F. Zhu, “Error control and concealment for video communication: A review,” *Proc. IEEE*, vol. 86, no. 5, pp. 974–997, May 1998.

[28] J. Wen and J. D. Villasenor, “Reversible variable length codes for efficient and robust image and video coding,” in *IEEE Data Compression Conference (DCC)*, 1998.

[29] Y. Wang, S. Wenger, J. Wen, and A. K. Katsaggelos, “Review of error resilient coding techniques for real-time video communications,” *IEEE Signal Processing Mag.*, vol. 17, no. 4, pp. 61–82, July 2000.

[30] G. Cote, S. Shirani, and F. Kossentini, "Optimal mode selection and synchronization for robust video communications over error prone networks," *IEEE J. Select. Areas Commun.*, vol. 18, no. 6, pp. 952–965, 2000.

[31] R. Zhang, S. L. Regunathan, and K. Rose, "Video coding with optimal inter/intra-mode switching for packet loss resilience," *IEEE J. Select. Areas Commun.*, vol. 18, no. 6, pp. 966–976, 2000.

[32] G. D. L. Reyes, A. Reibman, S. Chang, and J. Chuang, "Error-resilient transcoding for video over wireless channels," *IEEE Trans. Multimedia*, vol. 18, pp. 1063–1074, June 2000.

[33] B. Yan and K. Ng, "A survey on the techniques for the transport of MPEG-4 video over wireless networks," *IEEE Trans. Consumer Electron.*, vol. 48, no. 4, pp. 863–873, Nov. 2002.

[34] V. Hardman, M. A. Sasse, M. Handley, and A. Watson, "Reliable audio for use over the Internet," *Proceedings of INET*, 1995.

[35] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet loss recovery techniques for streaming audio," *IEEE Network*, vol. 12, no. 5, pp. 40–48, 1998.

[36] B. Wah, X. Su, and D. Lin, "A survey of error-concealment schemes for real-time audio and video transmissions over the Internet," in *IEEE Int. Symp. Multimedia Software Engineering*, 2000.

[37] Y. C. Lee and Y. Altunbasak, "Spatial error concealment using multi-frame recovery principle for MPEG-coded video delivery over error-prone networks," in *IEEE Int. Conf. Information, Communications, and Signal Processing (ICICS)*, 2001.

[38] Y. C. Lee, Y. Altunbasak, and R. M. Mersereau, "A temporal error concealment method for MPEG-coded video using a multi-frame boundary matching algorithm," in *IEEE Int. Conf. Image Processing (ICIP)*, 2001.

[39] ——, "Multiframe error concealment for MPEG-coded video delivery over error-prone networks," *IEEE Trans. Image Processing*, vol. 11, no. 11, pp. 1314–1331, 2002.

[40] D. Taubman and A. Zakhor, "Multirate 3-D subband coding of video," *IEEE Trans. Image Processing*, vol. 3, no. 5, pp. 572–588, 1994.

[41] M. van der Schaar and H. Radha, "Unequal packet loss resilience for fine-granular-scalability video," *IEEE Trans. Multimedia*, vol. 3, no. 4, pp. 381–394, Dec. 2001.

[42] V. K. Goyal, "Multiple description coding: Compression meets the network," *IEEE Signal Processing Mag.*, vol. 18, no. 5, pp. 74–93, 2001.

[43] J. G. Apostolopoulos, "Reliable video communication over lossy packet networks using multiple state encoding and path diversity," in *Visual Communications and Image Processing (VCIP)*, 2001.

[44] Y. C. Lee, J. Kim, Y. Altunbasak, and R. M. Mersereau, "Layered coding vs. multiple description coding for video over error-prone networks," *EURASIP Signal Processing: Image Communication*, vol. 18, no. 5, pp. 337–356, 2003.

[45] Y. C. Lee, Y. Altunbasak, and R. M. Mersereau, “An integrated application of multiple description transform coding and error concealment for error-resilient video streaming,” *EURASIP Signal Processing: Image Communication*, vol. 18, no. 10, pp. 957–970, 2003.

[46] A. Reibman, H. Jafarkhani, M. Orchard, and Y. Wang, “Performance of multiple description coders on a real channel,” in *IEEE Int. Conf. Acoust. Speech Sign. Processing (ICASSP)*, 1999.

[47] M. Podolsky, S. McCanne, and M. Vetterli, “Soft ARQ for layered streaming media,” *Technical Report UCB/CSD-98-1024, University of California, Computer Science Division*, 1998.

[48] Z. Miao and A. Ortega, “Optimal scheduling for streaming of scalable media,” in *Asilomar Conf. Signals, Systems, and Computers*, 2000.

[49] ——, “Expected run-time distortion based scheduling for delivery of scalable media,” in *Packet Video Wksp.*, 2002.

[50] ——, “Fast adaptive media scheduling based on expected run-time distortion,” in *Asilomar Conf. Signals, Systems, and Computers*, 2002.

[51] P. Decuetos and K. Ross, “Optimal streaming of layered video: Joint scheduling and error concealment,” in *ACM Multimedia*, 2003.

[52] P. A. Chou and Z. Miao, “Rate-distortion optimized streaming of packetized media,” *Microsoft Research Technical Report MSR-TR-2001-35*, 2001.

[53] J. Chakareski, P. A. Chou, and B. Girod, “Computing rate-distortion optimized policies for hybrid receiver/sender driven streaming of multimedia,” in *Asilomar Conf. Signals, Systems, and Computers*, 2002.

[54] ——, “Rate-distortion optimized streaming from the edge of the network,” in *IEEE Wksp. Multimedia Signal Processing (MMSP)*, 2002.

[55] P. A. Chou and A. Sehgal, “Rate-distortion optimized receiver-driven streaming over best-effort networks,” in *Packet Video Wksp.*, 2002.

[56] A. Sehgal and P. A. Chou, “Cost-distortion optimized streaming media over DiffServ networks,” in *IEEE Int. Conf. Multimedia and Expo (ICME)*, 2002.

[57] P. A. Chou and Z. Miao, “Rate-distortion optimized streaming of packetized media,” *IEEE Trans. Multimedia*, vol. 8, no. 2, pp. 390–404, Apr. 2006.

[58] M. Kalman, E. Steinbach, and B. Girod, “Rate-distortion optimized video streaming with adaptive playout,” in *IEEE Int. Conf. Image Processing (ICIP)*, 2002.

[59] M. Kalman, P. Ramanathan, and B. Girod, “Rate-distortion optimized streaming with multiple deadlines,” in *IEEE Int. Conf. Image Processing (ICIP)*, 2003.

[60] J. Chakareski and B. Girod, “Rate-distortion optimized packet scheduling and routing for media streaming with path diversity,” in *IEEE Data Compression Conference (DCC)*, 2003.

[61] ——, “Server diversity in rate-distortion optimized media streaming,” in *IEEE Int. Conf. Image Processing (ICIP)*, 2003.

[62] S. H. Kang and A. Zakhor, “Packet scheduling algorithm for wireless video streaming,” in *Packet Video Wksp.*, 2002.

[63] A. C. Begen, Y. Altunbasak, , and O. Ergun, “Multi-path selection for multiple description encoded video streaming,” in *IEEE Int. Conf. Communications (ICC)*, 2003.

[64] A. C. Begen, Y. Altunbasak, and O. Ergun, “Fast heuristics for multi-path selection for multiple description encoded video streaming,” in *IEEE Int. Conf. Multimedia and Expo (ICME)*, 2003.

[65] A. C. Begen, Y. Altunbasak, O. Ergun, and M. A. Begen, “Real-time multiple description and layered encoded video streaming with optimal diverse routing,” in *IEEE Int. Symp. Computers and Communications (ISCC)*, 2003.

[66] A. C. Begen, Y. Altunbasak, O. Ergun, and M. H. Ammar, “Multi-path selection for multiple description encoded video streaming over overlay networks,” *EURASIP Signal Processing: Image Communication*, vol. 20, no. 1, pp. 39–60, 2005.

[67] S. McCanne, V. Jacobson, and M. Vetterli, “Receiver-driven layered multicast,” in *ACM SIGCOMM*, 1996.

[68] X. Li, S. Paul, P. Pancha, and M. Ammar, “Layered video multicast with retransmission (LVMR): Evaluation of error recovery schemes,” in *ACM Int. Wksp. Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 1997.

[69] W. Tan and A. Zakhor, “Multicast transmission of scalable video using receiver-driven hierarchical FEC,” in *Packet Video Wksp.*, 1999.

[70] J. Apostolopoulos, W. Tan, S. Wee, and G. Wornell, “Modeling path diversity for multiple description video communication,” in *IEEE Int. Conf. Acoust. Speech Sign. Processing (ICASSP)*, 2002.

[71] S. Lin, S. Mao, Y. Wang, and S. Panwar, “A reference picture selection scheme for video transmission over ad-hoc networks using multiple paths,” in *IEEE Int. Conf. Multimedia and Expo (ICME)*, 2001.

[72] S. Mao, S. Lin, Y. Wang, and S. Panwar, “Reliable transmission of video over ad-hoc networks using automatic repeat request and multi-path transport,” in *IEEE Vehicular Technology Conference (VTC)*, 2001.

[73] R. Chow, C. Lee, and J. C. Liu, “Traffic dispersion strategies for multimedia streaming,” in *8th IEEE Wksp. on Future Trends of Distributed Computing Systems*, 2001.

[74] Y. J. Liang, E. Setton, and B. Girod, “Channel-adaptive video streaming using packet path diversity and rate-distortion optimized reference picture selection,” in *IEEE Wksp. Multimedia Signal Processing (MMSP)*, 2002.

[75] Y. Wang, S. Panwar, S. Lin, and S. Mao, “Wireless video transport using path diversity: Multiple description vs. layered coding,” in *IEEE Int. Conf. Image Processing (ICIP)*, 2002.

[76] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee, “On multiple description streaming with content delivery networks,” in *IEEE Int. Conf. Computer Communications (INFOCOM)*, 2002.

[77] T. Nguyen and A. Zakhor, “Path diversity with forward error correction (PDF) system for packet switched networks,” in *IEEE Int. Conf. Computer Communications (INFOCOM)*, 2003.

[78] R. Karrer and T. Gross, “Multipath streaming in best-effort networks,” in *IEEE Int. Conf. Communications (ICC)*, 2003.

[79] S. Mao, S. Lin, S. S. Panwar, Y. Wang, and E. Celebi, “Video transport over ad hoc networks: Multistream coding with multipath transport,” *IEEE J. Select. Areas Commun.*, vol. 21, no. 10, pp. 1721–1737, 2003.

[80] S. Mao, D. Bushmitch, S. Narayanan, and S. Panwar, “MRTP: A multi-flow real-time transport protocol for ad hoc networks,” in *IEEE Vehicular Technology Conference (VTC)*, 2003.

[81] H.-M. Hang and J.-J. Chen, “Source model for transform video coder and its application – part i: Fundamental theory,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 2, pp. 287–298, 1997.

[82] Z. He and S. K. Mitra, “A unified rate-distortion analysis framework for transform coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 12, pp. 1221–1236, 2001.

[83] M. Dai and D. Loguinov, “Analysis of rate-distortion functions and congestion control in scalable Internet video streaming,” in *ACM Int. Wksp. Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2003.

[84] M. Dai, D. Loguinov, and H. Radha, “Statistical analysis and distortion modeling of MPEG-4 FGS,” in *IEEE Int. Conf. Image Processing (ICIP)*, 2003.

[85] The MPEG software simulation group website (MSSG). [Online]. Available: http://www.mpeg.org/MPEG/MSSG/ (Accessed Sept., 2006)

[86] T. Chiang and Y. Zhang, “A new rate control scheme using quadratic rate distortion model,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 1, pp. 246–250, 1997.

[87] J. Kim, R. M. Mersereau, and Y. Altunbasak, “An integrated multiple-substream unequal error protection and error concealment algorithm for Internet video applications,” in *IEEE Int. Conf. Image Processing (ICIP)*, 2002.

[88] V. A. Vaishampayan, “Design of multiple description scalar quantizers,” *IEEE Trans. Inform. Theory*, vol. 39, no. 3, pp. 821–834, 1993.

[89] Y. C. Lee, Y. Altunbasak, and R. M. Mersereau, “A two-stage multiple description video coder with drift-preventing motion compensated prediction,” in *IEEE Int. Conf. Image Processing (ICIP)*, 2002.

[90] ——, “A collaborative multiple description transform coding and statistical error concealment method for error resilient video streaming over noisy channels,” in *IEEE Int. Conf. Acoust. Speech Sign. Processing (ICASSP)*, 2002.

[91] N. R. Council, *Looking Over the Fence at Networks: A Neighbor's View of Networking Research.* National Academies Press, 2001.

[92] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," in *18th ACM Symposium on Operating Systems Principles*, 2001.

[93] J. Touch and S. Hotz, "The X-Bone," in *Third Global Internet Mini-Conference at GLOBECOM*, 1998.

[94] Y.-H. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *ACM SIGMETRICS*, 2000.

[95] S. Lee, S. Das, G. Pau, and M. Gerla, "A hierarchical multipath approach to QoS routing: Performance and cost evaluation," in *IEEE Int. Conf. Communications (ICC)*, 2003.

[96] J. Apostolopoulos and S. Wee, "Unbalanced multiple description video communication using path diversity," in *IEEE Int. Conf. Image Processing (ICIP)*, 2001.

[97] Y. Wang, M. Claypool, and Z. Zuo, "An empirical study of realvideo performance across the Internet," in *ACM SIGCOMM Internet Measurement Wksp.*, 2001.

[98] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modelling of the temporal dependence in packet loss," in *IEEE Int. Conf. Computer Communications (INFOCOM)*, 1999.

[99] W. Jiang and H. Schulzrinne, "Modeling of packet loss and delay and their effects on real-time multimedia service quality," in *ACM Int. Wksp. Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2000.

[100] Y. C. Lee, Y. Altunbasak, and R. M. Mersereau, "Optimal packet scheduling for multiple description coded video transmissions over lossy networks," in *IEEE Global Communications Conference (GLOBECOM)*, 2003.

[101] A. Mukherjee, "On the dynamics and significance of low frequency components of Internet load," University of Pennsylvania, Tech. Rep. MS-CIS-92-83, 1992.

[102] RTP: A transport protocol for real-time applications. [Online]. Available: http://www.ietf.org/rfc/rfc1889.txt (Accessed Sept., 2006)

[103] RTP payload format for MPEG1/MPEG2 video. [Online]. Available: http://www.ietf.org/rfc/rfc2250.txt (Accessed Sept., 2006)

[104] E. W. Zegura, K. L. Calvert, and M. J. Donahoo, "A quantitative comparison of graph-based models for Internet topology," *IEEE/ACM Trans. Networking*, vol. 5, no. 6, pp. 770–783, 1997.

[105] J. Chen and S.-H. G. Chan, "Multipath routing for video unicast over bandwidth-limited networks," in *IEEE Global Communications Conference (GLOBECOM)*, 2001.

[106] A. C. Begen, Y. Altunbasak, M. R. Civanlar, and G. Gorbil, "High-resolution video streaming in mesh-networked homes," in *IEEE Int. Conf. Image Processing (ICIP)*, 2005.

[107] S. Cass, “Viva mesh Vegas,” *IEEE Spectr.*, vol. 42, no. 1, pp. 48–53, Jan. 2005.

[108] I. F. Akyildiz, X. Wang, and W. Wang, “Wireless mesh networks: a survey,” *Elsevier Computer Networks*, vol. 47, no. 4, pp. 445–487, Mar. 2005.

[109] The dynamic source routing protocol for mobile ad hoc networks (DSR). [Online]. Available: http://www3.ietf.org/proceedings/04mar/I-D/draft-ietf-manet-dsr-09.txt (Accessed May, 2005)

[110] Jini network technology. [Online]. Available: http://www.sun.com/software/jini/ (Accessed May, 2005)

[111] UPnP forum. [Online]. Available: http://www.upnp.org/ (Accessed May, 2005)

[112] Self-organizing neighborhood wireless mesh networks. [Online]. Available: http://research.microsoft.com/mesh/ (Accessed May, 2005)

[113] A. C. Begen, Y. Altunbasak, and M. A. Begen, “Rate-distortion optimized on-demand media streaming with server diversity,” in *IEEE Int. Conf. Image Processing (ICIP)*, 2003.

[114] A. C. Begen, M. U. Demircin, and Y. Altunbasak, “Packet scheduling for multiple description video streaming in multipoint-to-point networks,” in *IEEE Int. Conf. Communications (ICC)*, 2004.

[115] P. Rodriguez, A. Kirpal, and E. Biersack, “Parallel-access for mirror sites in the Internet,” in *IEEE Int. Conf. Computer Communications (INFOCOM)*, 2000.

[116] J. W. Byers, M. Luby, and M. Mitzenmacher, “Accessing multiple mirror sites in parallel: Using tornado codes to speed up downloads,” in *IEEE Int. Conf. Computer Communications (INFOCOM)*, 1999.

[117] A. Majumdar, R. Puri, and K. Ramchandran, “Rate-distortion efficient video transmission from multiple servers,” in *IEEE Int. Conf. Multimedia and Expo (ICME)*, 2002.

[118] SIP: Session initiation protocol. [Online]. Available: http://www.ietf.org/rfc/rfc2543.txt (Accessed Sept., 2006)

[119] M. Puterman, *Markov Decision Processes.* J. Wiley & Sons, 1994.

[120] H.264 AVC reference software. [Online]. Available: http://iphome.hhi.de/suehring/tml/download (Accessed Sept., 2006)

[121] A. C. Begen and Y. Altunbasak, “Videoconferencing over an intermediate-proxy,” in *IEEE Int. Conf. Image Processing (ICIP)*, 2004.

[122] ——, “Timely inference of late/lost packets in real-time streaming applications,” in *Picture Coding Symp. (PCS)*, 2004.

[123] ——, “Estimating packet arrival times in bursty video applications,” in *IEEE Int. Conf. Multimedia and Expo (ICME)*, 2005.

[124] ——, “Proxy-assisted interactive-video services over networks with large delays,” *EURASIP Signal Processing: Image Communication, Special Issue on Video Networking*, vol. 20, no. 8, pp. 755–772, 2005.

[125] A. C. Begen, M. A. Begen, Y. Altunbasak, and M. R. Civanlar, “Proxy selection for interactive video,” in *IEEE Int. Conf. Communications (ICC)*, 2006.

[126] Y. Wang, Z. Zhang, D. Du, and D. Su, “A networkconscious approach to endtoend video delivery over wide area networks using proxy servers,” in *IEEE Int. Conf. Computer Communications (INFOCOM)*, 1998.

[127] S. Sen, J. Rexford, and D. Towsley, “Proxy prefix caching for multimedia streams,” in *IEEE Int. Conf. Computer Communications (INFOCOM)*, 1999.

[128] R. Rejaie, M. Handley, H. Yu, and D. Estrin, “Proxy caching mechanism for multimedia playback streams in the Internet,” in *Int. Web Caching Wksp.*, 1999.

[129] Z. Miao and A. Ortega, “Proxy caching for efficient video services over the Internet,” in *Packet Video Wksp.*, 1999.

[130] F. Hartanto, M. Reisslein, and K. W. Ross, “Interactive video streaming with proxy servers,” *Information Sciences, an International Journal*, 2001.

[131] C. Venkatramani, O. Verscheure, P. Frossard, and K.-W. Lee, “Optimal proxy management for multimedia streaming in content distribution networks,” in *ACM Int. Wksp. Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2002.

[132] L. Gao, Z.-L. Zhang, and D. Towsley, “Proxy-assited techniques for delivering continous multimedia streams,” *IEEE/ACM Trans. Networking*, vol. 11, no. 6, pp. 884–894, Dec. 2003.

[133] B. Ford, P. Srisuresh, and D. Kegel, “Peer-to-peer communication across network address translators,” in *USENIX*, 2005.

[134] S. McCanne and S. Floyd. Network simulator. [Online]. Available: http://www.isi.edu/nsnam/ns (Accessed Sept., 2006)

[135] P. Karn and C. Partridge, “Improving round-trip time estimates in reliable transport protocols,” in *ACM SIGCOMM*, 1987.

[136] A. C. Begen and Y. Altunbasak, “Redundancy-controllable adaptive retransmission timeout estimation for packet video,” in *ACM Int. Wksp. Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2006.

[137] A. C. Begen, M. A. Begen, and Y. Altunbasak, “Predictive modeling of video packet delay in IP networks,” in *IEEE Int. Conf. Image Processing (ICIP)*, 2006.

[138] Q. Li, “Delay characterization and performance control of wide-area networks,” Ph.D. dissertation, Univ. of Delaware, Newark, May 2000.

[139] M. Kalman, E. Steinbach, and B. Girod, “Adaptive media playout for low delay video streaming over error-prone channels,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 6, pp. 841–851, June 2004.

[140] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM*, 1988.

[141] Computing TCP's retransmission timer. [Online]. Available: http://www.ietf.org/rfc/rfc2988.txt (Accessed Sept., 2006)

[142] D. Loguinov and H. Radha, "On retransmission schemes for real-time streaming in the Internet," in *IEEE Int. Conf. Computer Communications (INFOCOM)*, 2001.

[143] M. Allman and V. Paxson, "On estimating end-to-end network parameters," in *ACM SIGCOMM*, 1999.

[144] R. Ludwig and K. Sklower, "The eifel retransmission timer," *SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 3, pp. 17–27, 2000.

[145] Q. Li and D. L. Mills, "Jitter-based delay-boundary prediction of wide-area networks," *IEEE/ACM Trans. Networking*, vol. 9, no. 5, pp. 578–590, Oct. 2001.

[146] L. Ma, G. R. Arce, and K. E. Barner, "TCP retransmission timeout algorithm using weighted medians," *IEEE Signal Processing Lett.*, vol. 11, no. 6, pp. 569–572, June 2004.

[147] R. Sinha and C. Papadopoulos, "An adaptive multiple retransmission technique for continuous media streams," in *ACM Int. Wksp. Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2004.

[148] I. Rhee, "Error control techniques for interactive low bitrate video transmission over the Internet," in *ACM SIGCOMM*, 1998.

[149] M. Kalman and B. Girod, "Modeling the delays of successively-transmitted Internet packets," in *IEEE Int. Conf. Multimedia and Expo (ICME)*, 2004.

[150] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting.* Springer, 2003.

[151] P. Stoica and Y. Selen, "Model-order selection: a review of information criterion rules," *IEEE Signal Processing Mag.*, vol. 21, no. 4, pp. 36–47, July 2004.

[152] Simple network time protocol (SNTP) version 4 for IPv4, IPv6 and OSI. [Online]. Available: http://www.ietf.org/rfc/rfc2030.txt (Accessed Sept., 2006)

[153] R. Singh, A. Ortega, L. Perret, and W. Jiang, "Comparison of multiple description coding and layered coding based on network simulations," in *Visual Communications and Image Processing (VCIP)*, 2000.

[154] M. Gallant and F. Kossentini, "Rate-distortion optimized layered coding with unequal error protection for robust Internet video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 357–372, Mar. 2001.

VITA

Ali C. Begen received his B.S. degree in Electrical and Electronics Engineering from Bilkent University, Ankara, Turkey, in 2001, and the M.S. degree in Electrical and Computer Engineering from Georgia Institute of Technology, Atlanta, GA, in 2002. Since 2001, he has been a member of the Center for Signal and Image Processing at Georgia Tech, where he participated in several government and industry-sponsored research projects in the areas of media delivery over the Internet and wireless networks, networked entertainment, multimedia transport protocols and content distribution. He held an Interim Engineering Intern position at Qualcomm Standards Engineering Group between May and December 2004, where he received *Qualstar Hall of Fame Award* for his contributions to the standards development.

Ali C. Begen is a member of the IEEE and ACM, IEEE Communications and Signal Processing Societies, and Eta Kappa Nu. Since 2002 and 2004, he has been serving as the webmaster on the IEEE Signal Processing Society Image and Multidimensional Signal Processing Technical Committee and IEEE Communications Society Multimedia Communications Technical Committee, respectively. Between 2004 and 2006, he has also served as a member of the Organizing Committee for the IEEE Int. Conf. Image Processing (ICIP) 2006. Ali C. Begen is a TPC member for several international conferences and a reviewer for several journals including the IEEE Trans. Multimedia, IEEE Trans. Speech and Audio Processing, IEEE Trans. Image Processing, IEEE Multimedia, IEEE Trans. Circuits and Systems for Video Technology, IEEE Trans. Mobile Computing, IEEE Jour. Selected Areas in Communications, IEEE Trans. Vehicular Technology, EURASIP Signal Processing: Image Communication, and EURASIP Jour. Wireless Communications and Networking. In 2003, he received the *Best Student-paper Award* at IEEE ICIP in Barcelona, Spain.