Network and Content Adaptive Streaming of Layered-Encoded Video over the Internet

Philippe de Cuetos

Thèse en vue de l'obtention
du diplôme de docteur de
TELECOM PARIS
(Ecole Nationale Supérieure des Télécommunications)

Network and Content Adaptive Streaming of Layered–Encoded Video over the Internet

—

Streaming de Vidéos Encodées en Couches sur Internet
avec Adaptation au Réseau et au Contenu

Philippe de Cuetos

philippe.de-cuetos@eurecom.fr
Institut Eurécom, Sophia–Antipolis

———

Soutenue le 19 Septembre 2003 devant le jury :

S. Tohmé, ENST, président
C. Guillemot, IRISA, rapporteur
P. Frossard, EPFL, rapporteur
T. Turletti, INRIA, examinateur
B. Mérialdo, Institut Eurécom, examinateur
K. W. Ross, Polytechnic Univ., directeur de thèse

Abstract

In this thesis we propose new techniques and algorithms for improving the quality of Internet video streaming applications. We formulate optimization problems and derive control policies for transmission over the current best–effort Internet. This dissertation studies adaptation techniques that jointly adapt to varying network conditions (network–adaptive techniques) and to the characteristics of the streamed video (content–adaptive techniques). These techniques are combined with layered encoding of the video and client buffering. We evaluate their performance based on simulations with network traces (TCP connections) and real videos (MPEG–4 FGS encoded videos).

We first consider the transmission of stored video over a reliable TCP–friendly connection. We compare adding/dropping layers and switching among different versions of the video; we show that the flexibility of layering cannot, in general, compensate for the bitrate overhead over non–layered encoding.

Second, we focus on a new layered encoding technique, Fine–Granularity Scalability (FGS), which has been specifically designed for streaming video. We propose a novel framework for streaming FGS–encoded videos and solve an optimization problem for a criterion that involves both image quality and quality variability during playback. Our optimization problem suggests a real–time heuristic whose performance is assessed over different TCP–friendly protocols. We show that streaming over a highly variable TCP–friendly connection, such as TCP, gives video quality results that are comparable with streaming over smoother TCP–friendly connections. We present the implementation of our rate adaptation heuristic in an MPEG–4 streaming system.

Third, we consider the general framework of rate–distortion optimized streaming. We analyze rate–distortion traces of long MPEG–4 FGS encoded videos, and observe that the semantic content has significant impact on the encoded video properties. From our traces, we investigate optimal streaming at different aggregation levels (images, groups of pictures, scenes); we advocate scene–by–scene optimal adaptation, which gives good quality results with low computational complexity.

Finally, we propose a unified optimization framework for transmission of layered–encoded video over lossy channels. The framework combines scheduling, error protection through Forward Error Correction (FEC) and decoder error concealment. We use results on infinite–horizon average–rewards Markov Decision Processes (MDPs) to find optimal transmission policies with low–complexity and for a wide range of quality metrics. We show that considering decoder error concealment in the scheduling and error correction optimization procedure is crucial to achieving truly optimal transmission.

Résumé

Dans cette thèse nous proposons de nouvelles techniques et de nouveaux algorithmes pour améliorer la qualité des applications de streaming vidéo sur Internet. Nous formulons des problèmes d'optimisation et obtenons des politiques de contrôle pour la transmission sur le réseau Internet actuel sans qualité de service. Cette thèse étudie des techniques qui adaptent la transmission à la fois aux conditions variables du réseau (adaptation au réseau) et aux caractéristiques des vidéos transmises (adaptation au contenu). Ces techniques sont associées au codage en couche de la vidéo et au stockage temporaire de la vidéo au client. Nous évaluons leur performances à partir de simulations avec des traces réseau (connexions TCP) et à partir de vidéos encodées en MPEG–4 FGS.

Nous considérons tout d'abord des vidéos stockées sur un serveur et transmises sur une connexion TCP–compatible sans perte. Nous comparons les mécanismes d'ajout/retranchement de couches et de changement de versions ; nous montrons que la flexibilité du codage en couches ne peut pas compenser, en général, le surcoût en bande passante par rapport au codage vidéo conventionnel.

Deuxièmement, nous nous concentrons sur une nouvelle technique de codage en couches, la scalabilité à granularité fine (dite FGS), qui a été conçue spécifiquement pour le streaming vidéo. Nous proposons un nouveau cadre d'étude pour le streaming de vidéos FGS et nous résolvons un problème d'optimisation pour un critère qui implique la qualité des images et les variations de qualité durant l'affichage. Notre problème d'optimisation suggère une heuristique en temps réel dont les performances sont évaluées sur des protocoles TCP–compatibles différents. Nous montrons que la transmission sur une connexion TCP–compatible très variable, telle que TCP, résulte en une qualité comparable à une transmission sur des connexions TCP–compatibles moins variables. Nous présentons l'implémentation de notre heuristique d'adaptation dans un système de streaming de vidéos MPEG–4.

Troisièmement, nous considérons le cadre d'étude général du streaming optimisé suivant les caractéristiques débit–distorsion de la vidéo. Nous analysons des traces débit–distorsion de vidéos de longue durée encodées en MPEG–4 FGS, et nous observons que le contenu sémantique a un impact important sur les propriétés des vidéos encodées. A partir de nos traces, nous examinons le streaming optimal à différents niveaux d'agrégation (images, GoPs, scènes) ; nous préconisons l'adaptation optimale scène par scène, qui donne une bonne qualité pour une faible complexité de calcul.

Finalement, nous proposons un cadre d'optimisation unifié pour la transmission de vidéos encodées en couches sur des canaux à pertes. Le cadre d'étude proposé combine l'ordonnancement, la protection contre les erreurs par les FEC et la dissimulation d'erreur au décodeur. Nous utilisons des résultats sur les Processus de Décision de Markov (MDPs) à horizon infini et gain moyen, pour trouver des politiques de transmission optimales avec une faible complexité et pour un large éventail de mesures de qualité.

Nous montrons qu'il est crucial de considérer la dissimulation d'erreur au décodeur dans la procédure d'optimisation de l'ordonnancement et de la protection contre les erreurs afin d'obtenir une transmission optimale.

Acknowledgments

Many people have contributed to the accomplishment of this thesis. Some of them deserve a bigger slice of the thanks cake, starting with my advisor Keith Ross who allowed me to do this thesis in very good conditions. I valued his mentorship, his constructive judgment, and I also enjoyed the friendly discussions we had about everything, from movies to technology. I am grateful to Martin Reisslein for giving me the opportunity to change my routine by inviting me to Arizona State University for three months in 2002. I appreciated his enthusiasm and his guidance. I also wish to thank Despina, Jussi and David for their collaboration and their fruitful discussions about my work.

I am indebted to the Région Provence–Alpes–Côte d'Azur and Institut Eurécom for providing the financial support for this thesis, through a partnership with Wimba. I am grateful to the French Telecommunication Research Organism (RNRT) for giving me the opportunity to work with talented people through the research project VISI, especially Philippe Guillotel (Thomson Multimedia), Christine Guillemot (IRISA), Thierry Turletti (INRIA) and Patrick Boissonade (France Telecom R&D). Christine and Thierry, together with Prof. Samir Tohmé, Prof. Bernard Mérialdo, and Prof. Pascal Frossard, have honored me by accepting to be part of the thesis committee.

Finally, I would like to thank warmly all those who supported me during this time, by simply being there and bearing my changes of mood (I know that it has not always been a piece of cake): my family, and Sébastien; Sophie & Nicolas in Marseille; Emmanuelle & Raphaël, and Valérie & Fabien in Toulouse; Séphane, Alexandre & Nicolas here in the French Riviera; Nadia & Guillaume, and Stéphane & Sébastien in Paris; Hanna, Julia, Ike, and Leland in Tempe; Sabine, and Véronique in New York; Maciej in Uppsala; Alain & Claire in Rennes; and, from Eurécom, Ana, Carine, Caroline, and all past and present PhD students I hanged out with — I apologize for not citing everyone but I won't risk to forget anybody.

EURECOM
Sophia Antipolis

Région
Provence-Alpes-Côte d'Azur

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

The Demand for Networked Video Applications

Networked video applications today are still in their infancy. However, as digital video technology progresses and the Internet becomes more and more ubiquitous, the demand for networked video applications is likely to increase significantly in the next few years. Telecommunication operators worldwide are starting to deploy new services to get more revenue from their deployed infrastructures (e.g., xDSL, UMTS, WiFi), manufacturers are constantly creating new innovative products (e.g., cell phones, PDAs, set–top boxes, lightweight laptops), and the battle rages among the top media companies to sell digital multimedia content online (e.g., Pressplay, iTunes, MovieLink). These giant corporations may not change the way people work, communicate and entertain themselves within a year or two — as some investors wrongly thought at the early days of the Internet — but they will succeed eventually, because networked video applications can provide businesses and individuals with a real added value.

For businesses, networked video can provide faster and more efficient communication within companies. For important occasions, executives of some big companies now address all their employees through the company's Intranet. The speech can be watched live, or stored and transmitted on demand at any time of the day, for example to overseas employees. Video communication can also make training more flexible and less expensive than regular training. Thanks to video conferencing and remote collaboration through video, employees can avoid traveling for business, thereby saving money and time. Finally, businesses can use networked video applications to communicate about their products worldwide, directly from their web page. As an example, some companies now advertise the launch of a new product by posting the video of the launch event on the Internet.

Networked video applications also have the potential to improve communication and entertainment

of individuals. In 2002, French people spent on average 3h20 daily watching television[1]. Networked video can provide the user with a broader choice of programs than regular TV, as well as interactive applications and user–centered content, such as user–specific news. Besides TV programs, home video is also a popular application for entertainment (and an important source of revenue for the film industry[2]). Delivering movies on demand over the Internet would allow users to access a wider range of content than video stores, and in a much faster way. Finally, visual communication can enhance the ordinary telephone communication among individuals, for instance allowing remote families to keep in touch visually thanks to videophones.

Most of the previously mentioned applications require significant support from the network infrastructure, as well as specific hardware or software. The current evolution of the Internet and of communication terminals is favorable to networked video applications : these past few years, we have seen the deployment of high–speed Internet connections; also, the processing capacities of terminals (PCs, PDAs) has increased, and several video coding systems and standards have been designed specifically for video streaming. Still, today, Internet video applications are usually of poor quality. Streamed videos are often jerky and their quality can degrade significantly during viewing. The goal of this thesis is to propose new techniques and algorithms for improving the quality of Internet video streaming applications.

1.2 Context of the Thesis and Main Contributions

In this dissertation, we focus on unicast Internet video streaming applications, i.e., videos that are transmitted from one server (or proxy) to one client. We formulate optimization problems and derive control policies for video applications over the current best–effort Internet; we look particularly for low–complexity optimization procedures that are suitable to servers that serve a lot of users simultaneously.

Figure 1.1 gives an overview of a typical Internet video streaming system. At the server, the source video is first encoded. The encoded video images are stored in a file for future transmission, or they can be directly sent to the client in real–time. Adaptive techniques at the server consist in determining the way video packets are sent to the client, as a function of current network conditions and of reports that can be sent back from the client. At the client, video packets are usually temporarily buffered before being sent to the decoder. Finally, after decoding, the video images are displayed to the user.

In this dissertation, we study adaptation techniques that jointly adapt to varying network conditions (network–adaptive techniques) and to the characteristics of the streamed video (content–adaptive techniques). Adaptive techniques are combined with a particular form of video coding (layered encoding),

[1] According to France main audience measure company Médiamétrie (www.mediametrie.fr).

[2] At its peak, home video generated half of the film industry's sales and three quarter of its profits (The Economist, 19/09/2002).

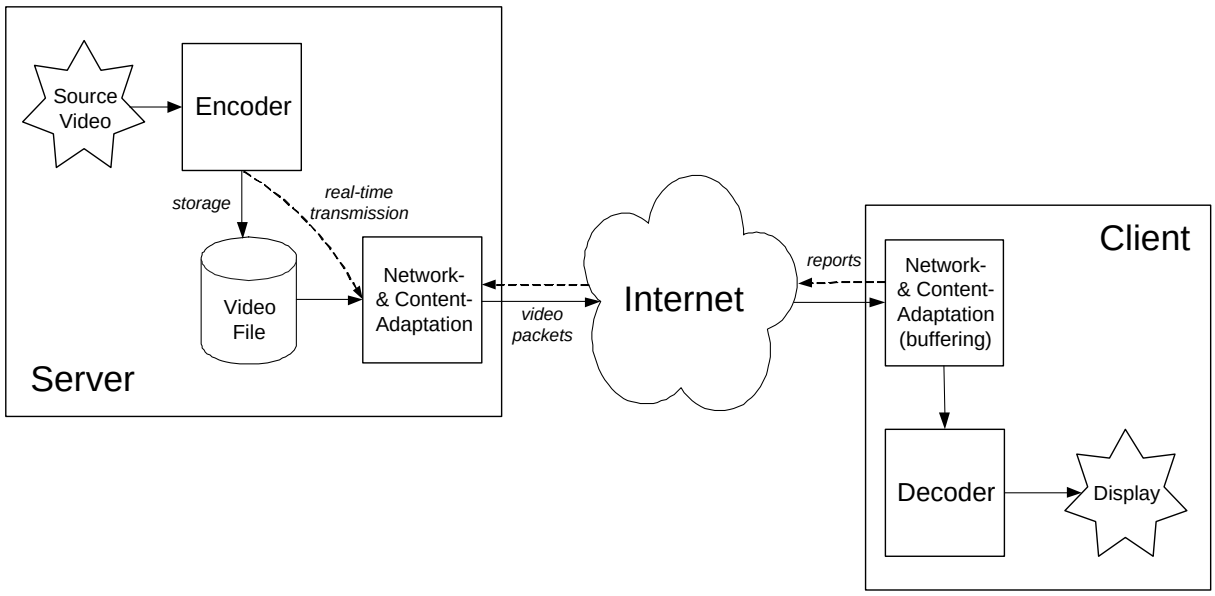


Figure 1.1: Streaming system

which encodes the video into two ore more complementary layers; the server can adapt to changing network conditions by adding/dropping layers.

This dissertation makes several major contributions:

– We compare adaptive control policies that add and drop encoding layers to control policies that switch among different versions of the same video. We show, in the context of reliable TCP–friendly transmission, that switching versions usually outperforms adding/dropping layers because of the bit–rate overhead associated with layering.

– We present a novel framework for low–complexity adaptive streaming of stored Fine–Grained Scalable (FGS) video over a reliable TCP–friendly connection. We formulate and solve an optimal streaming problem, which suggests a real–time heuristic policy. We show that our framework gives similar results with smooth or highly varying TCP–friendly connections. We present an implementation of our heuristic in a platform that streams MPEG–4 FGS videos.

– We analyze rate–distortion traces of MPEG–4 FGS videos; we find that the semantic content and that base layer coding have significant impact on the FGS enhancement layer properties. We formulate an optimization problem to compare rate–distortion optimized streaming at different aggregation levels; we find that scene–by–scene optimal adaptation can achieve good performance, for a lower computational complexity than image–by–image adaptation.

– We propose an end–to–end unified framework that combines scheduling, Forward Error Correction (FEC) and decoder error concealment. We use Markov Decision Processes (MDPs) over an infinite–horizon to find optimal transmission policies with low–complexity and for a wide variety of performance metrics. Using MPEG–4 FGS videos, we show that accounting for decoder error concealment can enhance the quality of the received video significantly, and that a static error protection strategy achieves near–optimal performance.

1.3 Outline of the Thesis

In Chapter 2 we give an overview of the Internet and digital video coding technologies. We focus on the characteristics of today's best–effort Internet and on video layered encoding techniques. We describe general application issues and transport issues for networked video applications. Leveraging the existing literature, we show that streaming applications should both adapt to changing network conditions and to the characteristics of the streamed video.

In Chapter 3 we compare two adaptive schemes for streaming stored video over a reliable TCP–friendly connection, namely, switching among multiple encoded versions of a video, and adding/dropping encoding layers. We develop streaming control policies for each scheme and evaluate their performance using simulations from Internet traces.

In Chapter 4 we focus on a new form of layered encoding, called Fine–Granularity Scalability (FGS). Streaming FGS–encoded video is more flexible than adding/dropping regular layers, and switching versions. We present a novel framework for streaming stored FGS video over a reliable TCP–friendly connection. Under the assumption of complete knowledge of bandwidth evolution, we derive an optimal policy for a criterion that involves both image quality and quality variability during playback. Based on this ideal optimal policy, we develop a real–time rate adaptation heuristic to stream FGS video over the Internet. We study its performance using real Internet traces, and with simulations over different TCP–friendly protocols. We also present its implementation in an end–to–end streaming application that uses MPEG–4 FGS videos.

In Chapter 5 we continue to explore adaptive techniques for streaming FGS–encoded video, by considering fine adaptation to the characteristics of the streamed video. In the context of rate–distortion optimized streaming, we analyze rate–distortion traces of MPEG–4 FGS encoded video. We define performance metrics that capture the quality of the received and decoded video both at the level of individual video frames (images) and at the level of aggregation of images: GoP (Group of Picture), scene, etc. Our analysis of the rate–distortion traces for a set of long videos from different genre provides a number of insights that are useful for the design of streaming mechanisms for FGS–encoded video. Using our traces, we investigate the rate–distortion optimized streaming at different video frame aggregation levels.

In Chapter 6 we extend our adaptive techniques for reliable transmission of layered video to the case when the enhancement layer is transmitted with partial protection against packet–loss. We consider streaming both regular and FGS layered–encoded video, and both streaming live and stored video. We propose an end–to–end unified framework that combines scheduling, FEC error protection and decoder error concealment. We formulate a problem for rate–distortion optimized streaming, which accounts for decoder error concealment. We use the theory of infinite–horizon, average–reward Markov Decision Processes (MDPs) with average–cost constraints to find optimal policies that maximize the quality of the video. We present simulations with MPEG–4 FGS video, for when the sender has perfect information about the state of the receiver and when it has imperfect information.

Finally, in Chapter 7, we summarize our contributions and give several areas of future work.

1.4 Published Work

Parts of the work presented in this dissertation have been published or are still in submission:

- P. de Cuetos, D. Saparilla, K. W. Ross, *Adaptive Streaming of Stored Video in a TCP–Friendly Context : Multiple Versions or Multiple Layers*, Packet Video Workshop (PV'01), Kyongju, Korea, April 30 – May 1, 2001.
- P. de Cuetos, K. W. Ross, *Adaptive Rate Control for Streaming Stored Fine–Grained Scalable Video*, Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'02), Miami, Florida, May 12–14, 2002.
- P. de Cuetos, P. Guillotel, K. W. Ross, D. Thoreau, *Implementation of Adaptive Streaming of Stored MPEG–4 FGS Video over TCP*, International Conference on Multimedia and Expo (ICME'02), Lausanne, Switzerland, August 26–29, 2002.
- P. de Cuetos, K. W. Ross, *Optimal Streaming of Layered Video: Joint Scheduling and Error Concealment*, to appear in ACM Conference on Multimedia, Berkeley, CA, November 2–8, 2003.
- P. de Cuetos, M. Reisslein, K. W. Ross, *Evaluating the Streaming of FGS–Encoded Video with Rate–Distortion Traces*, submitted, June 2003.
- P. de Cuetos, K. W. Ross, *Unified Framework for Optimal Video Streaming*, submitted, July 2003.

Chapter 2

Streaming Video over the Internet

In this chapter we give an overview of the main technologies that are involved in an Internet video streaming application, i.e., the Internet (Section 2.1) and digital video coding (Section 2.2). Based on these generalities, we focus on the particular issues of video streaming over the Internet (Section 2.3), and we detail related works on adaptive techniques for Internet video streaming (Section 2.4).

2.1 The Internet

One of the main technologies which are involved in an Internet video streaming application is the underlying network itself, i.e., the Internet. We start this section by reviewing the main characteristics of the current Internet that will influence the design of networked media applications. Then, we introduce the two available transport protocols which are used by Internet applications, i.e., TCP and UDP. Finally, we present some proposed changes in the current Internet infrastructure, which could improve the ability of the Internet to transport media in the near future.

2.1.1 The Best–Effort Internet

The Internet is the largest computer network with millions of users. It is based on the Internet Protocol (IP) which allows routing data packets between any pair of computers. On its path from source to destination, a data packet can be transported across many different sub–networks, on physical links with different capacities. Internet routers store incoming packets into drop–tail queues. Packets are forwarded to an output link after routing decision, which is based on the packet IP destination address. Most current Internet routers implement the FIFO policy, i.e., they discard all arriving packets when the incoming queue is full.

The current Internet is still *Best–Effort*. This means that it does not guarantee any Quality of Service (QoS) to applications. The QoS metrics that are essential to most Internet applications are the end–to–

end transmission delay, the packet loss rate and the available connection bandwidth. The best–effort Internet is characterized by both highly heterogeneous and varying network conditions.

Network Heterogeneity

Network heterogeneity comes from the diversity in topology of the Internet and the diversity in the hardware used throughout the network, such as the physical links [43]. In particular, the heterogeneity in available bandwidth for a given end–to–end connection can be related to what is commonly called the *"last mile problem"*, i.e., the bottleneck bandwidth[1] for a connection is very often located on the link between the end user and its connection to his provider (also called the access link). Today's commonly used Internet access links have different bandwidth capacities, which contributes to network heterogeneity.

Internet access networks can be grouped into three major categories [67]:

- **Residential access networks**: they connect a client to the Internet from his home. Today's most used access technologies are dial–up modems, ISDN (Integrated Services Digital Network), ADSL (Asymmetric Digital Subscriber Line), and HFC (Hybrid Fiber Coaxial cable). Dial–up modem speeds do not exceed 56 kbps, while ISDN telephone lines provide the user with an end–to–end digital transport of data at rates up to 274.1 Mbps, the basic service being 128 kbps. ADSL is an increasingly popular technology, which uses special modems over the existing twisted pair telephone lines. The available access rates can vary as a function of several parameters, such as the distance between the home modem and the central office modem, or the degree of the line electrical interference. For a high quality line, a downstream transmission rate of up to 8 Mbps is possible if the distance between the two modems is lower than 3 km. HFC is a concurrent technology to ADSL for broadband Internet access. Fiber optics are used to connect the cable head end to neighborhood level junctions, and regular coaxial cables connect the neighborhood junction to the user's homes. Transmission rates can go up to 10 Mbps but, as with Ethernet, HFC is a shared broadcast medium, so the bandwidth is shared among users connected to the same neighborhood junction.

- **Company access networks**: all terminals are interconnected to an edge router via a LAN (Local Area Network). The edge router is the access gate to the Internet for the company. The most used technology today is Ethernet with shared transmission rates of 10 Mbps to several Gbps.

- **Mobile access networks**: mobile terminals (e.g., cellular phones, laptops, or PDAs) access the Internet by using the radio spectrum to connect to a Base Station (BS). There are two types of mobile

[1] Note that the *bottleneck bandwidth* of a connection is the upper limit on how quickly the network can deliver data over this connection, while the *available bandwidth* denotes how quickly the connection can transmit data while still preserving network stability [95].

access networks. Wireless LAN is an increasingly popular technology, which is used for sharing Internet access within a short–range (tens of meters), typically within business offices, universities, hotels, coffee shops, etc. The IEEE 802.11 standard defines transmission rates of up to 11 Mbps (802.11b) and up to 54 Mbps (802.11a). Wide–area wireless access networks have a range similar to today's mobile phone service, i.e., the base station can be kilometers away from the clients. Packet transmission standards like GPRS (General Packet Radio Service) can reach transmission rates of 115 Kbps while the upcoming UMTS (Universal Mobile Telecommunications System) promises to provide access rates of up to 2 Mbps.

The diversity in Internet access technologies is only partially responsible for the heterogeneity in network conditions. Specifically, the heterogeneity in the average RTT (Round Trip Time) of a connection is also due to the physical location of the communicating client and server. For instance, the average RTT between two terminals connected to corporate LANs inside France is today typically at the order of tens of milliseconds, while transatlantic connections between France and the USA have an average RTT at the order of hundreds of milliseconds. Also, because of the presence of tail–drop queues inside routers and TCP's end–to–end congestion control algorithm (see Section 2.1.2), long–RTT connections tend to get a smaller share of bandwidth than short–RTT connections [98]. Finally, the average packet loss rate of a connection is usually between 1% and 5%. However, loss rates of more than 30% are also possible.

Varying Conditions

Besides having heterogeneous conditions, a given Internet connection also experiences short– and long–term varying conditions during its lifetime, such as varying loss rates and delays, or varying available bandwidth. These variations are mainly due to competing traffic inside network routers and to route changes. Route changes usually follow a router failure, or a routing decision after the increase of the traffic load inside a router.

Paxson [95] analyzed several TCP traces and showed that variations of the transmission delay (also denoted by delay *jitter*) occur mainly at short time scales of 0.1 to 1 second. High variations in transmission delay typically cause the reordering of packets, which has been shown to be very common in the best–effort Internet. Loguinov and Radha [77] report experiments of Internet transmissions between several U.S. cities, using dial–up modems. The average RTT was found to be around 750 ms, while some sample RTTs reached several seconds and the minimum RTT was around 150 ms.

Concerning variations in the average packet loss rate of a connection, Yajnick et al. [144] found that the packet loss correlation timescale is also 1 second or less. Packet loss episodes are often modeled as i.i.d. (independent and identically distributed), or as a 2–state Markov Chain (also called the *Gilbert model*). It has been shown that i.i.d. models give good approximations on time scales of seconds to minutes. But packet losses on time scales of less than a second are better approximated by the Gilbert

model [148, 149]. The Gilbert model takes into account the observation that packet losses usually occur in short–length bursts, for packets sent in a short time interval [19]. This can be partially explained by buffer overflows inside high–loaded routers.

As we explain later in Section 2.1.2, because of TCP's congestion control algorithm, the available bandwidth of a TCP connection can have important short–term variations. However, Zhang et al. [148] showed that the throughput of a long TCP connection does not widely fluctuate over a few minutes. Therefore, one can estimate throughput from observations of minutes in the past with reasonable accuracy (however, note that estimations from past observations of more than an hour can be misleading).

The heterogeneity in network conditions, as well as the variability in available bandwidth, delay and loss rate make the best–effort Internet both difficult to simulate [43] and difficult to predict [139]. This explains the difficulty in designing "QoS–sensitive" Internet applications, such as interactive video streaming or Internet telephony.

2.1.2 Transport Protocols: TCP vs. UDP

The Internet transport layer can use one of the following two protocols to provide an Internet service to applications. These are UDP (User Datagram Protocol) and TCP (Transmission Control Protocol).

UDP

UDP is a very simple transport protocol. It just provides multiplexing/demultiplexing service to the application layer, i.e., it allows delivery of data from the source application process to the destination application process [67]. Typical applications that run over UDP include streaming multimedia, Internet telephony and Domain Name Translation (DNS).

UDP is a connection–less protocol, which means that it does not require the setup of a virtual connection between the client and the server. It is datagram–oriented, i.e., it moves complete messages from the application to the network layer.

TCP

TCP is a reliable transport protocol. Unlike UDP, TCP retransmits the segments that have been lost by the underlying network. Typical applications that run over TCP include e–mail (SMTP), web (HTTP) and reliable file transfer (FTP).

TCP is a connection–oriented protocol: it requires establishing a connection between the client and the server before transmitting any application data (this is done through a 3–way handshake protocol). Also, TCP is byte–oriented, i.e., the sender writes bytes into a TCP connection and the receiver reads bytes out of the TCP connection. At the source host, TCP first buffers the incoming bytes from the

sending application. The buffered bytes are sent to the IP layer, when the size of the buffer has reached the Maximum TCP Segment Size (MSS), or after expiration of a Time Out (TO).

TCP implements two mechanisms to control the rate of the transmitted stream: *flow control* and *congestion control*. Flow control limits the sending rate in order to prevent overflow of the receiver's reception buffer; congestion control prevents network congestion by reducing the sending rate upon indication of network packet loss.

The congestion control algorithm which is implemented in the first version of TCP (TCP–Tahoe) is explained in [56]. The congestion window size $cwnd$ denotes the maximum number of unacknowledged segments that the sender can tolerate before transmitting new segments. During the initial phase called *slow–start*, its value is increased by one for each acknowledgment (ACK) received. This results in exponential growth of the congestion window. When $cwnd$ exceeds the threshold $ssthresh$, the server enters into the *congestion avoidance* phase, during which $cwnd$ is increased by one for each window of $cwnd$ segments acknowledged. Each TCP segment sent to the receiver triggers the start of a TO, which is canceled upon reception of the positive acknowledgment for this segment. The expiration of the TO for a given segment is considered as an indication that the segment is lost. In this case, the server re–enters the slow–start phase with $cwnd = 1$ and $ssthresh = ssthresh/2$. Because of the linear increase of the congestion window size during congestion avoidance and its sharp decrease upon the expiration of a TO, TCP congestion control algorithm is called an Additive–Increase Multiplicative–Decrease (AIMD) algorithm.

Other improved versions of TCP have been implemented since TCP–Tahoe, namely TCP–Reno and TCP–Vegas. TCP–Reno is now implemented in most operating systems. It includes a mechanism which triggers the retransmission of unacknowledged packets upon reception of 3 duplicate ACKs, without entering slow–start (*fast retransmit* and *fast recovery*).

TCP's congestion control algorithm has been designed to provide competing TCP connections with an equal share of a bottleneck bandwidth[2]. In [92], Padhye et al. have given an analytic characterization of the steady state TCP throughput. However, the AIMD nature of TCP also results in highly varying throughput at short–time scales, which is often considered as an impediment for multimedia streaming applications, as we discuss in Section 2.3.2. Still, TCP's congestion control mechanism seems essential to today's Internet scalability and stability [66].

2.1.3 Evolution of the Internet

When the Internet was designed and deployed, it was tailored to transport data files with no requirement on the transmission delay. Typical target applications were file transfer, e–mail and web. Now that

[2] in practice, this is only well verified for flows with the same propagation delay over an over–provisioned link [98].

the Internet is almost ubiquitous, it would be convenient to use it for real–time applications, such as telephony or video–conferencing. However, such applications are difficult to implement in the current best–effort Internet because, as we mentioned in Section 2.1.1, the network cannot guarantee any QoS, such as the maximum end–to–end transmission delay. Less "QoS–sensitive" Internet applications such as streaming stored–video or simple web browsing could also benefit from QoS guarantees, such as a minimum available bandwidth or a maximum loss rate.

Two main proposals have been made by the IETF group (Internet Engineering Task Force) to offer some quality of service guarantees to the Internet, namely *Integrated Service* (IntServ) and *Differentiated Service* (DiffServ). Both proposals require the modification of the current Internet architecture.

IntServ

The IntServ architecture [22] provides absolute QoS guarantees for each individual flow, which are typically guarantees of a minimum available bandwidth, a maximum tolerable end–to–end delay, or a maximum loss rate. IntServ is generally used in conjunction with RSVP (Resource ReSerVation Protocol), which provides signaling and admission control. Unlike in the current Internet infrastructure, IntServ requires maintaining per–flow states in routers.

DiffServ

While IntServ provides QoS guarantees on each individual flow, DiffServ works on traffic aggregates, i.e., a large set of flows with similar QoS requirements. The DiffServ architecture [18] distinguishes between two classes of routers: core routers and edge routers. At the edge routers, packets are classified (or marked) into different classes of service. The outgoing traffic is conditioned to match with a specified SLA (Service Level Agreement), which has been negotiated between the client and its ISP (Internet Service Provider). An SLA defines long–term expected traffic specifications in terms of various performance metrics such as throughput, drop probability or latency. Core routers achieve service differentiation by forwarding packets differently according to their class of service[3]. The IETF has standardized two router forwarding services, namely the Assured Forwarding (AF) [50] and the Expected Forwarding (EF) [57] services. The EF service gives absolute end–to–end guarantees of service to any class of traffic, in terms of bandwidth or latency. It is comparable to a virtual leased line. The AF service defines different levels of forwarding assurances for IP packets. The IETF has defined 4 different AF classes with 3 different dropping priorities for each class. Several studies have presented new packet marking mechanisms for providing applications with end–to–end QoS guarantees, such as throughput guarantees [28, 36, 112]. In [13], Ashmawi et al. present a video streaming application that uses the policing actions and rate

[3]In the simple case with only two different classes of service, packets corresponding to the highest class of service are marked as *in* packets (by opposition to *out* packets) by edge routers. In periods of congestion, unmarked or out packets are preferentially dropped by core routers inside the network.

guarantees of the EF service.

Both IntServ and DiffServ proposals have attracted many research efforts in the past few years. However, they are not deployed yet, so the current Internet is still best–effort. The main issue is the deployment scalability of both approaches in the current Internet architecture. With IntServ, end–to–end service guarantees cannot be supported unless all nodes along the path support IntServ; with DiffServ, end–to–end service guarantees can only be provided by the concatenation of local service guarantees, which requires SLA agreements at all customer/provider boundaries.

Other Changes

Besides the IntServ and DiffServ approaches, Internet applications could benefit from several other changes in the Internet architecture, including:

- **Active queue management**. This consists in using queue management algorithms other than FIFO inside routers. A popular queue management algorithm is RED (Random Early Detection) [42]. The main motivation is to control the average queueing delay inside routers, in order to prevent transient fluctuations in the queue size from causing unnecessary packet drops [39].

- **Explicit Congestion Notification (ECN)**. This allows routers to set the Congestion Experienced (CE) bit in the IP header as an indication of congestion, rather than dropping the packet. In this case, the TCP sender enters the congestion avoidance phase with no packet loss [39].

- **Multicast protocols**. Multicast communication consists in transporting data from one host to a group of hosts, by aggregating unicast connections. In such an approach, multicast routers need to replicate the datagrams that are sent to a given group, to each output link leading to hosts of the group. Despite potential gains in bandwidth, multicast routing raises concerns about scalability, because multicast routers need to maintain states for each multicast group [14].

- **IPv6**. Internet Protocol version 6 is the successor of the current protocol IPv4. It features several new functionalities, such as the increase in the number of possible Internet addresses, the support of multicast (avoiding the use of tunnels) and a simpler header than IPv4 [80]. However, IPv6 is unable to inter–operate with IPv4, which has contributed to delay its deployment.

The changes in the Internet infrastructure which have been presented in this section could help to make the current best–effort Internet more suitable to applications that have high quality of service requirements, such as real–time transmission of videos. In particular, these can alleviate network heterogeneity and varying network conditions, by limiting congestion or by providing statistical guarantees on the loss rate, the end–to–end transmission delay or the available bandwidth. However, the current

Internet is still best–effort, so networked applications have to cope with the lack of quality of service guarantees.

2.2 Digital Videos

Since the Internet has limited transmission capacities, videos need to be compressed before transmission. This is achieved by video coding. In Section 2.2.1, we recall some generalities about video coding, and we present some of the main standards, which are currently used in commercial products. We present the architecture of an MPEG–4 compliant system in Section 2.2.2. Finally, in Section 2.2.3 we focus on layered–encoding, which is an encoding technique that is particularly tailored to networked video applications.

2.2.1 Video Coding

The raw size of digital videos is usually very high. Video coding consists in exploiting the inherent redundancy of videos in order to cut down their representation size. Redundancies in the video signal can be spatial (within a same video frame) or temporal (within adjacent frames)[4]. As an example, the commonly used full–motion 300–frame sequence *Foreman*, encoded in MPEG–4 with high quality and CIF resolution (352x288 pixels), has an average bitrate of 1.23 Mbps, compared to 36.5 Mbps for the uncompressed video.

The most used standard codecs (coder/decoder) can be grouped into two subsets [47]:

- The first subset is composed of the standards from the ITU (International Telecommunication Union), mainly H.261 and H.263, which were standardized in 1990 and 1995, respectively. These codecs are oriented towards videoconferencing applications. H.261 yields bitrates between 64 kbps and nearly 2 Mbps. H.263 is an extension of H.261 for low bitrate video; it can produce small dimensional video pictures at 10 to 64 kbps, thus suitable for transmission over dial–up modems.

- The other subset is composed of the standards from the MPEG committee (MPEG stands for Motion Picture Expert Group). MPEG codecs are oriented to storage and broadcast applications. MPEG–1 was first standardized in 1992, followed by MPEG–2 in 1995 and MPEG–4 in 1999. MPEG–1 focuses on digital storage of VCR image quality videos, at target bitrates between 1 and 1.5 Mbps. It is suitable for storage on CD–ROMs, which have output rates of at least 1.2 Mbps. MPEG–2 was designed to match with a wider variety of applications, in particular the broadcast of high interlaced video or HDTV (High Definition Television), at high bitrates from 4 to 9 Mbps. Finally, the recent MPEG–4 standard introduces object–based coding and it can be used for a

[4]Throughout this dissertation, we use the terms *image* and *frame* interchangeably.

Standard	Target bitrate	Target applications	Year of standardization
H.261	$p * 64$ kbps, with $1 \leq p \leq 30$	Videoconferencing	1990
MPEG–1	$[1, 1.5]$ Mbps	Storage	1992
MPEG–2	$[4, 9]$ Mbps	Wide variety	1995
H.263	$[10, 64]$ kbps	Videoconferencing	1995
MPEG–4	[5 kbps, 10 Mbps]	Wide variety	1999

Table 2.1: Characteristics of common codec standards

broader range of target bitrates, from 5 kbps to 10 Mbps. It is suitable to almost all applications requirements, such as broadcast, content–based storage and retrieval, digital television set–top boxes, mobile multimedia and streaming over the Internet [9].

Table 2.1 summarizes the characteristics of the previously mentioned video codec standards. In this dissertation, we mainly focus on MPEG standards, and especially on MPEG–4. In MPEG encoded videos, images are grouped into GoPs (Group of Pictures). Inside a given GoP, frames can be of 3 types:

- **I–frames** (Intra–coded frames): they are independently encoded, i.e., without any temporal prediction from other frames.
- **P–frames** (Predicted frames): they are predicted from the previous I–frame of the current GoP.
- **B–frames** (Bi–directional predicted frames): they are predicted both from the previous and the next I or P–frame of the current GoP.

In digital video, each pixel is represented by one luminance value and two chrominance values. In conventional MPEG coding, the pixels are grouped into blocks of typically 8x8 pixels. The 64 luminance values in the block are transformed using the Discrete Cosine Transform (DCT) to produce a block of 8x8 DCT coefficients. The DCT coefficients are zig–zag scanned and then compressed using run–level coding. The run–level symbols are then variable–length coded (VLC). (The chrominance values are processed in similar fashion, but are typically sub–sampled prior to quantization and transformation.)

Videos can be encoded in VBR (Variable Bit–Rate) or CBR (Constant Bit–Rate). With VBR–encoding, the quantizers used for each type of image (I, P, B) are constant throughout the video. The goal of VBR–encoding is to achieve a roughly constant quality for all images of the video. The bitrate of the compressed bitstream varies as a function of the visual complexity of the original images. In contrast, CBR–encoded videos must respect a target average bitrate. This is achieved by a rate–control algorithm which determines the appropriate quantizer step to use for each image. Limiting the output bitrate comes with some degradations in quality compared to VBR–encoding [91].

Finally, note that videos have very strict real–time constraints during playback: every image has to be decoded and presented to the user at fixed time intervals. This interval corresponds to the *frame rate* of the video. The time at which a video packet should be decoded is called its *decoding deadline*.

2.2.2 MPEG–4

In this dissertation, we present experiments with MPEG–4 encoded videos. The MPEG–4 formal designation is ISO/IEC 14496 [6]. As we mentioned in the previous section, MPEG–4 is a recent codec that has been designed for a broad range of applications, such as video streaming. One of the main objectives of the standard is also the flexible manipulation of audio–visual objects. MPEG–4 introduces the concept of an audio–visual scene, which is composed of one or many audio–visual objects [64]. Audio–visual objects can be still images, video or audio objects.

The architecture of an MPEG–4 terminal is depicted in Figure 2.1. In the compression layer, the composition of the media objects in the scene is defined by a specific language for scene description: BIFS (BInary Format for Scene description). Each object is described by an *Object Descriptor* (OD), which contains useful information about the object, such as the information required to decode the object, its QoS requirements, or textual descriptors about the content (keywords) [51]. One object is composed of one or more *Elementary Streams* (ES). Object descriptors and scene description information are also carried in elementary streams.

An *Access Unit* (AU) is defined as the smallest element that can be attributed with an individual timestamp. An entire video frame is a typical AU. The SyncLayer packetizes the AUs with additional information such as timing. Timing is expressed in terms of decoding and composition timestamps [15].

The algorithms that are used to code video objects are defined in ISO/IEC 14496–2 [7]. A *Video Object* (VO) can be a rectangular frame or an arbitrarily shaped object, corresponding to a distinct object or the background of the scene. Each time sample of a video object is called a *Video Object Plane* (VOP). The MPEG–4 standard does not specify how to segment video objects from a scene. Therefore, as of today, most MPEG–4 encoded videos just comprise one video object, which is the rectangular–shaped video itself. In this case, a VOP just denotes a video frame or image.

The architecture of MPEG–4 systems has been designed to be independent of the transport. The Delivery Layer makes possible to access MPEG–4 content over a wide range of delivery technologies [44]. Delivery technologies are grouped into three main categories: interactive network technologies (Internet, ATM), broadcast technologies (Cable, Satellite) and disk technologies (CD, DVD). The FlexMux is an optional tool; it can be used to group elementary streams with similar QoS requirements, thus reducing the total number of network connections required. Finally, the DMIF[5] Application Interface (DAI) allows to isolate the design of MPEG–4 applications from the various delivery layers. The implementation

[5]DMIF stands for Delivery Media Integration Framework.

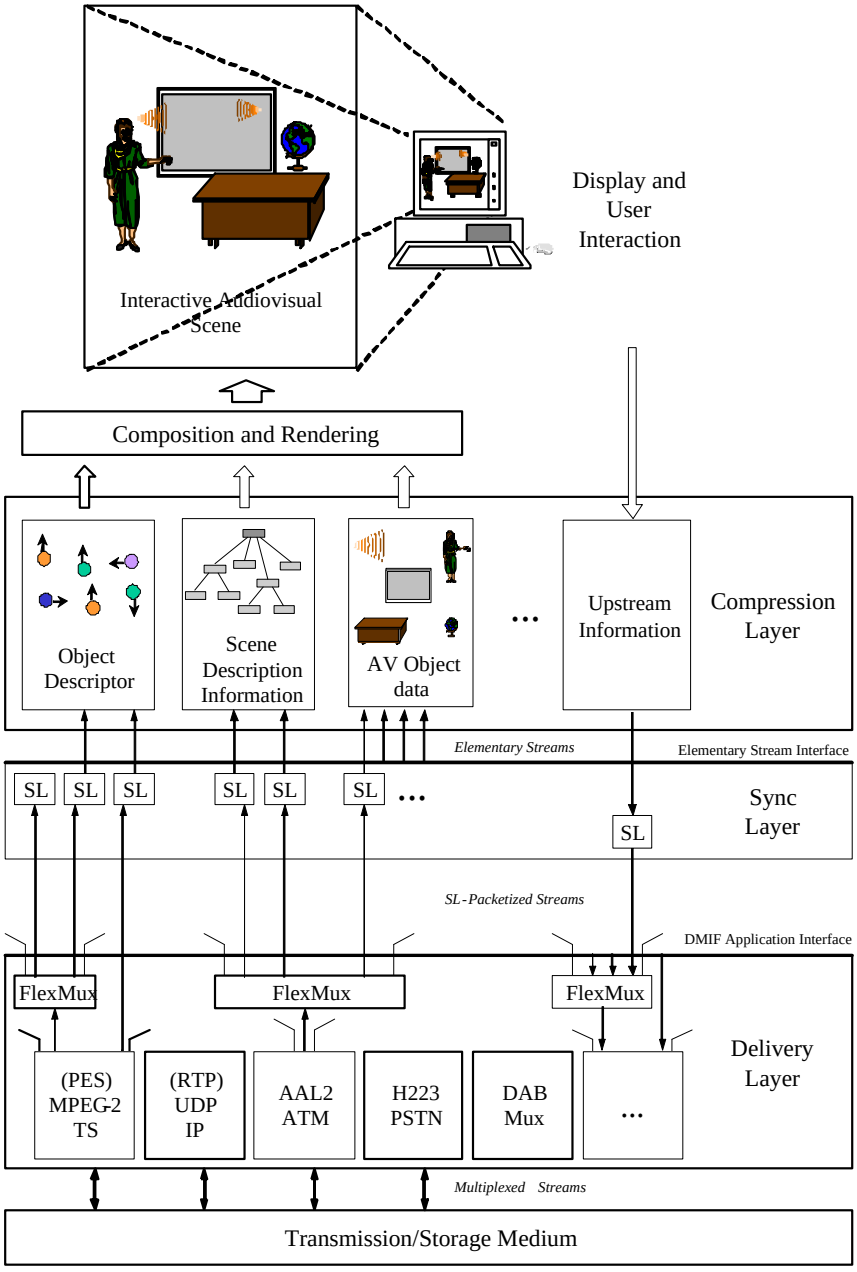


Figure 2.1: MPEG–4 system (© MPEG)

of a streaming MPEG–4 system supporting DMIF is presented in [59]; [17] discusses architecture issues for the delivery of MPEG–4 video over IP.

2.2.3 Layered–Encoding

Hierarchical encoding — also called *scalable* or *layered* encoding — is an encoding technique that is particularly well suited to networked video applications. Layered–encoding appears first in the MPEG–2 standard, and later in H.263+ (enhanced version of H.263) [29] and in MPEG–4. It was proposed to increase the robustness of video codecs against network packet loss [47]. The main concept of scalable encoding is to encode the video into several complementary layers: the *Base Layer* (BL), and one or several *Enhancement Layers* (ELs). The base layer is a low quality version of the video. It has to be decoded in order to show minimum acceptable video quality. The rendering quality of the video is then progressively enhanced by decoding each enhancement layer successively. All enhancement layers are hierarchically ordered: in order to decode the enhancement layer of order n, the decoder needs all lower order layers, i.e., the base layer and all enhancement layers of order 1 to $n-1$.

In this dissertation, we focus on using layered–encoding for video streaming to gracefully adapt the quality of the video to heterogeneous and variable network conditions. This property of scalable encoding is particularly useful with the increased mobility of users. However, scalable videos can be used for many other applications than streaming, such as universal media access (videos are layered–encoded only once and can be played on a large range of devices from PDAs to HDTV screens), or differentiated content distribution (the base layer is distributed free of charge, while the enhancement layers are encrypted and distributed for a fee).

There are several types of video layered–encoding which have been defined in most recent codecs. We detail these techniques below.

Data Partitioning

Data Partitioning (DP) is a simple video scalability scheme. All layers can be obtained directly from the non–layered compressed video bitstream. Each layer contains a different set of DCT coefficients for all image blocks. The base layer contains the first DCT coefficients, i.e., the lowest frequency coefficients. The enhancement layers contain the remaining DCT coefficients, i.e., the ones that correspond to higher frequencies. The number of DCT coefficients to allocate to each layer is given by what is called the Priority Break Point (PBP) values.

Temporal Scalability

In temporal scalability, the base layer is encoded at a reduced frame rate. The enhancement layers are composed of additional frames that increase the displayed frame rate of the video. For better coding

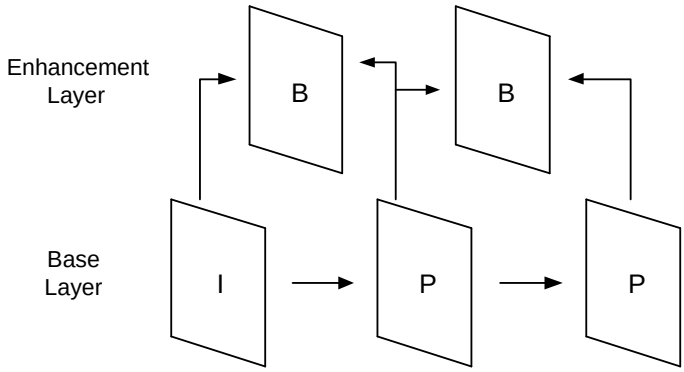


Figure 2.2: Example of temporal scalability

efficiency, the enhancement layer frames can be temporally predicted from the surrounding base layer frames. Temporal scalability can simply be implemented from a regular non–layered video containing all frame types I, P and B. Figure 2.2 shows the example of a video encoded into two layers, in which I– and P–frames form the base layer, while B–frames are allocated to the enhancement layer.

Spatial Scalability

With spatial scalability, the base layer is of a smaller spatial resolution than the original video. The enhancement layers contain information for higher spatial resolutions. We show in Figure 2.3 an example of a spatial scalable video encoded into two layers. When the decoder decodes only the base layer for a frame, it up–samples the frame to show it at full size but reduced spatial resolution. When the decoder decodes both the base layer and the enhancement layer it can show the frame at full size and full spatial resolution. For more coding efficiency, the enhancement layer encoding algorithm can use spatial prediction from the corresponding base layer frame and/or temporal prediction from the previous enhancement layer frames, as shown in Figure 2.3.

SNR Scalability

Signal–to–Noise Ratio (SNR) scalability consists in having layers with the same spatio–temporal resolution, but of different encoding qualities. As shown in Figure 2.4 for two layers, the base layer is obtained by encoding the video regularly with a coarse quantizer Q; the enhancement layer is obtained by encoding the error between the original video and the base layer decoded video, with a smaller quantizer Q'.

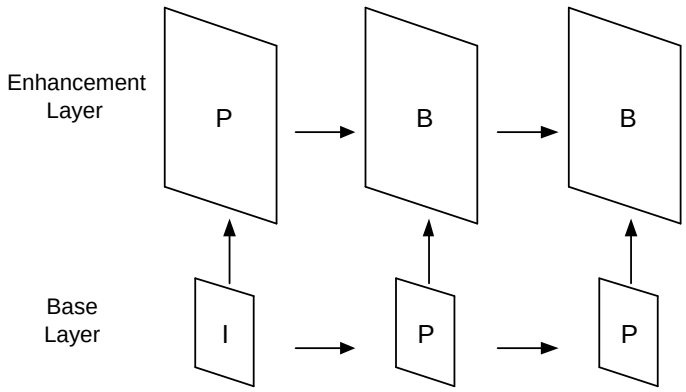


Figure 2.3: Example of spatial scalability

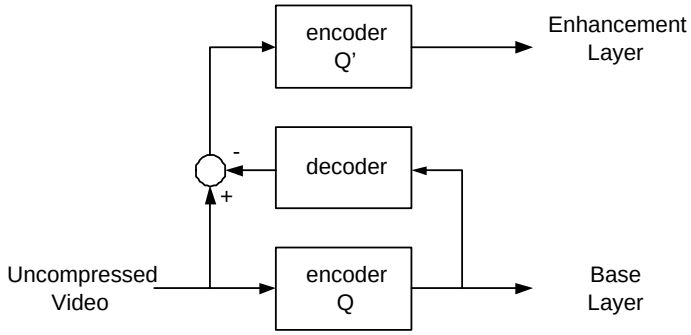


Figure 2.4: Implementation of SNR–scalability

Comparison

All types of scalability have different implementation complexities. Data partitioning is very easy to implement, because it only requires multiplexing/demultiplexing a non–layered compressed video. Also, as we mentioned earlier, temporal scalability may be simply obtained from a non–layered compressed video, by grouping the different types of pictures into different layers. However, SNR and spatial scalabilities usually require as many regular non–layered codecs as layers.

For a given target video quality, layered encoding usually comes with a bitrate penalty, compared to non–layered encoding. For data partitioning and temporal scalability, the overhead is only due to the replication of header information in all layers, such as frame numbers. This usually results in a negligible bitrate penalty. However, SNR and spatial scalabilities also replicate content information in all layers, which yields significantly larger bitrate overheads [63, 137].

The performance of all types of scalability for transmission over lossy channels has been evaluated in [12, 63]. It has been shown that, in general, layered encoding gives better resilience to transmission errors than non–layered encoding, in terms of the achieved rendering quality (i.e., better graceful degradation in quality in presence of transmission errors). Aravind et al. [12] compare the transmission, over ATM, of DP, SNR and spatial scalable videos. When the base layer is transmitted with full reliability, it is shown that spatial scalability provides the best performance, at the cost of a high implementation complexity. Kimura et al. [63] compare the transmission of DP, SNR and temporal scalable videos over a DiffServ–enabled Internet. The different layers are mapped into different priority levels. Temporal scalability is shown to perform poorly compared to data partitioning. Also, because of the large bitrate overhead associated with SNR scalability (in the range of 5% to 20%), data partitioning provides slightly better quality than SNR scalability.

Content Scalability

Content–based scalability comes directly from the possibility, given by the MPEG–4 standard, to code and decode different audio–visual objects independently [64]. The fundamental objects of the video can be grouped into the base layer, and the objects that are not crucial to the understanding of the video can be mapped into several enhancement layers[6]. As an example, during a videoconference, the head of the speaker can be considered as the base layer, while the background of the setting is considered as an enhancement layer. Note that content scalability requires the objects that are mapped to different layers to be encoded separately. This can be achieved either by capturing the original objects separately before

[6]Note that, in the case when the objects that compose the different enhancement layers can be decoded independently from lower layers, content scalability cannot be considered as a regular type of layered encoding. This is the case in [146] which considers the allocation of bandwidth to the different media objects composing the video according to their relative importance in the final rendered video quality.

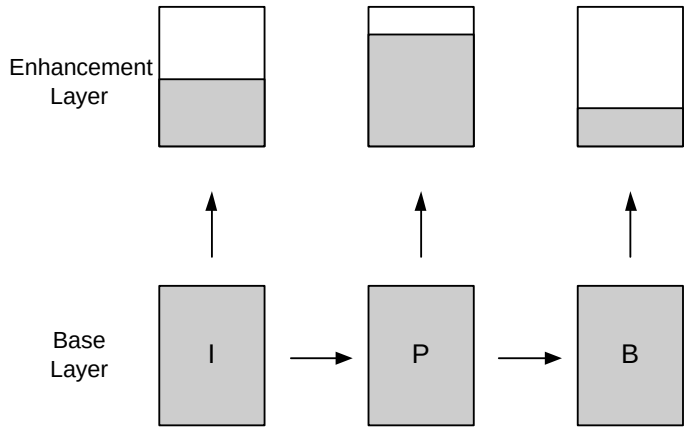


Figure 2.5: Example of truncating the FGS enhancement layer before transmission

encoding (for instance, by using a blue screen), or by segmenting a scene composed of several objects.

2.2.4 Fine Granularity Scalability

Fine Granularity Scalability (FGS) is a new type of layered encoding, which has been introduced in the MPEG–4 standard specifically for the transmission of video over the Internet [8]. The particularity of FGS encoding over the other types of scalability, is that the enhancement layer bitstream can be truncated anywhere during transmission, and the remaining part can still be decoded. Figure 2.5 shows an example of truncating the FGS enhancement layer before transmission over a network. For each frame, the shaded area in the enhancement layer represents the part of the FGS enhancement layer which is actually sent by the server to the client. Truncating the FGS enhancement layer for each frame before transmission allows the server to adapt its transmission rate to the changing available bandwidth of the connection. At the client side, the decoder can use the truncated enhancement layer to enhance the quality of the base layer stream.

MPEG–4 SNR FGS

In this dissertation, we focus on the MPEG–4 Signal–to–Noise Ratio (SNR) Fine Granularity Scalability [71, 72]. In SNR FGS, the FGS enhancement layer contains an encoding of the quantization error between the original video and the corresponding base layer decoded video. Figure 2.6 illustrates the architecture of a typical MPEG–4 SNR FGS decoder. According to the MPEG–4 standard, and as illustrated in this figure, only the base layer frames are stored in frame memory and used for motion compensation (predictive encoding). There is no motion compensation within the FGS enhancement layer. This makes the enhancement layer highly resilient to transmission errors, and subsequently well

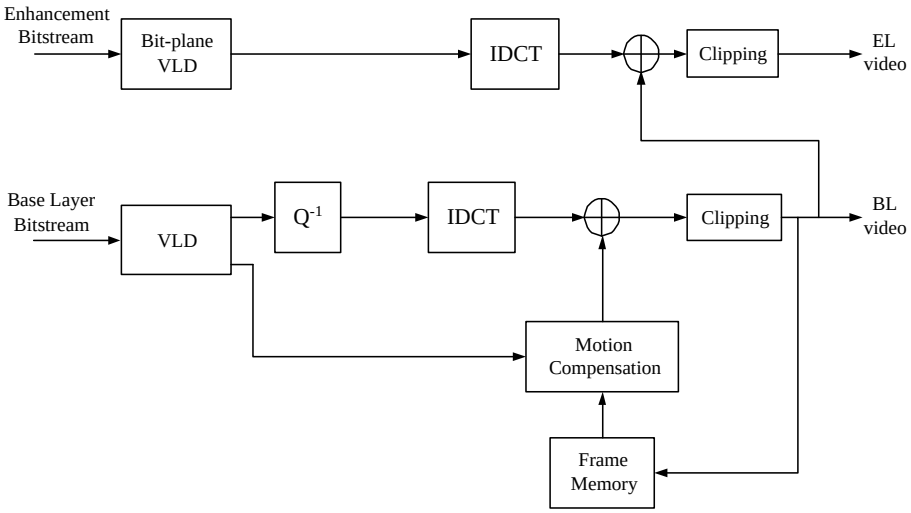


Figure 2.6: MPEG–4 SNR FGS decoder structure

suited to the transmission over error–prone networks such as the best–effort Internet. A typical scenario for transmitting MPEG–4 FGS encoded videos over the Internet has been proposed by the MPEG–4 committee in [10]. In this scenario the base layer is transmitted with high reliability (which is achieved through appropriate resource allocation and/or channel error correction) and the FGS enhancement layer is transmitted with low reliability (i.e., in a best effort manner and with low error control). It has been shown in [71] that FGS–encoding is more efficient than multilayer SNR–scalability.

The main difference between conventional MPEG encoding and FGS encoding is that the DCT coefficients of the enhancement layer are not run–level encoded in the FGS encoding, but instead bitplane encoded, which we now illustrate with an example. Consider the 8x8 block of enhancement layer DCT coefficients in the left part of Figure 2.7. The coefficients are scanned in zig–zag order to give the sequence of 64 integers starting with 7, 0, 5, ... Each integer is then represented in binary format (e.g., 7 is represented by 111, 3 is represented by 011). The representation for each integer is written in a vertical column as illustrated in the middle of Figure 2.7 to form an array that is 64 columns wide and 3 rows deep, as the largest integer in this example has a 3 bit binary representation (in practice 8 bit representations are typically used). The bitplanes are obtained by scanning the rows of the array horizontally. Scanning the row containing the most significant bit (the top row in the illustration) gives the Most Significant Bitplane (MSB). Scanning the row containing the least significant bit (the bottom row in the illustration) gives the Least Significant Bitplane (referred to as MSB-2 in this example, or more generally, LSB)[7]. Next, each bitplane is encoded into $(RUN,\ EOP)$ symbols. RUN gives the number

[7]Note that, because they have higher entropy, less significant bitplanes are usually represented with more bits than higher significant bitplanes [128].

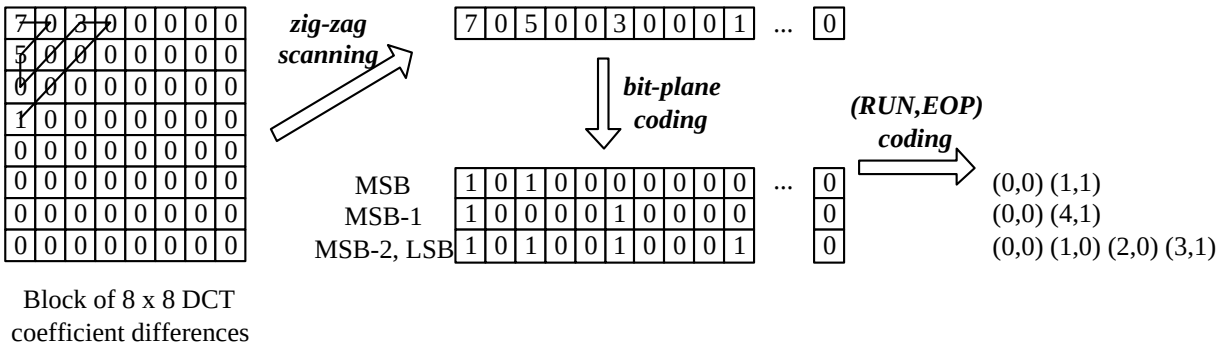


Figure 2.7: Example of bitplane coding

of consecutive "0"s before a "1". EOP is set to 0 if there are some "1"s left in the bitplane; if there is no "1" left in the bitplane then EOP is set to 1, as illustrated in Figure 2.7. The $(RUN,\ EOP)$ symbols are finally variable–length coded.

FGS Advanced Features

The MPEG–4 standard has specified several improvements for the basic FGS coding technique. These include frequency weighting and selective enhancement. *Frequency weighting* consists in using different weighting coefficients for the different DCT components. Bits pertaining to the most visually important frequency components are put in the bitstream ahead of the bits from the least important components. *Selective enhancement* consists in using different weighting for the different spatial locations of a frame. Bitplanes of some parts of a frame are put into the enhancement layer bitstream ahead of the bitplanes of other parts of the frame.

MPEG–4 also defines the *FGS Temporal Scalability* (FGST). This consists in adding a second FGS enhancement layer which increases the frame rate of the video. Each FGST frame can be coded using bi–directional prediction from the base layer. Like in regular temporal scalability, the temporal enhancement layer frames can be dropped during transmission. However, with FGS temporal scalability, parts of the FGST–EL can be used to increase the temporal quality of the video.

Finally, Progressive FGS (PFGS) is another improvement of the basic FGS technique, which has been proposed recently in [143, 146]. Unlike the MPEG–4 FGS, PFGS features partial motion compensation among the FGS bitplanes, while still achieving the fine granularity property. This usually improves the coding efficiency of regular FGS, but at the cost of a decrease in error resilience [142].

2.3 Transmission of Videos over the Internet

In the previous sections we have presented the two main technologies that are involved in an application that transmits videos over the Internet: the underlying best–effort transmission network and video coding. In this section we now focus on the networked video application itself. We describe the main issues and design choices of an Internet video application.

2.3.1 General Application Issues

Streaming vs. Downloading Stored Video

Transmission of stored video is also called Video on Demand (VoD). In such an application, the complete video has been encoded off–line and is stored as a data file, located in a server or proxy. The video is then delivered to the user upon request. There are two different ways of consuming stored video content across the Internet: downloading and streaming.

Downloading a video is similar to downloading a data file: the user can start watching the video only after the complete file has been received. With streaming, the user can start watching the video shortly after the transmission of the video file has begun; the user is watching the parts of the video that have been received, while the video server is transmitting future portions of the video.

When the user downloads a video, he has to wait for a large start–up delay, which corresponds to the transmission time of the video file. This time depends on the size of the video file and on the current characteristics of the Internet connection. The size of the video file is in turn a function of the source video characteristics (duration, image complexity), as well as the particular video encoder used and the target quality (image resolution). As an example, the size of a broadcast quality 30–mn video encoded with the MPEG–4 DivX codec [1], which is one of today's most advanced codecs, is around 100 MBytes. Over high speed ADSL connections with maximum downstream rates of 1 Mbps, this makes a minimum start–up delay of around 14–mns, i.e., almost half the total duration of the video. This delay is clearly restrictive compared to today's basic TV service. Also, this makes switching between different video programs tedious: each time the user wants to watch a new program, he has to wait for the transmission delay of the whole program.

The main goal of streaming is to reduce this start–up delay by allowing the user to start watching the video before complete reception of the video file. However, because, as we mentioned in Section 2.1.1, the current Internet does not provide any QoS to applications, there is no guarantee that the user will be able to watch the full quality video without interruption. Therefore, video streaming applications require specific network–adaptive techniques in order to maximize the rendered video quality.

Streaming Stored vs. Live Video

Unlike stored video applications that fully encode the video before its transmission, the transmission of live video requires the video frames to be produced, encoded and transmitted almost at the same time. Therefore, the transmission of live videos is only possible through streaming. Live videos can be truly interactive, such as in video conferencing applications, or non–interactive, such as in live broadcast applications (sports or other TV shows). Live streaming applications usually have more strict delay requirements than stored video streaming applications. When the image is produced at the remote site, it should be available at the client within a short delay. For instance, video conferencing applications usually require that the total transmission time of video and audio signals be less that 150 ms, in order to provide a seamless telecommunication service, while delays higher than 400 ms usually give poor quality [67]. For non–interactive live events, delays can be a little higher, depending on the content.

Streaming Layered–Encoded Videos

Network heterogeneity and varying conditions are usually addressed by using appropriate video encoding techniques. There are four popular techniques for adapting the streaming to long–term varying network conditions: on–the–fly encoding, switching among multiple encoded versions of the video, adding/dropping layers and adding/dropping descriptions.

On–the–fly encoding consists in encoding (or transcoding) the video at the same time that it is transmitted to the client. This is the default technique for live video applications, but it can be used also for stored video and non–interactive live video applications. This technique allows to fine–tune the encoding parameters, in order to match the video output rate to the current connection available bandwidth. In [21] Bolot and Turletti present a rate adaptation scheme for streaming real–time videos, that controls the output rate of the encoder by adjusting encoding parameters such as the frame grabbing rate, the quantizer or the movement detection threshold. However, encoding is CPU intensive and thus generally regarded as unsuitable for servers that stream a large number of videos simultaneously.

Switching versions is widely used in the industry. This technique consists in encoding the video at different output bitrates, corresponding to different quality levels. Each version is stored and available at the server. The server switches among the versions to adapt the transmission rate (and hence the quality level) to the connection available bandwidth.

An alternative technique to switching versions, is using a scalable, or layered–encoded, video (layered encoding has been introduced in Section 2.2.3). Video layers can be added or dropped by the server as a function of changing network conditions. Algorithms for optimal adaptation of layered–encoded videos to varying network conditions are presented in Section 2.4. In particular, the use of MPEG–4 FGS videos for Internet streaming has received much interest lately [99, 100, 128, 150].

Finally, another technique is using Multiple Description Coding (MDC). This consists in coding the

video into several descriptions, each of equal importance. This is different from scalable encoding which produces hierarchical layers (i.e., higher layers need lower layers to be decoded). Using MDCs is more flexible than using scalable video; however, this flexibility comes with a bitrate penalty. Using MDC for streaming video has been presented in [103, 123]; also, we refer the interested reader to [70, 104] for studies that compare streaming media with MDC and layered coding.

In Chapter 3 we investigate the differences between adaptive streaming of layered–encoded video and different versions of the same video, while in Chapters 4, 5 and 6 we focus on mechanisms that use layered–encoded videos.

2.3.2 Transport Issues

Multicast vs. Unicast Streaming

Video streaming over multicast has received much attention (see [73, 78] for comprehensive surveys). In particular, scalable encoding has been found to be well suited to the heterogeneity of receivers involved in a multicast video streaming application. McCanne et al. [81,82] derived a rate–adaptation protocol called RLM (Receiver–driven Layered Multicast) which is combined with video layering. In this approach, video layers are transported to different multicast groups; clients join and leave multicast groups according to their network capacity. Other works on audio and video multicast include [25, 61, 122, 125, 131].

Multicast transport is well adapted to streaming popular videos, when many users are likely to watch the same video at the same time (especially broadcast videos such as regular TV programs). The aggregation of many unicast connections makes more efficient use of network bandwidth. However, as we have mentioned in Section 2.1.3, this requires considerable network–layer support. Unicast transport is more appropriate for the transmission of non–popular videos, or videos requiring a high interactivity (such as a small start–up delay). In this dissertation we focus exclusively on unicast video streaming.

TCP vs. UDP

While transmission over multicast requires the use of UDP as a transport protocol, unicast networked video applications can use either TCP or UDP transport. However, TCP is usually considered as inappropriate for video streaming. This is mainly due to the following two reasons. First, TCP's full reliability is not necessary for video streaming applications. Indeed, videos are error resilient: because of the inherent redundancy of video, streaming applications over error–prone networks can usually tolerate a certain amount of packet loss without much degradation in perceived rendering quality. The full reliability of TCP also comes with a higher average transmission delay caused by the retransmission of lost packets, which may be an impediment for real–time video streaming. Secondly, TCP's AIMD congestion control algorithm is often considered inappropriate for video streaming applications because

the resulting throughput is highly varying at short–time scales. Throughput variations can result in fluctuating rendering quality of the video frames, which is not desirable.

Because of these reasons, many researchers advocate streaming media over UDP rather than over TCP [106, 121]. Since UDP only provides a multiplexing service to the application, this requires the implementation of mechanisms for smooth–rate congestion control and partial error correction in the application layer.

And yet, TCP's additional delays for retransmissions are usually bearable for streaming stored or non–interactive live videos. These delays are at the order of hundreds of milliseconds, and can therefore be accommodated, in general, by small client buffering. Additionally, as we show in Chapter 4, larger client buffering can smooth TCP's short–term throughput variations. Finally, videos that are streamed over the Internet are usually highly compressed, so they require transmission with a high amount of error correction, close to full–reliability. Therefore, as do Krasic et al. [66], we strongly believe that streaming over TCP is still an alternative to UDP, for stored video and non–interactive live video. TCP has many practical advantages over UDP: TCP already implements congestion control and error correction, it is widely available and has been proven to be stable. Another practical reason to use TCP instead of UDP is that firewalls in corporate LANs usually block incoming UDP streams. As a matter of fact, a wide–scale experimental study with RealVideos [3] has shown that almost half systems were still streaming video over TCP [136]. An alternative approach to using TCP as it is now, is to modify TCP's congestion control mechanism such as in [53], so that it slows down its transmission rate without loosing packets by using ECN (see Section 2.1.3).

TCP–Friendly Congestion Control

Because UDP does not have any congestion control mechanism, video streaming applications over UDP should implement a mechanism to react to network congestion, in order to be fair to competing TCP traffic [40]. This TCP fairness issue has lead to the notion of TCP–friendly streams.

TCP–friendliness is defined in [40] as the following: "A flow is said *TCP–friendly* if its arrival rate does not exceed the arrival rate of a conformant TCP connection in the same network circumstances". Several TCP–friendly rate adjustment protocols have been recently developed by researchers [93, 118, 125,130] (see [138] for a more extensive list). Among today's most popular protocols, we find RAP [107], TFRC [41] or SQRT [16]. All protocols have different limitations, and achieve TCP–friendliness only under specific types of scenarios. For example, some of the proposed protocols are not TCP–friendly at loss rates higher than 5% [107, 118], others are specific to multicast applications [125, 130].

As does TCP, TCP–friendly algorithms react to indications of network congestion by reducing the transmission rate of the application. The transmission rate of TCP–friendly algorithms is typically smoother than that of TCP [145]. Nevertheless, because network congestion occurs at multiple time

scales [95], the bandwidth available to TCP–friendly streams still fluctuates over several time scales [41, 145].

Existing video streaming systems with TCP–friendly congestion control over UDP can be grouped into two categories: those that implement their own TCP–friendly congestion control mechanism, such as in [69, 106, 146], and those that do not rely on a specific TCP–friendly algorithm, such as in [27, 88, 113, 114]. In this dissertation, we consider TCP–friendly transmission, either over TCP or over UDP, and our frameworks do not rely on a particular TCP–friendly mechanism.

Error Correction: Selective Retransmissions vs. FEC

In addition to TCP–friendly congestion control, video streaming applications over UDP also need to implement partial error correction. Because streamed videos can be highly compressed, the loss of only one video packet can degrade the quality of a given image significantly. Also, most encoders do predictive encoding, i.e., they encode an image by prediction from surrounding images (e.g., I, P, B frames in MPEG codecs). In this case, the loss of a packet pertaining to one single image can propagate to all the images that are predictively encoded from that image. The most popular error correction techniques for video communication are selective retransmissions and FEC (Forward Error Correction). (An extensive study of all error correction techniques for video communication can be found in [137].)

FEC consists in adding redundancy to source video packets. The most used FEC codes are Reed–Solomon (RS) codes. RS(n, k) codes consist in adding $n-k$ redundant packets to k source packets before transmission. The reception of any k packets from the n transmitted packets allows the receiver to recover all k original source packets. FEC codes are generally used for interactive real–time communications, such as Internet telephony (see [96] for a survey on loss recovery techniques for streaming audio). They provide error correction with less delay than retransmissions, but at the cost of an increase in the required transmission rate. They are also typically used in situations where a feedback channel cannot be used, such as in multicast applications [89]. The analytical performance of FEC for multimedia streaming applications has been studied in [45].

Selective retransmissions of continuous media consists in retransmitting packets that are likely to be received before their decoding deadline. Selective retransmissions are typically used for streaming stored video applications, that can accommodate a large playback delay [94].

Because, in the best–effort Internet, the packet loss rate of a connection can vary significantly during its lifetime, error correction schemes should be adaptive to varying network conditions. In the next section we present such network–adaptive mechanisms for error correction, as well as some other general adaptive techniques for video streaming.

2.4 Adaptive Techniques for Internet Video Streaming

In this section we describe existing systems and previous research works for adaptive video streaming over the Internet. Section 2.4.1 focus on general techniques that adapt the streaming to changing network conditions. Because of the specificities of video, these network–adaptive techniques are usually complemented with techniques that adapt the streaming to the characteristics of the streamed video. We present such content–adaptive techniques in Section 2.4.2.

2.4.1 Network–Adaptive Video Streaming

Because of the varying and heterogeneous characteristics of the best–effort Internet, systems for video streaming should implement mechanisms that adapt the transmission to the current state of the network.

Playback Buffering

Today's most popular media streaming applications, i.e., Real Media [3], Windows Media [4] and Quicktime [2], all require a small *start–up delay* (a few hundreds of milliseconds) before the video can be rendered to the user. During this time, the server sends the initial portion of the video into a dedicated client buffer, which we call *playback buffer* — this is also called *playout buffer*. After the rendering of the video has started, the server continues sending new data to the end of the client playback buffer, while the decoder consumes the data available at the beginning of the buffer. We denote by *playback delay*, the delay between the moment an image is sent by the server and the moment it should be displayed to the user at the client (this depends on the image decoding deadline). In this dissertation, we make a distinction between the initial playback delay and the start–up delay: the start–up delay is the waiting time perceived by the user between the time he requests the video and the time the first image is displayed; the the initial playback delay corresponds to the time–worth of video data that has been sent by the server during the start–up delay.

During the streaming, the playback delay fluctuates as a function of varying network conditions, such as the available bandwidth, and the end–to–end transmission delay. Adaptive playback buffering mechanisms are designed to maintain a sufficient playback delay for the whole duration of the streaming, in order to accommodate network jitter (variation of the transmission delay). Such mechanisms have been derived for real–time audio applications in [87,110] and video applications in [37,68,119]. The problem that is usually addressed is to find the minimum buffer size at the receiver that smoothes out network jitter, so that the requirements of maximum late packets and maximum acceptable delays are satisfied. In [116] Sen et al. present a technique that uses client playback buffering to accommodate the streaming of VBR–encoded videos.

The playback delay should stay below a maximum value, denoted by the maximum sustainable playback delay, which is determined by the service requirements of the application. Interactive live streaming applications such as video conferencing typically require a maximum playback delay of tens of milliseconds, while non–interactive live streaming applications can usually sustain playback delays of hundreds of milliseconds.

Streaming stored video applications have relaxed delay requirements compared to live video streaming applications. So, they can sustain a higher playback delay, usually up to a few seconds. A higher playback delay allows the application to accommodate not only jitter, but also short–term variations of the available bandwidth. Indeed, with stored video all future video images are available in the server storage at any time. Therefore, in periods when the connection available bandwidth is higher than the encoded video bitrate, the server can use the extra bandwidth to send future portions of the video to the client, thereby increasing the playback delay. The extra buffered data can be used in periods when the available bandwidth becomes lower than the video source bitrate. This approach is fully rewarding if the server sends data at the maximum possible transmission rate. This is the rate given by the available TCP bandwidth, in case of transmission over TCP, or the fair share TCP–friendly rate in case of transmission over UDP. Maintaining a high playback delay can also allow the server to retransmit lost video packets before their decoding deadline expires[8].

Loss Adaptation Techniques

Error correction mechanisms through selective retransmissions or FEC should also adapt to changes in network conditions.

Because video packets have strict decoding deadlines, mechanisms for selective retransmission should be adaptive to varying end–to–end delays. Works on network–adaptive selective retransmission include [31,76,94,120,141]. In [31], Dempsey et al. propose a mechanism for selective retransmission for interactive voice communication over a channel with varying delays. Papadopoulos and Parulkar [94] present an implementation of a selective retransmission scheme with, playout buffering to increase the time available for recovery, gap–based loss detection, and receiver conditional retransmission requests to avoid triggering late retransmissions.

Network–adaptive FEC mechanisms have been studied for voice communication in [20,110]. Rosenberg et al. [110] show the need for coupling adaptive playback buffer algorithms and FEC. In [20], Bolot et al. present a scheme which adapts the amount of FEC to the available bandwidth and the current loss rate, in order to make the effective packet loss rate below a minimum target loss rate.

Finally, there are adaptive schemes that combine FEC and selective retransmission. In [109], Rhee

[8]Note that, in addition to the client playback buffer, decoders also have a very small buffer. In [99], Radha et al. present a buffer model that eliminates the separation of the playback buffer from the video decoder buffer.

presents an error–recovery technique for layered–encoded videos. Late reference images are retransmitted to stop error propagation; the base layer is protected by FEC.

Streaming Stored Layered–Encoded Video with Quality Adaptation

As we mentioned in Section 2.3.1, adding/dropping video layers is a popular technique to adapt the transmission to varying network conditions. Using layered–encoded video to adapt to changing TCP–friendly bandwidth has been addressed in [106, 113, 114].

Saparilla and Ross [113] study the optimal bandwidth allocation to the video layers. Their approach asks for one playback buffer per layer. This work introduces threshold–based network–adaptive policies that add/drop layers according to the amount of data available inside the receiver playback buffers. Subsequently, in [114], Saparilla and Ross propose a heuristic for computing near–optimal playback buffer thresholds that trigger the adding/dropping of layers. Simulations from TCP bandwidth traces show that maintaining high playback delays (up to several minutes) can improve the quality of the rendered video significantly.

In [106], Rejaie et al. define another mechanism to add and drop layers according to the state of the network and the amount of data already stored in the client buffer. This mechanism is called long–term coarse–grain adaptation. [106] also defines a fine–grain mechanism which allocates the bandwidth dynamically among the layers to accommodate bandwidth variations due to congestion control (over time–scales of RTTs); the available bandwidth is allocated to each layer so that the application can survive one or many backoffs of the congestion control mechanism.

In contrast with the fine–grain inter–layer bandwidth allocation in [106], [114] always streams the layers proportionally to their decoding rate and relies on a minimum playback delay to accommodate short time–scales bandwidth variations. The mechanism in [114] is valid for any TCP–friendly scheme, such as TCP, while [106] has only been derived for RAP, and subsequently for SQRT in [35]. RAP and SQRT are two TCP–friendly rate control schemes that have a more regular sawtooth shape than TCP, and whose properties are simpler to predict.

2.4.2 Content–Adaptive Video Streaming

Besides adapting to changing network conditions, streaming systems can benefit from adapting to the specificities of the streamed videos. The techniques presented in the previous section aim to maximize network–oriented performance measures, such as minimum loss rate or maximum bandwidth usage. While these measures can provide good indications of overall quality, in general the achieved rendering quality of the streamed video is not directly proportional to the number of bits received or the proportion of packets received. Therefore, in order to truly maximize the perceived quality at the receiver, given network conditions, network–adaptive streaming applications should also consider the specificities of the

source video and of the encoding used. This is what we call *content–adaptive streaming* of video[9].

We first present two adaptation techniques that specifically account for the layered encoding of the video: the control of quality fluctuations, and unequal error protection between video layers. Then, we present the general framework of rate–distortion optimized streaming.

Control of the Quality Fluctuations for Layered–Encoded Video

Using layered–encoded video to adapt to changing network conditions can result in high variations in quality in the rendered video. This occurs with network–adaptive systems that add and drop layers too frequently in order to adapt closely to the changing available bandwidth. High variations in quality between successive images may decrease the overall perceptual quality of the video. For example, a video with alternating high and low quality images may have the same average image quality as when the video is rendered with medium but constant image quality, but the quality perceived by the user is likely to be much lower.

The network–adaptive mechanisms presented in [106,114] both try to enforce — but do not guarantee — minimum fluctuations in quality of the rendered video. In [88], Nelakuditi et al. present an algorithm that uses layered encoding and playback buffering to smooth variations in image quality for network–adaptive video streaming. The authors design metrics that capture the smoothness of rendering of the video, and develop an off–line and on–line adaptive algorithm to maximize these metrics. As in [106, 114], a given layer is transmitted to the client only if the amount of data available in the receiver playback buffer is sufficient in order to maintain the streaming of the layer for a long time. This ensures a small number of fluctuations in the rendering quality.

In this dissertation, we investigate quality variations for switching versions and adding/dropping layers in Chapter 3. In Chapter 4 we formulate and analyze an optimization problem to find optimal transmission policies that minimize the variations in quality for streaming FGS–encoded videos.

Unequal Error Protection for Layered–Encoded Video

One of the main properties of layered–encoded video is that lower layers need to be available at the client in order to decode higher layers. In particular, without the base layer, any received enhancement layer cannot be decoded. Therefore, lower layers should be more protected against packet loss than higher layers. This is called *Unequal Error Protection* (UEP), by opposition to equal error protection (EEP). In a QoS–enabled Internet such as DiffServ (see Section 2.1.3), UEP can be performed by allocating each layer to a different class of service. In the current best–effort Internet, UEP is usually achieved by protecting each layer with different amount of FEC packets, or by retransmitting the most important

[9] We stress that content–adaptive streaming techniques can account for **both** the characteristics of the encoding used and the characteristics of the source video.

layers before the least important layers.

UEP through FEC for network–adaptive streaming of layered video has been addressed in [52, 128, 146]. Horn et al. [52] optimize the amount of redundancy to be added to each layer as a function of network parameters. Van der Schaar and Radha [128] focus on UEP for MPEG–4 FGS videos. They present a scheme to provide fine–grained unequal error protection within the FGS enhancement layer through FEC.

Mechanisms that achieve UEP of video layers through retransmissions include [75, 97, 108]. Podolsky et al. [97] study optimal strategies for delay–constrained retransmission of scalable media. The optimization problem is to find, at any given time, which packet to retransmit between an older, less important layer that expires soon, and a newer, more important layer that expires later. Results for a 2–layer video indicate that, for given network conditions, the best transmission policy is time–invariant. Rejaie and Reibman [108] propose a sliding–window approach that insures that losses of lower layers are repaired before losses of higher layers.

Rate–Distortion Optimized Streaming

Rate–distortion optimized streaming denotes a family of systems that use the rate–distortion characteristics of the particular streamed video (layered or non–layered), in order to maximize the rendering quality at the receiver. Streaming applications evaluate the end–to–end distortion of the rendered video after transmission, as a function of the distortion of the individual video packets, the interdependency between the packets, and the state of the transmission channel. This results in optimal scheduling and error correction policies that maximize the expected quality of the rendered video.

Chou and Miao [26, 27] give a general framework for evaluating the end–to–end distortion of a video for a wide range of streaming environments. They develop a heuristic algorithm for finding a sub–optimal scheduling policy. Zhang et al. [146] address rate–distortion optimized streaming in the particular context of MPEG–4 layered video with UEP through FEC; they use their own TCP–friendly algorithm, MSTFP. In [46], Frossard and Verscheure study the optimal allocation of bandwidth to the source video and FEC codes that minimizes a measure of perceptual distortion; the transmission scheme adapts to both the scene complexity and network parameters. Other works for rate–distortion optimized streaming include [84, 85, 147], which describe new ways to estimate the expected end–to–end distortion that can provide a tractable optimization.

In this dissertation, we present an analysis of the rate–distortion properties of MPEG–4 FGS–encoded videos in Chapter 5. We find useful insights for streaming algorithms, and investigate rate–distortion optimized streaming at different video frame aggregation levels. In Chapter 6 we investigate rate–distortion optimized streaming with accounting for decoder error concealment.

Chapter 3

Multiple Versions or Multiple Layers?

In this chapter we compare switching among multiple encoded versions of a video and adding/dropping encoding layers. In the context of transmission of stored video over a reliable TCP–friendly connection, we develop streaming control policies for each scheme and evaluate their performance using trace–driven simulation.

3.1 Introduction

The most straightforward mechanism to adapt the video transmission rate to the varying available bandwidth of an Internet connection, without re–encoding the video, is to store different quality versions of the video at the server and to switch between the different versions. This requires designing streaming control policies that should decide which version to stream, as a function of the current system state and observed network conditions.

With a layered–encoded video, the server can adapt to varying network conditions by just adding and dropping encoded layers. In this case, streaming control policies decide whether to add or drop a layer. Although adding/dropping layers is an effective video transmission technique, layered encoding increases the complexity of the video coding significantly, and results in inferior coding efficiency than non–layered compression (see Section 2.2.3). Layering introduces a coding overhead at the source coder and the transport layer, which is a function of several factors, including the particular layering structure employed (temporal, SNR, spatial scalability, . . .), the bitrate of the video stream, and the spatial and temporal resolution of the source [12]. Nevertheless, when the base layer is delivered with high reliability (by combining layered encoding with transport prioritization mechanisms, such as FEC), adding/dropping layers offers a higher resilience to transmission errors relative to a non–layered scheme [12, 63]. Also, the storage requirement at the server is lower with adding/dropping layers, which requires the storage of only one version of the video, than with switching between different versions of a same video.

In this chapter we make two main contributions. First, we design equivalent adaptive streaming policies for adding and dropping layers, and for switching among versions. Second, we compare the two schemes with simulations from TCP traces. Results show that, in the context of reliable TCP–friendly transmission, switching versions usually outperforms adding/dropping layers because of the bit–rate overhead associated with layering.

3.1.1 Related Work

The comparison between using layers and using versions for streaming video has been addressed in [49, 61]. Hartanto et al. [49] study caching strategies that use both versions and layers; they found that mixed strategies provide the best overall performance. In [61], Kim and Ammar study the comparison between both schemes in the context of multicast communication. In this dissertation, we focus on adaptive streaming policies for unicast communication.

This chapter is organized as follows. In Section 3.2 we present our model and the performance metrics we use to compare the two schemes. We then design streaming control policies for adding/dropping layers, and for switching versions. The heuristic streaming policies developed for each scheme are analogous, which permits a fair comparison of the two schemes. In Section 3.4 we compare the adaptive streaming schemes in simulation experiments in which we vary critical conditions, such as (i) the average available bandwidth, and (ii) the percent bitrate of overhead that is associated with layered video. Our simulation experiments use TCP throughput traces collected from Internet experiments. We conclude in Section 3.5.

3.2 Model and Assumptions

Similarly to [113, 114], we develop a fluid transmission model to analyze the video streaming problem. We denote by $X(t)$ the TCP–friendly bandwidth that is available to the streaming application at time t. For $X(t)$ we use Internet traces, with each trace being averaged over 1–second intervals. Our model allows for an initial playback delay, denoted by Δ_0, which is set to four seconds in all numerical work. We assume that the transmission delay between the server and the client is negligible. When streaming stored video, this assumption is reasonable given that round trip times are relatively small, and often an order of magnitude smaller than the maximum sustainable initial playback delay ($RTT << \Delta_0$).

We suppose that the source host always sends data at rate $X(t)$, and that all data sent into the network are eventually consumed at the receiver (i.e., the server only transmits data that will meet their deadline for consumption). The channel is supposed to be fully reliable: our model maintains a sufficient playback delay between the server and the client, so that full reliability can be ensured by the retransmission of all lost video packets. Thus, in our model, packet loss from the video stream occurs only due to client

buffer starvation. Finally, we suppose that the playback buffer at the client is not restricted in size (this assumption is motivated by disk sizes in modern PCs).

For simplicity, we suppose that the video is CBR–encoded. In the multiple versions scheme, we consider two non–layered versions, each encoded at a different rate and stored at the server. We denote the encoded bitrate of the low quality version by r_l bits per second, and the encoded bitrate of the high quality version by r_h bits per second. (Note that $r_l < r_h$.) We refer to the low quality (low bitrate) version as version v_l and to the high quality (high bitrate) version as version v_h. In the layered video scheme, we suppose that the base layer (BL) can be decoded independently to generate a comparable video quality to the the low quality version v_l. We also suppose that there is a single enhancement layer (EL), which, when decoded with the base layer, delivers a quality comparable to version v_h. We let r_b and r_e denote the rates of the base and enhancement layers, respectively. Table 3.1 summarizes the notations used in this chapter.

3.2.1 Comparison of Rates

In order to fairly compare the two streaming approaches, we need to define the relationship between the encoding bitrates of the layers and those of the two versions. As we mentioned earlier, layered encoding results in a lower compression gain than non–layered coding. The coding penalty of layering depends on several factors, including the particular scalability technique used. Kimura et al. [63] evaluated the coding overhead of SNR scalability between 5 and 20%; Wang and Zhu [137] indicated that data partitioning has about 1% overhead compared to non–layered coding.

We denote the overall percent coding overhead of layering by H. As explained in [137], the overhead introduced by layering can be due to source coding or transport, including protocol and packetization overheads. In data partitioning, for instance, the coding overhead is due to additional headers in the enhancement layer stream, which are needed for synchronization with the base layer stream. In SNR scalability, the enhancement layer is a re–quantization of the base layer at a finer resolution, so that both layers include some information about all DCT coefficients.

In this study, we assume that the bitrate coding overhead is associated with the enhancement layer. We identify the following relationships for rates r_b, r_e, and r_l, r_h :

$$r_b + r_e = (1 + H) \cdot r_h \qquad \text{and} \qquad r_b = r_l. \tag{3.1}$$

3.2.2 Performance Metrics

We compare the performance of the two adaptive streaming schemes based on three metrics:

$X(t)$	Available bandwidth at time t
r_b	Coding rate of the base layer
r_e	Coding rate of the enhancement layer
Δ_0	Initial playback delay
v_l	low quality (low bitrate) version of the video
v_h	high quality (high bitrate) version of the video
r_l	Coding rate of the low quality version v_l
r_h	Coding rate of the high quality version v_h
H	Coding overhead of layering (in percent)
t_{high}	Fraction of high quality viewing time
t_{ndisp}	Fraction of time the decoder cannot display the video
N_{fluc}	Number of fluctuations in quality
$\pi_b(t)$	Fraction of bandwidth allocated to the BL at time t
$\pi_e(t)$	Fraction of bandwidth allocated to the EL at time t
$Y_b(t)$	Content of the client BL playback buffer at time t
$Y_e(t)$	Content of the client EL playback buffer at time t
$X_{\text{avg}}(t)$	Estimate of the available bandwidth at time t
C_{pred}	Prediction interval of the available bandwidth
$Y_l(t)$	Content of the client playback buffer in version v_l at time t
$Y_h(t)$	Content of the client playback buffer in version v_h at time t
r_{high}	Ratio of the highest quality video bitrate over the average available bandwidth

Table 3.1: Summary of notations for Chapter 3

– *Fraction of high quality viewing time.* We denote by t_{high} the fraction of time that the best quality of the encoded video can be viewed at the client. In the case of layered video, t_{high} is the fraction of time both layers are delivered and decoded together. In the case of multiple versions, t_{high} is the fraction of time that version v_h is rendered to the client.

– *Fraction of time the decoder cannot display the video.* We denote by t_{ndisp} the fraction of time during which the decoder cannot render a part of the video of either quality to the receiver. In the case of layered video, t_{ndisp} is the fraction of time that the base layer playback buffer is starved. In the case of switching versions, t_{ndisp} is the fraction of time that neither v_l nor v_h data is available for consumption.

– *Quality fluctuations.* We denote by N_{fluc} the total number of times that the video quality changes at the decoder. In the case of layered video, there are quality fluctuations when the enhancement layer is dropped or added, or when the enhancement layer playback buffer at the client is starved. In the case of multiple versions, N_{fluc} is the total number of switches among the two versions, i.e., switching from version v_l to v_h, and vice versa.

3.3 Streaming Control Policies

In this section we develop control policies for adding/dropping layers and for switching among versions. A control policy for adding/dropping layers determines when to add and drop the enhancement layer, as well as how to allocate bandwidth among the layers (when both layers are being streamed). Results in [113] have shown that control policies based on the content of the playback buffers at the client attain good performance in a TCP–friendly context. Some simple threshold control policies were introduced in [114] for layered–encoded videos. In this chapter we design equivalent control policies for adding/dropping layers and for switching versions, so that we can fairly compare the two adaptive streaming schemes.

3.3.1 Adding/Dropping Layers

We begin by describing the control policy for adding/dropping layers. At any time instant, the server must determine how to allocate bandwidth among the layers. We first define some useful notation for describing the bandwidth allocation problem. We denote $\pi_b(t)$ and $\pi_e(t)$ the fraction of available bandwidth $X(t)$ that is allocated to the base layer and the enhancement layer at time t, respectively. We denote by $Y_b(t)$ and $Y_e(t)$ the contents of the base and enhancement layer playback buffers at the client at time t, respectively. Figure 3.1 illustrates the system of client playback buffers. As shown in the figure, at time t the base layer playback buffer is fed at rate $\pi_b(t)X(t)$; when nonempty, the buffer is drained at rate r_b. An analogous statement is true for the enhancement layer. We suppose that the server can estimate the amount of data in the client playback buffers using regular receiver reports (e.g., RTP/RTCP reports).

Based on the playback buffer contents at the client and on available bandwidth conditions, the server decides at each time instant whether to add or drop the enhancement layer. The goal of our control policy is to maintain continuous viewing of the base layer (i.e., avoid buffer starvation) while at the same time maximize the perceived playback quality by rendering the enhancement layer for long periods. Figure 3.2 shows a state transition diagram for the adding/dropping layers policy. The server begins by streaming only the base layer; in state 1, all available bandwidth is allocated to the base layer, i.e., $\pi_b(t) = 1$. When the enhancement layer is added, the server begins sending data from both layers and the process

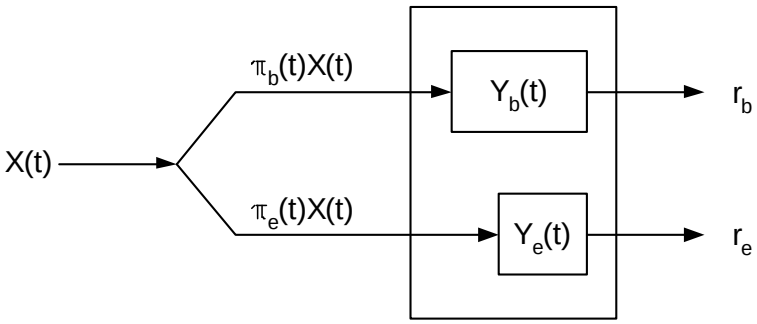


Figure 3.1: System of client playback buffers in layered streaming model

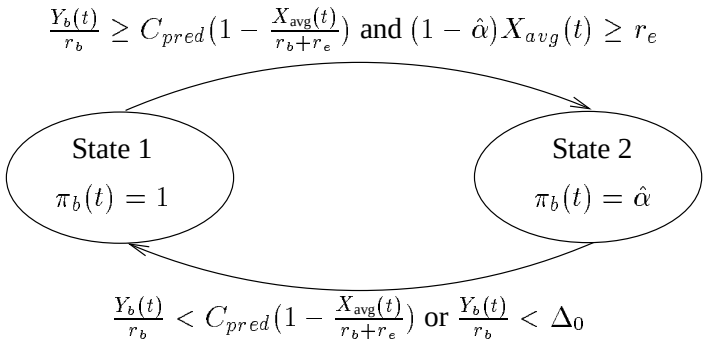


Figure 3.2: State transition diagram of a streaming control policy for adding/dropping layers

transitions to state 2. In state 2, available bandwidth is allocated among the layers in proportion to each layer's consumption rate, i.e., $\pi_b(t) = \hat{\alpha}$ where $\hat{\alpha} = \frac{r_b}{r_b+r_e}$.

Two conditions control the transition from state 1 to state 2. The first condition requires that when the server adds the enhancement layer, the amount of buffered base layer data at the client is enough to avoid future buffer starvation. In particular the server adds the enhancement layer at time s if the following condition holds:

$$\frac{Y_b(s)}{r_b} \geq C_{pred}(1 - \hat{\alpha}\frac{X_{\text{avg}}(s)}{r_b}) \tag{3.2}$$

In the above expression, $X_{\text{avg}}(s)$ is the most recent estimate of average available bandwidth. This estimate is obtained using a Weighted Exponential Moving Average (WEMA) of all previous bandwidth observations at the server. C_{pred} is a constant denoting the prediction interval over which the estimate of average bandwidth is considered useful. Condition (3.2) requires that the amount of buffered base layer data at time s is enough to avoid starvation in the next C_{pred} seconds, given that the average available bandwidth during the next C_{pred} seconds equals the most recent bandwidth estimate $X_{\text{avg}}(s)$, and that a fraction $\hat{\alpha}$ of the available bandwidth will henceforth be allocated to the base layer. The second condition for adding the enhancement layer at time s requires that the estimated transmission rate of enhancement

layer data at time s (and during the prediction interval) exceeds the consumption rate, i.e.:

$$(1 - \hat{\alpha}) \cdot X_{\text{avg}}(s) \geq r_e. \tag{3.3}$$

Condition (3.3) aims at avoiding rapid quality fluctuations caused by frequently starving the enhancement layer buffer. The server adds the enhancement layer at time s only if both conditions (3.2) and (3.3) hold.

The server drops the enhancement layer when the likelihood of base layer buffer starvation becomes high. To avoid starvation of the base layer buffer, the server drops the enhancement layer at time s if (3.2) does not hold. The server also drops the enhancement layer, regardless of the estimated bandwidth conditions, if the amount of buffered data drops below the amount needed to mitigate jitter and short time scale bandwidth variations, i.e., if $Y_b(t) < r_b \cdot \Delta_0$, where Δ_0 is the initial playback delay in seconds.

A key issue in the design of the adaptation mechanism for adding/dropping layers is determining which portion of the enhancement layer to transmit once the layer has been added. A straightforward implementation is to send the enhancement layer data with the same playback deadline as the base layer currently being transmitted. In other words, the base and enhanced portions of each future frame are transmitted together. As a result, the base layer data that are already in the client playback buffer are consumed without enhancement.

3.3.2 Switching Versions

Now, we develop a streaming control policy for switching versions. Figure 3.3 shows the client playback buffer in the version streaming model. We suppose that there is a unique playback buffer that contains video parts from both versions[1]. We denote the playback buffer contents in version v_l at time t by $Y_l(t)$. Similarly, we denote the playback buffer contents in version v_h at time t by $Y_h(t)$. As we shall see, to minimize the risk of buffer starvation, the server should not stream highest version v_h, unless a reserve of future data exists in either one of the two versions. Once again, we suppose that the amount of buffered data at the client can be estimated at the server using regular receiver reports.

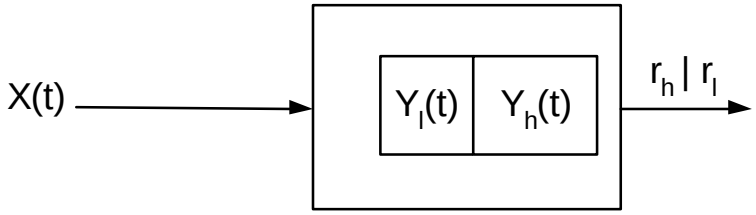


Figure 3.3: System of client playback buffers in versions streaming model

[1]Our control policy is also valid for a model with one buffer for each version of the video.

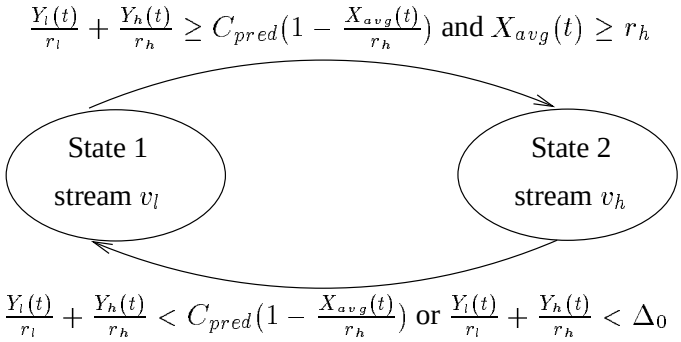


Figure 3.4: State transition diagram of a streaming control policy for switching versions

Figure 3.4 shows the state transition diagram for the switching versions policy. The server begins by streaming version v_l at the available bandwidth rate. After the initial playback delay, the client begins to consume buffered data from version v_l. Whenever $X(t)$ exceeds r_l, future portions of version v_l are stored into the corresponding client playback buffer. In a similar way as in the heuristic for adding/dropping layers, the server switches to version v_h if two conditions hold. The first condition requires that the total amount of buffered data at the client is enough to avoid buffer starvation during the next C_{pred} seconds, i.e., the server switches to v_h at time s if:

$$\frac{Y_l(s)}{r_l} + \frac{Y_h(s)}{r_h} \geq C_{pred}(1 - \frac{X_{\text{avg}}(s)}{r_h}). \tag{3.4}$$

The second condition for switching to v_h at time s requires that $X_{\text{avg}}(s) \geq r_h$. Similar to the adding/dropping layers heuristic, this condition minimizes the quality fluctuations caused by frequently switching among the versions. The server continues streaming v_h until the likelihood of buffer starvation becomes high, or until the amount of data buffered at the client falls below the required initial build–up, i.e., the server switches back to v_l if condition (3.4) does not hold, or if $\frac{Y_1(s)}{r_1} + \frac{Y_2(s)}{r_2} < \Delta_0$. To permit the fair comparison of adding/dropping layers and switching versions, $X_{avg}(s)$ is computed using the same estimation procedure (i.e., WEMA) as in the case of adding/dropping layers.

Similar to our implementation of adding/dropping layers, the server transmits data from v_h beginning with the first video frame that has not yet been buffered in v_l. In this case, the data from v_l that are in the playback buffer are decoded and displayed in their entirety before the consumption of version v_h data begins.

3.3.3 Enhancing Adding/Dropping Layers

In our proposed mechanism for adding/dropping layers in Section 3.3.1, the server streams both layers synchronously, i.e., it streams the base and enhanced streams pertaining to the same portion of the video

Trace	Source – Destination	date	Throughput (Mbps)		
			Peak	Mean	Std.
A1	US-FR	29-06-99 15:00	2.41	0.70	0.43
A2	US-FR	29-06-99 16:00	3.89	1.10	0.82

Table 3.2: Summary of 1-hour long traces.

at the same moment. This implementation does not fully take advantage of the flexibility offered by layering. Indeed, when conditions are favorable enough to stream the enhancement layer, the server can enhance the base layer stream that is stored in the client buffer, instead of simply enhancing the part of the base layer currently being sent. By transmitting the enhancement layer data with the earliest playback deadline, the playback quality at the client can be enhanced immediately. We refer to this variation of the adding/dropping layers control policy as the *immediate enhancement* mechanism.

Immediate enhancement of the playback quality in the switching versions scheme can also be implemented. The server can switch to version v_h by transmitting the v_h data with the earliest playback deadline. We note, however, that in the case of versions, the immediate enhancement mechanism results in a waste of bandwidth: the portion of the v_l data stored in the playback buffer is useless, so it should be deleted from the buffer.

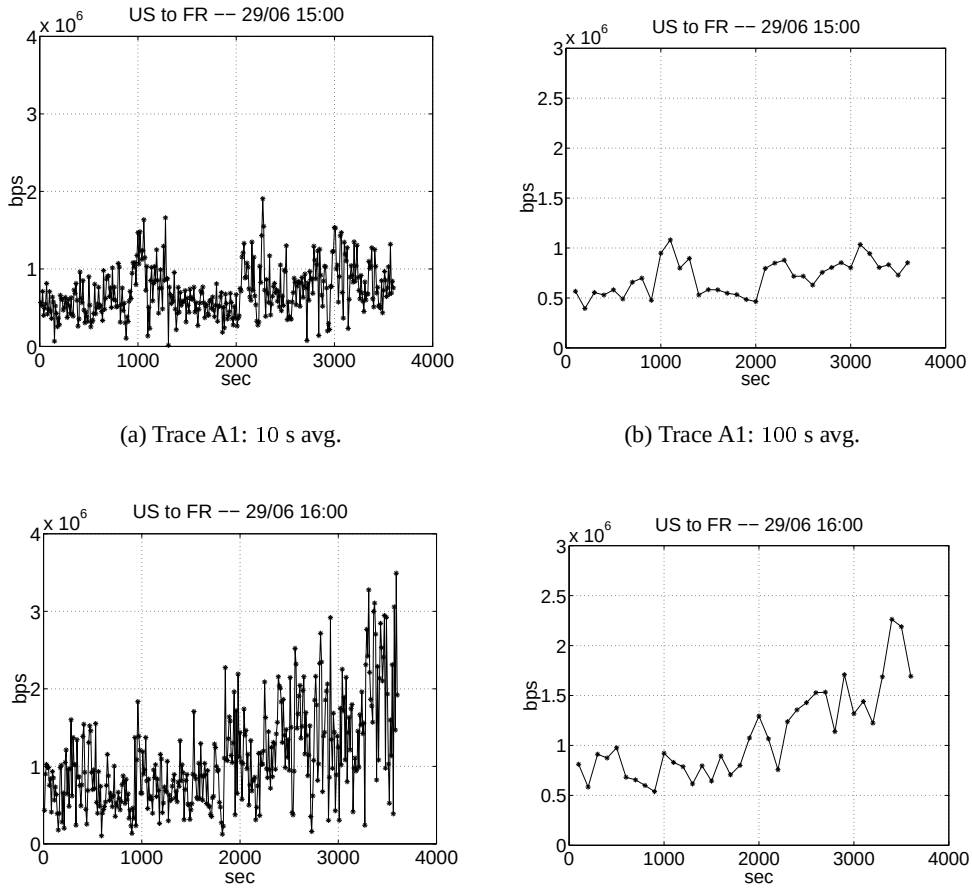
3.4 Experiments

We compare adding/dropping layers and switching versions in simulation experiments. The simulations implement the streaming control policies described in the previous section, and use TCP traces as explained below.

3.4.1 TCP–Friendly Traces

We do not use any of the TCP–friendly rate adjustment schemes in the literature to generate TCP–friendly traces. Instead, it is natural to suppose that the perceived rate fluctuations of a TCP–friendly scheme exhibit similar behavior as the fluctuations of TCP throughput over medium (seconds) and long (minutes) time scales. To obtain TCP–friendly bandwidth conditions we use throughput traces from TCP connections on the Internet.

We used two 1–hour long instantaneous throughput traces for TCP flows, denoted by A1 and A2. These were collected at different times of the day between a host in the United States and a host in France. Table 3.2 summarizes statistics for the two traces, and Figure 3.5 shows the average throughput of these traces over time scales of 10 and 100 seconds. As shown in the figure, both traces exhibit a



(a) Trace A1: 10 s avg.

(b) Trace A1: 100 s avg.

(c) Trace A2: 10 s avg.

(d) Trace A2: 100 s avg.

Figure 3.5: Average throughput over time scales of 10 and 100 seconds for traces A1 and A2.

high degree of variability and burstiness over both time scales. We denote by $\bar{X}$ the average available bandwidth for the duration of the connection.

3.4.2 Numerical Results

Table 3.3 summarizes results obtained with trace A1. The two schemes are compared based on the three performance metrics discussed in Section 3.2.2 when the percent bitrate overhead associated with layering, H, varies between 1 and 10%. Metrics t_{high}, t_{ndisp} and N_{fluc} are studied for different video consumption rates. Although, the video consumption rate is approximately constant for a certain quality video, we choose to vary the consumption rate in order to study the behavior of our heuristic policies under several bandwidth conditions. Parameter r_{high} is the ratio of the highest quality video consumption rate for the case of versions (i.e. r_h) over the average trace bandwidth, $\bar{X}$. We distinguish between the

Scheme	$r_{high} = 0.7$			$r_{high} = 1.0$			$r_{high} = 1.3$		
	t_{high}	t_{ndisp}	N_{fluc}	t_{high}	t_{ndisp}	N_{fluc}	t_{high}	t_{ndisp}	N_{fluc}
Versions	**98.42**	0	1	**85.12**	0	1	**52.68**	0	7
Layers $H = 0\%$	98.42	0	1	85.12	0	1	52.68	0	7
Layers $H = 1\%$	98.42	0	1	81.92	0	3	48.1	0	7
Layers $H = 5\%$	98.22	0	1	77.23	0	3	49.15	0	9
Layers $H = 10\%$	95.92	0	3	72.82	0	1	43.23	0	10
Layers–imm $H = 0\%$	99.03	0	3	**85.45**	0	19	**53.61**	0	37
Layers–imm $H = 1\%$	98.97	0	3	83.68	0	21	51.01	0	41
Layers–imm $H = 5\%$	**98.72**	0	3	79.93	0	21	49.15	0	49
Layers–imm $H = 10\%$	97.25	0	5	75.99	0	23	44.38	0	55
Versions–imm	97.64	0	3	76.42	0	13	34.41	2.2	26

Table 3.3: Results for trace A1

Scheme	$r_{high} = 0.7$			$r_{high} = 1.0$			$r_{high} = 1.3$		
	t_{high}	t_{ndisp}	N_{fluc}	t_{high}	t_{ndisp}	N_{fluc}	t_{high}	t_{ndisp}	N_{fluc}
Versions	**94.31**	0	5	**57.66**	0	5	**43.78**	0.4	5
Layers–imm $H = 0\%$	95.56	0	9	62.36	0	21	44.41	0.4	19
Layers–imm $H = 1\%$	**95.06**	0	11	62.07	0	21	**43.94**	0.4	21
Layers–imm $H = 5\%$	91.71	0	17	**60.34**	0	25	41.27	0.4	25
Layers–imm $H = 10\%$	88.22	0	27	57.23	0	25	38.67	0.4	31

Table 3.4: Results for trace A2

following cases:

- $r_{high} = 1$ means that the transmission channel can accommodate, on average, the streaming of the high quality version of the video (v_h),
- $r_{high} > 1$ corresponds to the unfavorable case when the average available bandwidth is not sufficient to stream the high quality version of the video, i.e. $r_h > \bar{X}$,
- $r_{high} < 1$ corresponds to the favorable case when the average available bandwidth exceeds the coding rate of the high quality version of the video, i.e. $r_h < \bar{X}$.

In Table 3.3, *Layers-imm* and *Versions-imm* represent the immediate enhancement mechanisms, for adding/dropping layers and switching versions respectively.

The results shown in the first two rows of Table 3.3 indicate that the performance of adding/dropping layers and switching versions are identical when $H = 0\%$. This confirms that the adaptation mechanisms

used in the two schemes are equivalent. When $H = 1\%$, the performance of adding/dropping layers deteriorates as bandwidth conditions become less favorable (i.e., for $r_{high} = 1.0$ and $r_{high} = 1.3$). In general, higher coding overhead results in further performance degradation of the adding/dropping layers scheme. When $H = 10\%$, the fraction of high quality viewing time t_{high} is lower in the case of layers than in the case of versions under all r_{high} values. The fluctuations in quality (N_{fluc}) generally increase with H, although N_{fluc} remains reasonably low in all cases. Finally, we note that t_{ndisp} is in all cases equal to 0, indicating that the video is delivered to the client without interruption. The low value of t_{ndisp} and N_{fluc} demonstrate that our streaming policies for both schemes can adapt to the varying available bandwidth with very good performance.

We next consider the performance of adding/dropping layers and switching versions when the immediate enhancement mechanism is employed. Results in Table 3.3 indicate that with no overhead ($H = 0\%$), adding/dropping layers performs slightly better than switching versions in terms of the fraction of high quality viewing time, t_{high}. When $r_{high} = 0.7$, adding/dropping layers with immediate enhancement attains higher t_{high} than switching versions (regardless of whether immediate enhancement is employed in the case of versions) for coding overhead values as high as 5%. Under more adverse bandwidth conditions (i.e., $r_{high} = 1.0$ and $r_{high} = 1.3$), the immediate enhancement mechanism in the case of layers does not offer sufficient improvement in performance in the presence of coding overhead. Indeed, when bandwidth conditions are scarce, the amount of data buffered at the client at any time is small. As a result the immediate enhancement scheme is overly aggressive, resulting in high N_{fluc} and no significant improvement in t_{high}.

Our results also demonstrate the inefficiency of employing the immediate enhancement mechanism in the case of versions: this results in both lower t_{high} and higher N_{fluc}. As discussed earlier in Section 3.3.3, immediate improvement in playback quality results in a waste of bandwidth in the case of versions. The utilization of available resources is less efficient, resulting in decreased overall performance.

Table 3.4 shows results obtained with trace A2 for switching versions with no immediate enhancement and adding/dropping layers with immediate enhancement. We observe that the adding/dropping layers scheme can result in higher t_{high} than switching versions, even in the presence of coding overhead. This is true for coding overheads up to 1% when $r_{high} = 0.7$ or $r_{high} = 1.3$, and up to 5% when $r_{high} = 1.0$. Again, for all values of r_{high}, increased performance in terms of t_{high} is attained at the expense of quality fluctuations. Note that for $r_{high} = 1.3$ both schemes experience 0.4% of playback buffer starvation, whatever the value for H.

3.5 Conclusions

In this chapter we have designed equivalent streaming policies for adding/dropping layers and for switching among versions, in order to compare the two adaptation schemes under different critical conditions. In the context of reliable transmission of stored video, our simulations showed that our heuristics for both schemes successfully adapt the streaming to exploit the available TCP–friendly bandwidth. But, in the simplest implementation, the overhead introduced by layering makes switching versions always perform better than adding/dropping layers.

When using the immediate enhancement implementation for adding/dropping layers, neither scheme seems to dominate: for low values of the layering coding overhead, the enhanced flexibility provided by layering can compensate the loss in high quality viewing time due to the overhead, but at the expense of more fluctuations in quality. For high values of the coding overhead ($H \geq 10\%$), switching versions reaches better performance than layers with immediate enhancement in all bandwidth situations.

In conclusion, when streaming video over a reliable connection, it is crucial to use efficient layered encoding schemes, in order to keep the bitrate overhead of layering small; otherwise, switching among different non–layered versions of the video achieves better performance[2].

We have investigated the comparison between layering and versions, for conventional scalable encoding schemes that encode the video into a specified limited number of layers. In contrast to conventional scalable coding, Fine Granularity Scalability is a new coding scheme that allows the server to cut the enhancement layer bitstream into an arbitrary number of layers, during transmission (see Section 2.2.4). With the fine granularity property, streaming control policies for FGS–encoded layers can flexibly adapt to changes in the available bandwidth. This constitutes a major advantage over conventional layered encoding, as well as over switching versions, where the bitrate of each version is fixed. In Chapter 4 we study optimal control policies for streaming of stored FGS–encoded videos.

[2]This result for a reliable connection contrasts with the case of a lossy connection, for which it has been shown that layering gives better performance than using non–layered video [12, 63].

Chapter 4

Streaming Stored FGS–Encoded Video

In this chapter, we investigate adaptive streaming of stored FGS video. We present a novel framework for low–complexity streaming of FGS video over a reliable TCP–friendly connection. We derive an optimal transmission policy for a criterion that involves both image quality and quality variability during playback. Based on this ideal optimal policy, we develop a real–time heuristic to stream FGS video over the Internet. We study its performance using real Internet traces, and simulations with an MPEG–4 FGS streaming system.

4.1 Introduction

Fine Granularity Scalability (FGS) has recently been added to the MPEG–4 video coding standard [8] in order to increase the flexibility of video streaming. With FGS coding the video is encoded into a base layer (BL) and one enhancement layer (EL). In contrast to conventional scalable video coding, which requires the reception of complete enhancement layers to improve upon the basic video quality, with FGS coding the enhancement layer stream can be cut anywhere before transmission. The received part of the FGS enhancement layer stream can be successfully decoded and improves upon the basic video quality (see Section 2.2.4 for a more detailed overview of FGS coding). The FGS enhancement layer can be cut at the granularity of bits. This fine granular flexibility was the key design objective of FGS coding, along with good rate–distortion coding performance. With the fine granularity property, FGS–encoded videos can flexibly adapt to changes in the available bandwidth. This flexibility can be exploited by video servers to adapt the streamed video to the available bandwidth in real–time (without requiring any computationally demanding re–encoding).

The first contribution of this chapter is a new framework for streaming stored FGS video over a reliable TCP–friendly connection. Our framework is intended to be valid for any 2–layer FGS–encoded video, and not only MPEG–4 FGS videos. This new framework calls for client playback buffering,

and synchronous transmission across base and enhancement layers. Policies that operate within this framework require minimum real–time processing, and are thus suitable for servers that stream a large number of simultaneous unicast streams.

Our second contribution is to formulate and solve an optimal streaming problem. Our optimization criterion is based on simple and tractable metrics, which account for the total video display quality as well as variability in display quality. We develop a theory for determining an optimal streaming policy under ideal knowledge of the evolution of the future bandwidth. The optimal ideal policy provides bounds on the performance of real–time policies and also suggests a real–time heuristic policy. Simulations from real Internet traces show that the heuristic performs almost as well as the ideal optimal policy for a wide-range of scenarios. We present an implementation of our framework and heuristic in an MPEG–4 streaming platform.

We also compare streaming stored–FGS video over an ordinary TCP connection to streaming over a TCP–friendly connection. The performance of current TCP–friendly congestion control algorithms is usually assessed in terms of their fairness with TCP, responsiveness to changes in network congestion and smoothness of throughput [145]. Because popular TCP–friendly algorithms have an available bandwidth that is typically smoother than TCP available bandwidth, one expects TCP-friendly algorithms to perform better, particularly for reducing quality fluctuations. However, our experiments show that video quality fluctuations are in the same range for both TCP and TCP–friendly algorithms.

4.1.1 Related Work

The streaming of FGS–encoded video has recently received significant attention. Tan and Zakhor [121] describe a low–latency error resilient scheme for real–time streaming of fine granular videos over a TCP–friendly connection. Fine granularity is achieved using a 3–D subband decomposition. General frameworks for MPEG–4 FGS video streaming have been given by Radha et al. in [99, 100, 128]: [99] presents a mechanism for retransmitting video packets using client buffering; [100] highlights the flexibility of MPEG–4 FGS for supporting multicast and unicast streaming applications over IP; and [128] focuses on the error resilience properties of the FGS enhancement layer.

An efficient approach for the decoding of streamed FGS video is proposed in [124]. Streaming of FGS video over multicast [132] and to wireless clients [129] has also been considered, while issues of FGS complexity scaling and universal media access are addressed in [24]. A streaming mechanism for the FGS–temporal enhancement layer is studied in [62]. Finally, reducing quality variations when streaming FGS–encoded videos, using specific encoding of the base layer, has been addressed in [150].

In this dissertation, we present a novel optimization framework for unicast streaming of stored FGS–encoded videos over a TCP–friendly connection. We derive a real–time algorithm and compare its performance with the performance of optimal policies. In Section 4.7, we present simulations with an

MPEG–4 FGS streaming platform. To our knowledge, our platform is one of the first implementations of an Internet streaming application which combines the use of FGS–encoded video, a network–adaptive algorithm, and video scene–based segmentation, in order to smooth perceptual changes in image quality, while maintaining high bandwidth efficiency. In [146], Zhang et al. also present a complete end–to–end system that streams FGS–encoded videos, but using a specific smooth TCP–friendly protocol (MSTFP); as for the system from Radha et al. [99], it does not aim at smoothing fluctuations in image quality between consecutive images.

This chapter is organized as follows. In Section 4.2 we present our framework for streaming FGS–encoded video over a single TCP–friendly connection. In Section 4.3 we formulate the optimization problem and define the performance metrics considered in this chapter. In Section 4.4 we develop a theory for optimal streaming under ideal knowledge of future bandwidth evolution. In Section 4.5, we present our real–time rate adaptation heuristic, which is inspired from the optimization theory. We use simulations with Internet traces to study the performance of the heuristic. In Section 4.6 we compare the performance of our heuristic when run on top of TCP to when run on top of popular TCP–friendly algorithms. Finally, in Section 4.7 we present experiments with a streaming platform for MPEG–4 FGS–encoded videos, and we conclude in Section 4.8.

4.2 Framework

As in Chapter 3, we denote by $X(t)$ the available bandwidth at time t. By permitting playback buffering into client buffers, our server streams the video at the maximum rate $X(t)$ at each instant t. We suppose that the connection is made reliable, e.g., by using retransmissions [76], so that losses may only occur due to missed deadlines. Selective retransmissions are possible because of client buffering. Buffering also allows us to neglect in our analysis the transmission delay between the server and the client.

The stored video is encoded into two layers, the base layer and the FGS enhancement layer. For simplicity of the analysis, we assume that the video is CBR–encoded. We denote the encoding rates of the base and enhancement layers by r_b and r_e, respectively. The length of the video is denoted by T seconds.

Figure 4.1 shows the architecture of the server. The server stores the video as two separate files: one file contains the bitstream pertaining to the base layer and the other file contains the bitstream pertaining to the enhancement layer. Because the server transmits at maximum rate $X(t)$, at any given instant of time the server may be sending frames to the client that are minutes into the future. To reduce the server complexity, we require that the server always sends the base and enhancement instances of the same frame together, thus acting synchronously. This implies that the number of base layer frames that are stored into the client playback buffer is always equal to the number of stored enhancement layer frames.

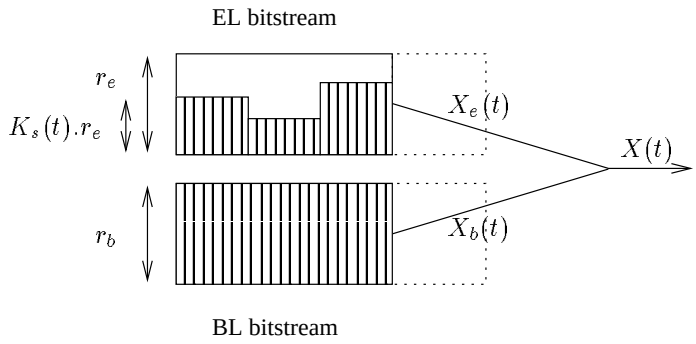


Figure 4.1: Server model

For each transmitted frame, the server sends the entire base layer of the frame and a portion of the enhancement layer of the frame. Because of the fine granular property of the enhancement layer, the server can truncate the enhancement layer portion of the frame at any level. Thus, at each instant, the server must decide how much enhancement layer data to send. In our design, time is broken up in slots $[t_k, t_{k+1})$, for $k = 0, \ldots, N_{slot} - 1$, where $t_0 = 0$ and $t_{N_{slot}} = T$. At the beginning of a slot $[t_k, t_{k+1})$, the server determines the enhancement layer level, denoted by $K_s(k) \cdot r_e$, that it streams for the duration of the slot. Thus, as shown in Figure 4.1, all frames sent during slot $[t_k, t_{k+1})$ include the entire base layer and a same fraction of the enhancement layer, $K_s(k) \in [0, 1]$. The frames sent during the slot may be frames for display seconds or even minutes into the future. The length of a slot can be chosen so that the slot is composed of one or more complete video scenes. This would keep constant the fraction of enhancement layer that is transmitted for each video scene, avoiding changes in perceptual image quality within the same video scene[1]. The length of a time slot should be on the order of seconds, which also helps to maintain low server complexity. For simplicity, we assume that the slot length is equal to a constant C_{slot}, so that we can write $t_k = k \cdot C_{slot}$, for $k = 0, \ldots, N_{slot}$.

The rate at which the server transmits frames into the network depends on the available bandwidth during the slot, i.e $X(t)$ for $t_k \leq t < t_{k+1}$. Because we are requiring that the base and enhancement layer components of a frame be sent at the same time, the available bandwidth dedicated to the base layer and to the enhancement layer at time $t \in [t_k, t_{k+1})$ is respectively:

$$X_b(t) = \frac{r_b}{r_s(k)} X(t) \text{ and } X_e(t) = \frac{K_s(k) \cdot r_e}{r_s(k)} X(t) \tag{4.1}$$

where $r_s(k) = r_b + K_s(k) \cdot r_e$ is the total coding rate of the video being streamed between times t_k and t_{k+1}. By extension, the total coding rate of the video being streamed at time t is denoted by $r_s(t) = r_b + K_s(t) \cdot r_e$.

[1]We address this matter into more details in Chapter 5.

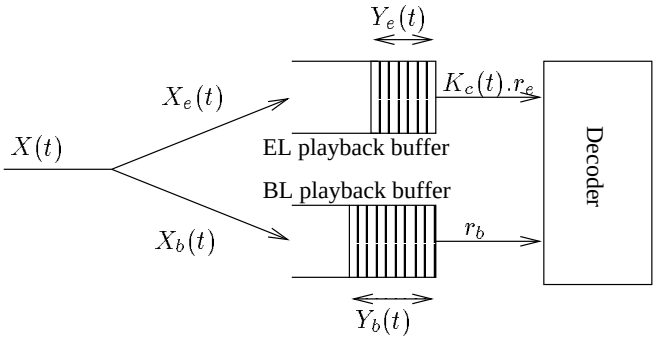


Figure 4.2: Client model

Figure 4.2 illustrates the architecture of the client. The client stores temporarily the data coming from the network in base and enhancement layer playback buffers. Let $Y_b(t)$ and $Y_e(t)$ denote the amount of data stored in the playback buffers at time t. At time t, the decoder drains both buffers at rates r_b and $K_c(t) \cdot r_e$, where $K_c(t) \in [0, 1]$ is the fraction of enhancement layer available at the client for the frame which is scheduled to be decoded at time t. The encoding rate of the video being displayed at the client at time t can be expressed as $r_c(t) = r_b + K_c(t) \cdot r_e$.

As in Chapter 3, we denote by Δ_0 the initial playback delay between the server and the client. At time $t = 0$, the client starts to decode the data from its playback buffers and to render the video, while the server streams the rest of the video frames. To simplify our analysis, we suppose that, once the playback starts at the client, the user is going to watch the video until the last frame without performing any VCR command. Let t_{end} denote the time when the server stops sending video frames. We have $t_{end} < T$ if the server has sent the last video frame before video is fully rendered; otherwise, we have $t_{end} = T$.

We denote $\Delta(t)$ for the playback delay at time t. Since we neglect transmission delays, $\Delta(t)$ corresponds to the number of seconds of video stored in the client playback buffers at time t. Since the server acts synchronously and each layer is streamed in proportion to its encoding rate, the base and enhancement playback buffers always have the same value of $\Delta(t)$ for each time t. Because the server changes the encoding rate of the enhancement layer at each time t_k, the enhancement layer playback buffer contains parts of the video encoded at different bit–rates. However, because the base layer is never truncated, we can write at each time t, $\Delta(t) = \frac{Y_b(t)}{r_b}$. Since, at time t, the server streams a video frame which is going to be decoded by the client $\Delta(t)$ seconds into the future, the total coding rate of the video which is decoded at time t can be expressed as:

$$r_c(t + \Delta(t)) = r_s(t), \quad \text{or} \quad r_c(t) = r_s(t - \Delta(t)) \tag{4.2}$$

(recall that $r_s(t)$ is the total coding rate of the video being streamed at time t).

To simplify notation, we denote by Δ_k the playback delay at beginning of slot $[t_k, t_{k+1})$, i.e. $\Delta_k = \Delta(t_k)$. We assume that, at each time t_k, the server knows the value of Δ_k (e.g., through periodical receiver reports).

Because we suppose that the connection is made reliable, losses at the client only occur when data arrive at the client after their decoding deadline. Such data are not decoded. Assuming a playback delay of $\Delta_k > 0$ at time t_k, losses may only start to happen at time $t_k < t < t_{k+1}$ when $\Delta(t) = 0$ and when $X(t) < r_s(t)$, i.e., when the client buffers are empty and the available bandwidth is not high enough to feed the decoder at the current video encoding rate. Since the server acts synchronously for both layers and each layer is sent in proportion to its coding rate, losses can only happen for both layers at the same frame time. When there is loss at time $t \in (t_k, t_{k+1})$, we do not suppose that the server is able to react: it keeps streaming the current part of the video, even if the frames will not meet their decoding deadline. Meanwhile, the client keeps incrementing its decoding time–stamp and waits for the part of the video that has the new current decoding deadline. A negative $\Delta(t)$ indicates that the part of video arriving from the server has an earlier decoding deadline than the part of the video that the client is waiting for. Therefore, there is loss of data whenever $\Delta(t) < 0$.

Table 4.1 summarizes the notations used in this chapter.

4.3 Problem Formulation

A *transmission policy*, denoted by $\boldsymbol{r}_s = (r_s(0), \ldots, r_s(N_{slot} - 1))$, is a set of successive video encoding rates, each of which is chosen by the server at the beginning of a time slot $[t_k, t_{k+1})$, for video sequence k. In order to provide the user with the best perceived video quality, the transmission policy can minimize a measure of total distortion [27, 146], such as the commonly employed MSE (Mean Squared Error), and can as well minimize variations of distortion between successive images. However, such optimization problem is usually difficult to solve, because each video sequence may have different rate–distortion characteristics. Also, as we show in Chapter 5, for a given video sequence k, the distortion of a frame does not vary linearly with the sequence chosen coding rate $r_s(k)$.

In this study, we restrict to transmission policies that (i) ensure a minimum of quality, by ensuring the decoding of the base layer data without loss, (ii) maximize the bandwidth efficiency, i.e., the total number of bits decoded given the available bandwidth, which gives a good indication of the total video quality, and (iii) minimize the variations of the rendered coding rate between successive video sequences, which gives an indication of the variations in distortion.

Although these metrics are simple and independent of the rate–distortion characteristics of a particular video, our analysis remains useful for deriving real–time heuristics, and for comparing different transport protocols.

$X(t)$	Available bandwidth at time t
r_b	Coding rate of the base layer
r_e	Coding rate of the enhancement layer
T	Length of the video (in seconds)
$K_s(k)$	Proportion of the FGS–EL sent by the server at time t_k
$K_c(t)$	Proportion of the FGS–EL decoded by the client at time t
$r_s(k)$	Total coding rate of the video sent between t_k and t_{k+1}
$r_c(t)$	Total coding rate of the video decoded at time t
C_{slot}	Length of a server slot (in seconds)
N_{slot}	Number of time slots
$X_b(t)$	Available bandwidth dedicated to the BL at time t
$X_e(t)$	Available bandwidth dedicated to the EL at time t
$r_s(k)$	Coding rate of the video that is sent between times t_k and t_{k+1}
$\boldsymbol{r_s}$	Transmission policy at the server, $= (r_s(0), r_s(1), \ldots, r_s(N_{slot}-1))$
t_{end}	Ending time of the streaming
$\Delta(t)$	Playback delay at time t
Δ_k	Playback delay at time t_k
E	Bandwidth efficiency
V	Coding rate variability
N_{last}	Last time slot of the streaming
$X_{avg}(k)$	Average goodput of slot k
α	Smoothing factor
r_{low}	Ratio of the base layer coding rate over the average available bandwidth
N_{seg}	Number of video segments
$r_e(k)$	Coding rate of EL to stream to the client for video segment k
Δ_{end}	content of the client playback buffers when the server has finished streaming the video
V_{seg}	Variability in quality between successive video segments

Table 4.1: Summary of notations for Chapter 4

4.3.1 Bandwidth Efficiency

We define the bandwidth efficiency E as the ratio between the average number of bits decoded at the client by seconds over the total rate of the video:

$$E = \frac{\bar{r_c}}{r_b + r_e} \text{ where } \bar{r_c} = \frac{\text{total number of bits decoded}}{T} \tag{4.3}$$

Observing that the video data at the receiver may come either from the initial build up or from the streaming, we can write :

$$E = \frac{\Delta_0 \cdot (r_b + r_e) + \int_0^{t_{end}} X(t)dt - \text{(nb of bits lost)}}{T \cdot (r_b + r_e)} \tag{4.4}$$

Recall that the "number of bits lost" is the number of bits sent by the server that do not make their deadline at the client.

4.3.2 Coding Rate Variability

Previous studies in [88] have designed various measures to account for the rate variability in the case of layered video with a small number of layers. Here, we propose a measure for the case of FGS encoding, for which the displayed video encoding rate $r_c(t)$ can take continuous values between r_b and $r_b + r_e$. Since from (4.2), $r_c(t) = r_s(t - \Delta(t))$, and since $r_s(t) = r_s(k)$ over all intervals $[t_k, t_{k+1})$, differences in consecutive values for $r_c(t)$ at the client are accounted by differences in consecutive values for $r_s(k)$ at the server. Therefore the following measure accounts for differences in consecutive values for the encoding rate of the video being displayed to the user:

$$V = \frac{1}{\bar{r_s}} \sqrt{\frac{1}{N_{last}} \sum_{k=0}^{N_{last}-1} [r_s(k) - r_s(k+1)]^2} \tag{4.5}$$

where $N_{last} < N_{slot}$ is the index of the last time slot during which the server has data left to stream, i.e., the streaming ends at time $t_{end} \in [t_{N_{last}}, t_{N_{last}+1})$. $\bar{r_s}$ denotes the mean value of the time series $\{r_s(k)\}$, for $k = 0, \ldots, N_{last}$.

Because the human eye is more likely to perceive a high variation in quality than a small one, this measure penalizes high differences in consecutive values for $r_s(k)$. Moreover, in FGS coding, the less important bit planes correspond to higher values of $r_s(k)$. Therefore, for the same video with both layers encoded at the same r_b and r_e, the higher the mean value of the transmitted coding rates $r_s(k)$, the less visible the differences in consecutive values for $r_s(k)$. This is why our measure of rate variability is normalized by $\bar{r_s}$.

4.4 Optimal Transmission Policy

In this section we assume that the available bandwidth for the connection, $X(t)$, from beginning to end of transmission, is known a priori. This allows us to formulate and solve an optimal stochastic control problem. The analysis and solution serves two purposes. First, it provides a useful bound on the achievable performance when bandwidth evolution is not known a priori. Second, the theory helps us design an adaptation heuristic for the realistic case when the bandwidth is not known. The optimization criteria studied in this chapter prioritize three metrics: base layer loss, bandwidth efficiency, and coding rate fluctuations.

4.4.1 Condition for No Losses

Losses of base layer data at the decoder degrade considerably the perceived video quality. Depending on the level of error resilience used by the coding system, losses may cause freezing of the image for some time. Thus, base layer losses can be more disturbing for the overall quality than the number of bits used to code the video (represented by $\int_0^{t_{end}} X(t)dt$ in (4.4)). In this subsection we determine a necessary and sufficient condition for the transmission policy $\boldsymbol{r}_s = (r_s(0), \ldots, r_s(N_{slot}-1))$, to have no base layer loss. Recall that in our synchronous model no base layer loss implies no enhancement layer loss. To this end, denote:

$$\beta_k(\Delta) = \min_{t \in [t_k+\Delta, t_{k+1}]} \frac{\int_{t_k}^{t} X(u)\, du}{t - (t_k + \Delta)} \tag{4.6}$$

Theorem 4.1

The transmission policy $(r_s(0), \ldots, r_s(N_{slot}-1))$ *yields no loss of data over the whole decoding duration* if and only if*, for all* $k = 0, \ldots, N_{slot}-1$*,*

$$r_s(k) \leq \beta_k(\Delta_k) \text{ whenever } \Delta_k < C_{slot} \tag{4.7}$$

This theorem provides, for each slot, an upper–bound on the video coding rate that yields no loss. This bound depends on the content of the playback buffers at the beginning of the slot, Δ_k, and on the available bandwidth for the duration of the slot (in $\int_{t_k}^{t} X(u)\, du$).

Proof. Having no loss of data over $[0, T]$ is equivalent to having no loss of data over each interval $[t_k, t_{k+1})$. Fix a $k \in \{0, \ldots, N_{slot}-1\}$. If $\Delta_k \geq C_{slot}$, there is enough data in the client playback buffers at time t_k to insure the decoding without loss during the current slot $[t_k, t_{k+1})$ of length C_{slot} seconds.

Now suppose that $\Delta_k < C_{slot}$. Clearly there is no loss in the interval $t \in [t_k, t_k + \Delta_k]$, as the data in the playback buffers at time t_k is sufficient to feed the decoder up through time $t_k + \Delta_k$. At time $t_k + \Delta_k$ all of the data that was already in the playback buffer at time t_k is consumed. Subsequently, the client starts to consume data that was sent after time t_k, which has been encoded at rate $r_s(k)$. Thus, after time $t_k + \Delta_k$, and at least up to time t_{k+1}, the client attempts to consume data at rate $r_s(k)$.

It follows that there is no loss if and only if for every time $t \in [t_k + \Delta_k, t_{k+1}]$ the total amount of data that the decoder attempts to consume in $[t_k + \Delta_k, t]$ is less than the amount of data that was transmitted in the interval $[t_k, t]$ with coding rate $r_s(k)$, that is, if and only if for all $t \in [t_k + \Delta_k, t_{k+1}]$

$$r_s(k) \cdot [t - (\Delta_k + t_k)] \leq \int_{t_k}^{t} X(u)\, du \tag{4.8}$$

Rearranging terms in the above equation gives the condition in the theorem.

■

Definition 4.1

Let $\mathcal{L}(\Delta_0)$ be the set of all possible transmission policies $\boldsymbol{r}_s$ that satisfy Theorem 4.1:

$$\mathcal{L}(\Delta_0) := \{\boldsymbol{r}_s \in [r_b, r_b + r_e]^{N_{slot}} : \ \textit{no loss in } [0, T]\} \tag{4.9}$$

4.4.2 Maximizing Bandwidth Efficiency

In this subsection we consider the problem of maximizing bandwidth efficiency over all policies that give no loss. When the no loss condition in Theorem 4.1 holds, then the overall efficiency E given in (4.4) is maximized if and only if $\int_0^{t_{end}} X(t)\, dt$ is maximized, which is equivalent to maximizing t_{end}.

Let $t_{end}(\boldsymbol{r}_s) \leq T$ be the time at which the server finishes streaming under transmission policy $\boldsymbol{r}_s \in \mathcal{L}(\Delta_0)$. For a fixed value of Δ_0, we can define the maximum ending time of streaming under all transmission policies in $\mathcal{L}(\Delta_0)$, as:

$$t_{end}^{max} = \max_{\boldsymbol{r}_s \in \mathcal{L}(\Delta_0)} t_{end}(\boldsymbol{r}_s) \tag{4.10}$$

We can observe that t_{end}^{max} only depends on Δ_0 and $X(t)$ for $t \in [0, T]$.

Now, let E^* be the maximum value of E that can be attained by a policy $\boldsymbol{r}_s \in \mathcal{L}(\Delta_0)$, and let $\mathcal{E}(\Delta_0)$ be the set of policies $\boldsymbol{r}_s \in \mathcal{L}(\Delta_0)$ that attain E^*. Since maximizing E is equivalent to maximizing t_{end}, we have:

Theorem 4.2

The set of transmission policies that maximizes E satisfies:

$$\mathcal{E}(\Delta_0) = \{\boldsymbol{r}_s \in \mathcal{L}(\Delta_0) : \ t_{end}(\boldsymbol{r}_s) = t_{end}^{max}\} \tag{4.11}$$

In particular, the maximum value of E is given by:

$$E^* = \frac{\Delta_0}{T} + \frac{\int_0^{t_{end}^{max}} X(t)\, dt}{T \cdot (r_b + r_e)} \tag{4.12}$$

This simply states that, in order to maximize bandwidth efficiency, the streaming application should try to exploit the transmission channel as long as possible. A transmission policy which is not sufficiently aggressive in its choice of $r_s(k)$ may have streamed all the video frames long before the end of rendering at the client ($t_{end}(\boldsymbol{r}_s) < t_{end}^{max} \leq T$), and thereby not use the additional bandwidth that is available until the end of rendering.

Let $\bar{X} = \frac{1}{T}\int_0^T X(t)\, dt$ be the average available bandwidth during the playback interval $[0, T]$. A special case to consider is when $t_{end}^{max} = T$, i.e., when there exists transmission policies in $\mathcal{L}(\Delta_0)$ that can maintain the streaming until the end of the rendering at the client. In this case, we have: $E^* = \frac{\Delta_0}{T} + \frac{\bar{X}}{r_b + r_e}$. Otherwise, when $t_{end}^{max} < T$, we have: $E^* < \frac{\Delta_0}{T} + \frac{\bar{X}}{r_b + r_e}$.

4.4.3 Minimizing Rate Variability

From all the transmission policies $\boldsymbol{r}_s$ that yield no loss of data and maximize bandwidth efficiency E, i.e. $\boldsymbol{r}_s \in \mathcal{E}(\Delta_0)$, we now look for those which minimize the rate variability V, given the available bandwidth $X(t)$. We show that this problem can be solved by finding the shortest path in a graph. Let V^* be the minimum value of V that can be attained by a policy $\boldsymbol{r}_s \in \mathcal{E}(\Delta_0)$.

Definition 4.2

We define the optimal state graph $\mathcal{G}$ of our system as the graph represented in Figure 4.3, whose nodes represent, at each time t_k, all the possible sampled values for the playback delay at time t_k, i.e., Δ_k. The arcs represent the evolution of the playback delay after the streaming of the video from time t_k to time t_{k+1}, such that there is no loss of video data over the entire duration of the streaming. The initial state of the graph represents the initial playback delay of $\Delta(0) = \Delta_0$ seconds of video data, present in the client playback buffers at time $t = 0$. The final state is reached at the time when the server shall finish streaming its video data in order to maximize the bandwidth efficiency E, i.e., at time $t = t_{end}^{max} \in [t_{N_{last}}, t_{N_{last}+1})$.

Theorem 4.3

The problem of finding an optimal transmission policy $\boldsymbol{r}_s^* \in \mathcal{E}(\Delta_0)$ which minimizes the variability V can be solved by finding the shortest path in the system state graph $\mathcal{G}$.

Proof. From the definition of the optimal state graph $\mathcal{G}$, the no loss condition expressed in Theorem 4.1 is satisfied. Considering the streaming of the video by the server between times t_k and t_{k+1}, we can easily

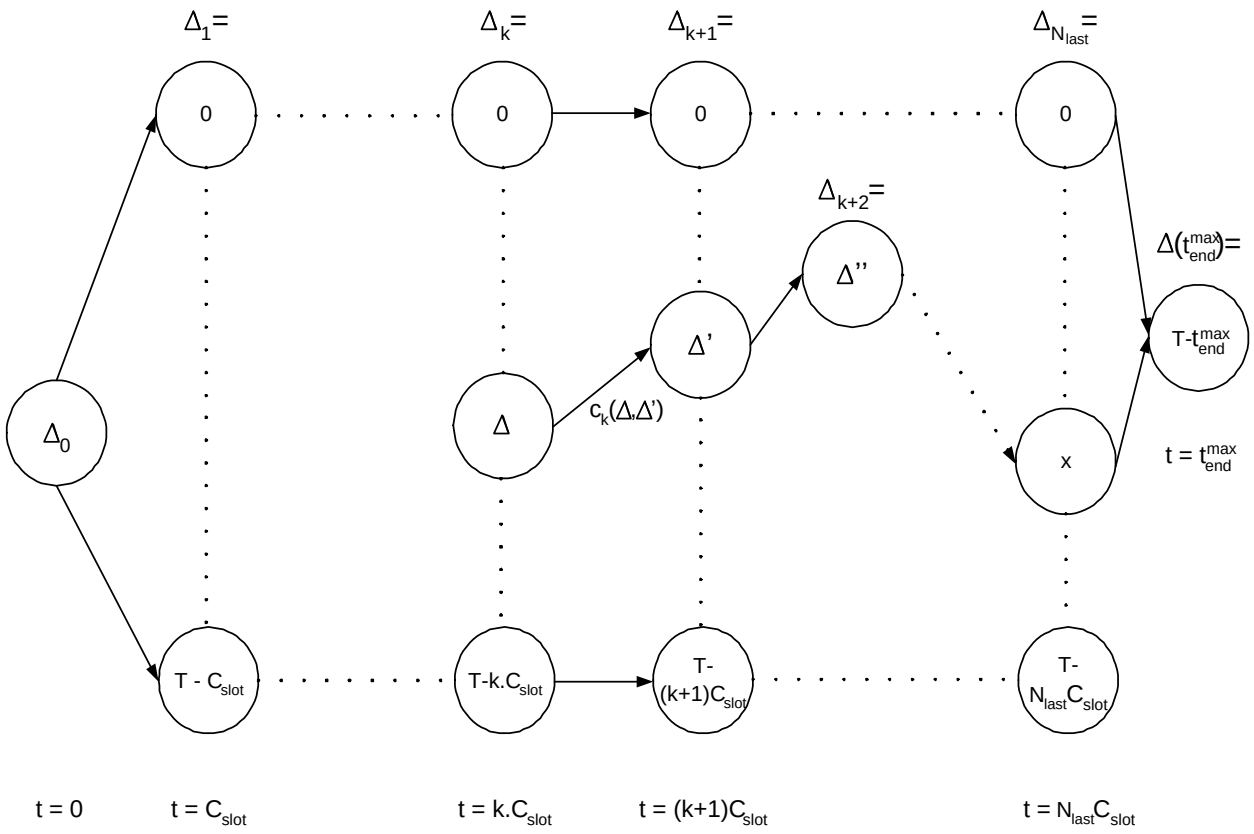


Figure 4.3: Optimal state graph $\mathcal{G}$

show that:

$$\forall k \in \{0, \ldots, N_{last} - 1\}, \Delta_{k+1} = \Delta_k + \frac{\int_{t_k}^{t_{k+1}} X(t)dt}{r_s(k)} - C_{slot} \tag{4.13}$$

This means that the transition from one state at time t_k to the next possible state at time t_{k+1} is completely determined by the choice of $r_s(k)$. Therefore, all the possible paths between the initial state to the final state give all the possible transmission policies $\boldsymbol{r}_s \in \mathcal{L}(\Delta_0)$.

The final state of the graph ensures that all these transmission policies satisfy $t_{end}(\boldsymbol{r}_s) = t_{end}^{max}$, thus ensuring by Theorem 4.2 that the maximum bandwidth efficiency is reached.

The cost of an arc from state $\Delta_k = \Delta$ to $\Delta_{k+1} = \Delta'$ is denoted $c_k(\Delta, \Delta')$, as shown on Figure 4.3. This cost is obtained recursively during the computation of the shortest path from the initial state to the final state, obtained by dynamic programming. It is defined as $(r - r')^2$, where r is the unique value of $r_s(k)$ that makes the system transition from state $\Delta_k = \Delta$ to $\Delta_{k+1} = \Delta'$, and r' the value of $r_s(k+1)$ that makes the system transition from state $\Delta_{k+1} = \Delta'$ to the next state in the shortest path

from $\Delta_{k+1} = \Delta'$ to the final state. This way, the shortest path from the initial state to the final state yields $\boldsymbol{r}_s^*$ which minimizes the measure of variability V, as defined in (4.5).

■

Given Δ_0 and $X(t)$, this theorem assumes the knowledge of the value of t_{end}^{max}. Let's assume that $\mathcal{E}(\Delta_0) \neq \emptyset$ (it also means that $\mathcal{L}(\Delta_0) \neq \emptyset$, i.e., we can at least stream all the base layer without loss). In this case, t_{end}^{max} is defined in $(0, T]$. We first set its value to T and decrease it recursively until we find a shortest path in the optimal state graph $\mathcal{G}$. Indeed, if for a given possible value of t_{end}^{max} there exists no possible path in $\mathcal{G}$ from the initial state to the final state, it means that $\mathcal{E}(\Delta_0) = \emptyset$, which contradicts our hypothesis.

Given Δ_0 and $X(t)$, we have implemented the algorithm for finding the shortest path in graph $\mathcal{G}$ as well as t_{end}^{max}, yielding the minimum variability V^*. Note that the actual number of nodes in the graph depends on the size of $\mathcal{L}(\Delta_0)$, the value of t_{end}^{max}, and the sampling precision of the buffering delays Δ_k.

4.5 Real–time Rate Adaptation Algorithm

Henceforth, we no longer assume that the available bandwidth $X(t)$ is known a priori for the whole duration of the streaming. Motivated by the theory of section 4.4, we provide a heuristic real–time policy that adapts on–the–fly to the variations of $X(t)$. The theory of the previous section also provides a useful bound to which we can compare the performance of our heuristic.

4.5.1 Description of the Algorithm

Algorithm 4.1 presents our real–time heuristic. At the beginning of each time slot of length C_{slot} seconds, i.e., at each time t_k, the server fixes the encoding rate for the slot, i.e., it fixes $r_s(k)$. Recall that, in our model, the server knows the number of seconds of video data contained in the client playback buffers, Δ_k, at each time t_k. The server can compute the average goodput as seen in the previous slot, between time t_{k-1} and t_k, denoted by $X_{avg}(k-1)$. If there is no loss of data in the previous time slot, then $X_{avg}(k-1)$ is expressed as:

$$X_{avg}(k-1) = r_s(k-1) \cdot \frac{\Delta_k - \Delta_{k-1} + C_{slot}}{C_{slot}}. \tag{4.14}$$

This is obtained by rearranging the terms in (4.13).

At the beginning of each slot k the algorithm operates according to the value of Δ_k:

– When $\Delta_k \leq C_{slot}$, there is potential loss in the upcoming slot; because minimizing base layer loss is our most important objective, we set $r_s(k) = r_b$, that is, we send only the base layer during

for $k = 0$ *to* $N_{slot} - 1$ **do**

Retrieve/estimate the value of Δ_k from the client

Compute $X_{avg}(k-1)$

Compute the value of $r_s(k)$*:*

if $\Delta_k \leq C_{slot}$ **then**

$r_s(k) = r_b$

else if $C_{slot} < \Delta_k \leq 2 \cdot C_{slot}$ **then**

$r_s(k) = \alpha \cdot X_{avg}(k-1) + (1-\alpha) \cdot r_s(k-1)$

else if $\Delta_k \geq 2 \cdot C_{slot}$ **then**

$r_s(k) = \alpha \cdot X_{avg}(k-1) \cdot \frac{\Delta_k}{2 \cdot C_{slot}} + (1-\alpha) \cdot r_s(k-1)$

Make sure $r_s(k)$ *stays within bounds:*

if $r_s(k) < r_b$ **then** $r_s(k) = r_b$

if $r_s(k) > r_b + r_e$ **then** $r_s(k) = r_b + r_e$

end

Algorithm 4.1: Real–time heuristic

the slot. The no loss condition as expressed in Theorem 4.1, i.e., $r_s(k) \leq \beta_k(\Delta_k)$, would give a less conservative choice for $r_s(k)$, but $\beta_k(\Delta_k)$ strongly depends on the variations of the available bandwidth in the next C_{slot} seconds, which is very difficult to predict. Additionally, this choice attempts to maintain a minimum of C_{slot} seconds of data in the client buffer, which should be sufficient to mitigate jitter and allow for retransmission of lost packets.

– When $C_{slot} < \Delta_k \leq 2 \cdot C_{slot}$, the server can start increasing the value of $r_s(k)$. In order to maintain a high bandwidth efficiency E, we know from Theorem 4.2 that we must make the streaming last as long as possible, i.e., we need to maximize t_{end}. Therefore, we use a video encoding rate that tracks the average available bandwidth of the connection. However, the values of the averages $X_{avg}(k)$ may have large fluctuations. So, we include a smoothing factor $\alpha \in (0,1)$, which aims to smooth the variations of $X_{avg}(k)$. By choosing $r_s(k) = \alpha \cdot X_{avg}(k-1) + (1-\alpha) \cdot r_s(k-1)$, we try to minimize the differences in consecutive values of $r_s(k)$, while getting close to a smoothed average of the available bandwidth. The value of α can be chosen to trade off small quality variability (small α) with better overall bandwidth utilization (high α).

– When $\Delta_k > 2 \cdot C_{slot}$, our heuristic is more aggressive with respect to the available bandwidth (by a factor of $\frac{\Delta_k}{2 \cdot C_{slot}}$). The heuristic increases the value of $r_s(k)$ proportionally to the amount of data stored in the playback buffers. This depletes the client buffers, thus preventing the streaming from

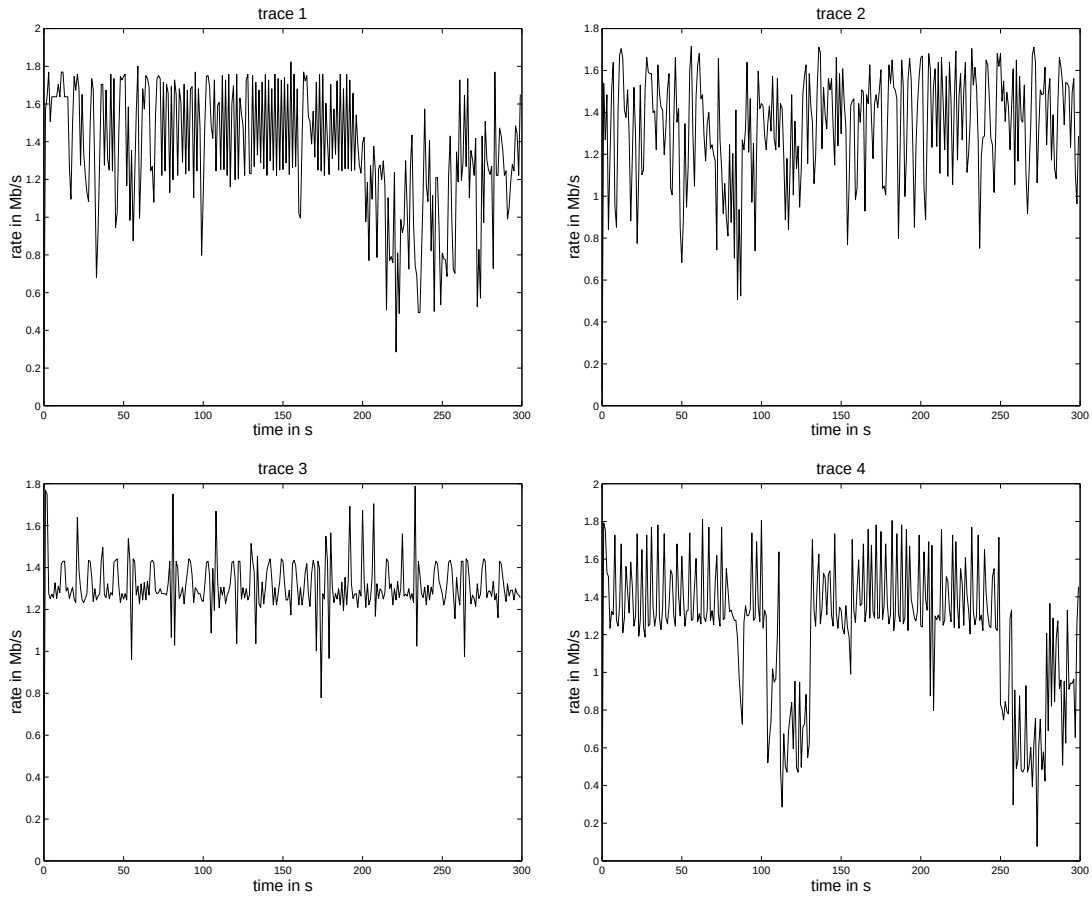


Figure 4.4: 1 second average goodput of the collected TCP traces

ending too early.

Finally, we make sure that the computed value of $r_s(k)$ stays within $[r_b, r_b + r_e]$.

4.5.2 Simulations from Internet Traces

We made simulations from real Internet TCP traces. We used `snoop` on Solaris to collect goodput from 5mn–long TCP connections at different times of the day between University of Pennsylvania in Philadelphia, and Institut Eurecom in France. The simulations presented here are made with a time slot length of $C_{slot} = 5$ s, and a video length of $T = 300$ s. The client playback buffers collect $\Delta_0 = 6$s of video data before the client starts decoding and rendering. We used the four traces whose 1 second averages are depicted in Figure 4.4.

Figure 4.5 shows the results of a simulation, for which we used TCP trace 1 with average available bandwidth $\bar{X}_1 = 1.35$ Mbps. The coding rates of both layers are set to $r_b = r_e = 0.75 \cdot \bar{X}_1 = 1$

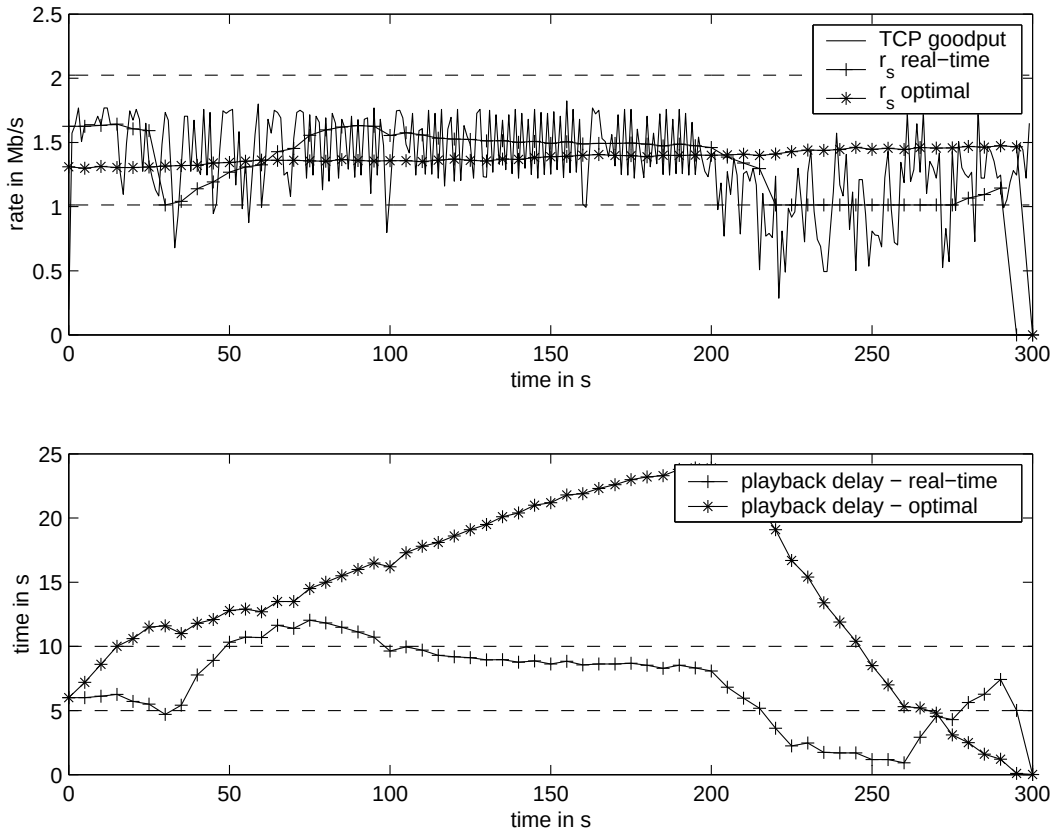


Figure 4.5: Rate adaptation for trace 1

Mbps, so that the total coding rate of the video is strictly superior to the average available bandwidth ($r_b + r_e = 1.5$ Mbps). The smoothing factor is set to $\alpha = 0.2$. The top plot shows the transmitted video encoding rate of our real–time heuristic, the encoding rate of the optimal transmission policy (given by Theorem 4.3) and the connection goodput. The plot below shows, at each time t_k, the amount of video data in seconds in the client playback buffers, Δ_k, for both the real–time and the optimal transmission policies. We observe that the optimal policy stores up to 24 seconds of video data into the client buffers during playback to smooth bandwidth variations and achieve the maximum bandwidth efficiency. The real–time transmission policy stores up to 12 seconds during playback to smooth bandwidth fluctuations, although it does not smooth as well as the optimal transmission policy. The top plot shows that the real-time algorithm adapts well to the varying available bandwidth. In particular, after time $t = 200$ms the streamed video coding rate drops to its minimum value, r_b, because the playback delay drops below $C_{slot} = 5$ seconds, as shown on the bottom plot. The real–time transmission policy provides a bandwidth efficiency E that is close to the maximum ($E = .63$ compared to $E^* = .68$). Indeed, the real–time policy nearly satisfies Theorem 4.2: it does not stop transmitting video frames until just one time slot before the

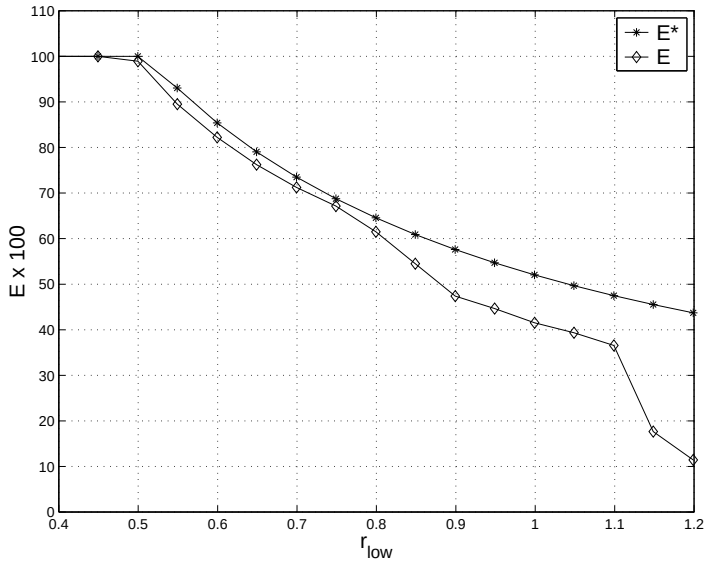


Figure 4.6: Bandwidth efficiency as a function of the normalized base layer rate

last one ($r_s(295) = 0$ on the graph).

In order to study the performance of our real–time algorithm in various bandwidth situations with respect to the video coding rate, we define, similarly to r_{high} in Chapter 3, the normalized base layer coding rate $r_{low} = \frac{r_b}{\bar{X}}$. We distinguish between the following cases:

- $r_{low} = 1$ corresponds to the situation when the average available bandwidth is sufficient to stream only the base layer,
- $r_{low} < 1$ means that the average available bandwidth exceeds the average base layer rate, which should allow for the streaming of a part of the enhancement layer (this is the target situation for our study),
- $r_{low} > 1$ corresponds to the unfavorable situation when the average available bandwidth is not sufficient to stream the base layer without any interruption.

For simplicity, we assume $r_b = r_e$. Figures 4.6 and 4.7 show the evolution of the measures E and V as a function of r_{low} for the same trace as in Figure 4.5 and for the same parameter values. Figure 4.6 depicts the variations of E compared to the maximum achievable bandwidth efficiency E^* given in (4.12). We see that the real–time heuristic's overall bandwidth efficiency is close to the maximum for r_{low} between 0.4 and 0.8, corresponding to the favorable situation when the average available bandwidth is high enough to sustain the streaming of the base layer and a part of the enhancement layer without loss. When $r_{low} > 0.8$, the average available bandwidth is closer to the base layer coding rate, resulting in

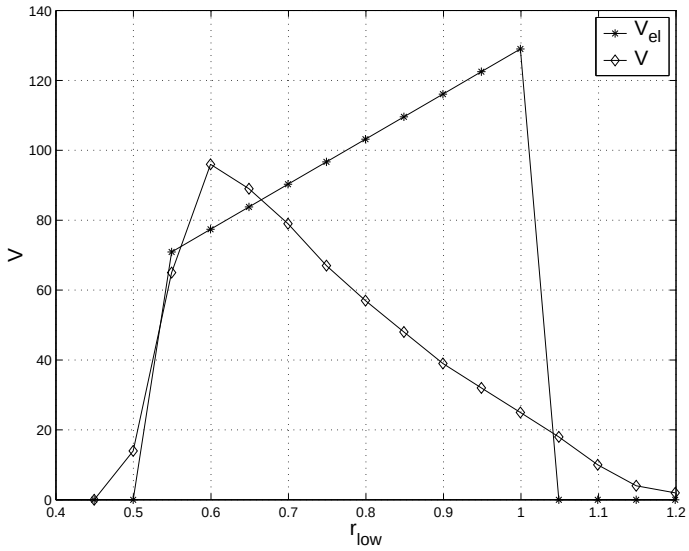


Figure 4.7: Rate variability as a function of the normalized base layer rate

some loss of base layer data, which increases the difference between the achieved bandwidth efficiency and the maximum.

Figure 4.7 shows the evolution of the variability V given by our adaptation heuristic. It also shows the value of V, denoted by V_{el}, obtained with the streaming policy which adds or removes the entire enhancement layer just once for the whole duration of the streaming. We did not plot the minimum variability obtained by the optimal allocation because it is always very close to zero, and thus insignificant. When we consider the variability of our real–time algorithm for FGS video, we notice from Figure 4.7 that V reaches a maximum at $r_{low} = 0.6$, then decreases as r_{low} increases. Indeed, as r_{low} increases, the average available bandwidth becomes lower than $r_b + r_e$, resulting in fewer opportunities to stream a high bit–rate video, i.e., to choose high values for $r_s(k)$. We also see that, in most bandwidth conditions, the total variability obtained by our heuristic is lower or roughly equal to V_{el}, which gives an indication of the low level of rate variations achieved by our algorithm.

In Table 4.2, we give the results in terms of E, V and losses of video data for the three other traces in Figure 4.4, and for different values of r_{low}. These results confirm the good performance of our real–time algorithm in terms of high bandwidth efficiency and low variability for a variety of bandwidth scenarios.

4.6 Streaming over TCP–Friendly Algorithms

In order to reduce image quality variation, a number of studies advocate the use of smooth–rate TCP–friendly algorithms for streaming [41, 106, 121]. In this section we show that, for stored FGS–encoded video, streaming over a highly varying TCP–friendly algorithm, such as the ordinary TCP, can achieve

	Trace 2 $\bar{X} = 1.34$ Mbps			Trace 3 $\bar{X} = 1.31$ Mbps			Trace 4 $\bar{X} = 1.40$ Mbps		
	E/E^*	V/V_{el}	losses	E/E^*	V/V_{el}	losses	E/E^*	V/V_{el}	losses
$r_{low} = 0.6$	.83/.85	52/116	0	.84/.85	68/116	0	.83/.85	106/116	0
$r_{low} = 0.75$	.66/.69	33/97	0	.67/.69	46/97	0	.63/.69	86/97	0.7s
$r_{low} = 0.9$	.56/.58	19/77	0	.56/.58	25/77	0	.52/.58	50/77	1.1s

Table 4.2: Simulations from Internet traces

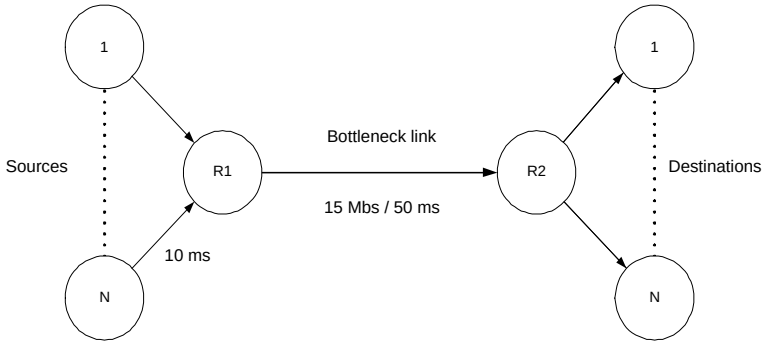


Figure 4.8: Network configuration

essentially the same low level of quality variability that can be achieved with a smooth–rate TCP–friendly algorithm. We achieve this low–level of variability by combining playback buffering and coarse–grained rate adaptation, as provided by our real–time heuristic of Section 4.5.

Our real–time adaptation algorithm can be run on top of TCP or on top of a TCP–friendly algorithm such as TFRC (TCP-Friendly Rate Control), defined in [41]. We used `ns` to collect TCP and TFRC traces under the same network conditions. The network topology is the commonly employed "single bottleneck", as shown in Figure 4.8, for which congestion only occurs in the link between routers R1 and R2 (we used access links with delay 10 ms and a 15 Mbps bottleneck link with delay 50 ms).

If the playback delay between the server and the client, $\Delta(t)$, is maintained at an order of a few seconds, the server has the ability to retransmit lost packets. Therefore we assume that our TFRC connection can be made fully reliable. We use a RED bottleneck queue to avoid global TCP synchronization[2]. From the collected traces, we run the real–time adaptation algorithm for different values of r_{low} and α, and compare the results in terms of measures E and V. We define the network $load$ as the number of simulta-

[2]TCP connections with the same RTT transmitted over a Drop–Tail queue adjust their congestion window in synchrony, which results in an underutilization of congested links and unfairness between the TCP streams; using RED queues makes the global synchronization of TCP less pronounced [39, 98].

load	V^*_{tfrc}	V^*_{tcp}	V_{tfrc}	V_{tcp}	V_{el}	α_{tfrc}	α_{tcp}
5	6	6	42	74	97	0.15	0.01
10	6	10	61	55	97	0.01	0.01
15	5	8	42	43	97	0.07	0.04
20	7	12	8	85	97	0.02	0.02
25	6	8	49	93	97	0.15	0.01
30	6	9	50	23	97	0.15	0.01
35	6	6	49	82	97	0.05	0.2
40	6	11	83	116	97	0.01	0.04

Table 4.3: Performance as a function of network load

neous TCP and TFRC connections inside the bottleneck, apart from the connection which is monitored. For example, a value of $load = 10$ means that the bottleneck link is shared by 10 TCP connections, 10 TFRC connections, and by the TCP or TFRC connection which is monitored. In our simulations, we vary the value of $load$ between 5 and 40.

The first two columns of Table 4.3 show, as a function of the network load, the minimum variability V^*_{tfrc} and V^*_{tcp} achieved by the optimal transmission policy, for the monitored TFRC or TCP connection, respectively. We consider a base layer normalized coding rate of $r_{low} = 0.75$,

As we can see, the minimum variability when using TCP is only slightly higher than the minimum variability that can be achieved when using TFRC, even though the TFRC long–term throughput is considerably smoother than TCP, especially at low loss rates (low $load$ value) [41,145]. We also observe that for both cases V^* remains low for all network loads, which indicates that our application–layer smoothing approach has the potential to work well in a wide range of network conditions.

We then applied our real–time algorithm to both the TCP and TFRC traces. For a given network load, we varied the smoothing parameter α between 0.01 and 0.95, which gives different values for the couple (E, V). Then, among the choices of α that bring E to within 1% of the maximum, we keep the α that minimizes the variability V. The last columns of Table 4.3 give the results for V_{tfrc} and V_{tcp}, along with the variability obtained with a transmission policy that would add the full enhancement layer just once, denoted by V_{el}. The last two columns show the corresponding value of α, denoted by α_{tfrc} and α_{tcp}, respectively. We first observe that V_{tfrc} is, for most network loads, less than V_{tcp}. However, both values remain low (usually less than V_{el}); the difference may probably not be noticed by the user. We also observe that for a network load with less than 30 competing TCP and TFRC connections, the smoothing parameters that minimizes V while insuring a high E satisfy $\alpha_{tfrc} \geq \alpha_{tcp}$. Indeed, because TFRC has smoother rate variations than TCP in a low loss environment, the application needs to smooth

bandwidth variations of TCP more than TFRC.

Finally, Figure 4.9 and Figure 4.10 show the rate adaptation provided by the optimal policy and the real–time algorithm for TCP and TFRC connections, respectively. The bottleneck link is shared by 25 long–lived TCP and 25 long–lived TFRC connections, and the base layer normalized encoding rate is set to $r_{low} = 0.75$. In both cases, we use the optimal value of the smoothing parameter α in our real-time algorithm (as given in Table 4.3) for a network load of 25, i.e., we use $\alpha_{tfrc} = 0.15$ and $\alpha_{tcp} = 0.01$. As in Figure 4.5, the top plot in both figures shows the coding rate of our real–time heuristic, the coding rate of the optimal transmission policy, and the connection goodput. The bottom plot shows, at each time t_k, the amount of data in seconds in the client playback buffers, Δ_k, for both the real–time and the optimal transmission policies.

We first compare the real–time transmission policies in both figures. We see that the real–time rate adaptation algorithm yields very smooth variations in the transmitted video coding rate for both TCP and TFRC, which is consistent with the low values obtained for V_{tfrc} and V_{tcp} in Table 4.3 when $load = 25$. Furthermore, the real–time algorithm sustains the duration of the streaming almost until the end of the rendering in both cases, ensuring a high overall bandwidth efficiency. When comparing the playback delays, we see however that the real–time algorithm needs to store up to 30 seconds of video into the client buffers to smooth variations of TCP, while it needs to store only 13 seconds of video in the case of TFRC. When comparing the optimal transmission policies in both figures, we see that, in both cases, the variability attained is negligible and the optimal transmission policies require up to 10 seconds of video data to be stored into client buffers.

4.7 Implementation of our Framework

In this section, we present an implementation of our framework and real–time heuristic for streaming stored FGS video. The implementation consists of a video streaming server and a client with an MPEG–4 FGS decoder[3]. Our implementation runs over TCP. As we mentioned in Section 2.1.2, we believe that TCP is a viable choice for video streaming. TCP is still ubiquitous in the Internet, it has been proven to be stable and is available to use. Also, as we have demonstrated in Section 4.6, combining adaptive rate–control for FGS–encoded videos with sufficient playback buffering can accommodate bandwidth fluctuations of TCP with very good performance. Nevertheless, the full–reliability of TCP is not necessary for video streaming applications that can benefit from partially–reliable schemes such as selective retransmissions or FEC. Therefore, our system has been designed to be potentially run over any partially reliable TCP–friendly RTP/UDP connection.

[3]This system was implemented through the French national project VISI, together with Thomson Multimedia, Irisa, Inria, Edixia and France Telecom R&D.

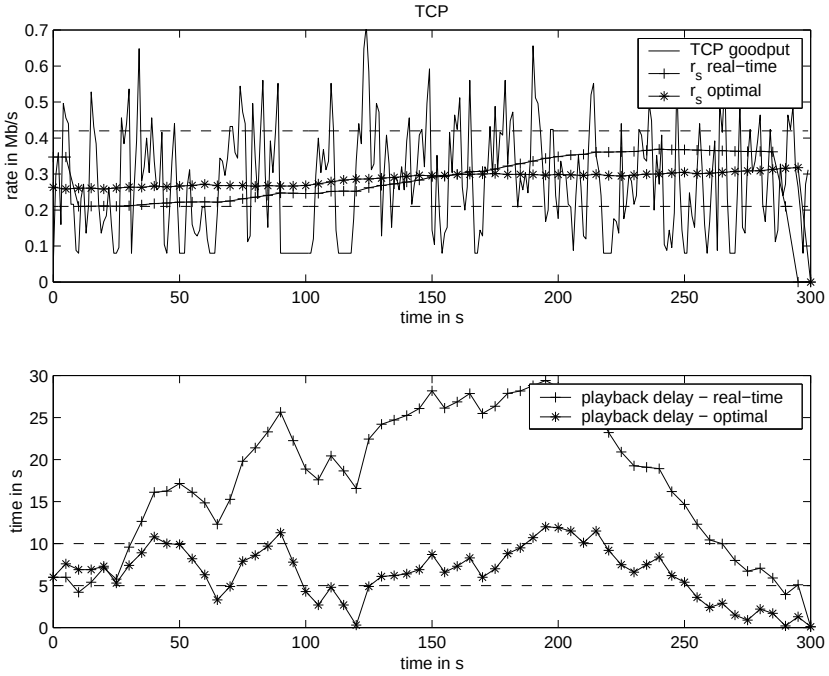


Figure 4.9: Rate adaptation for TCP

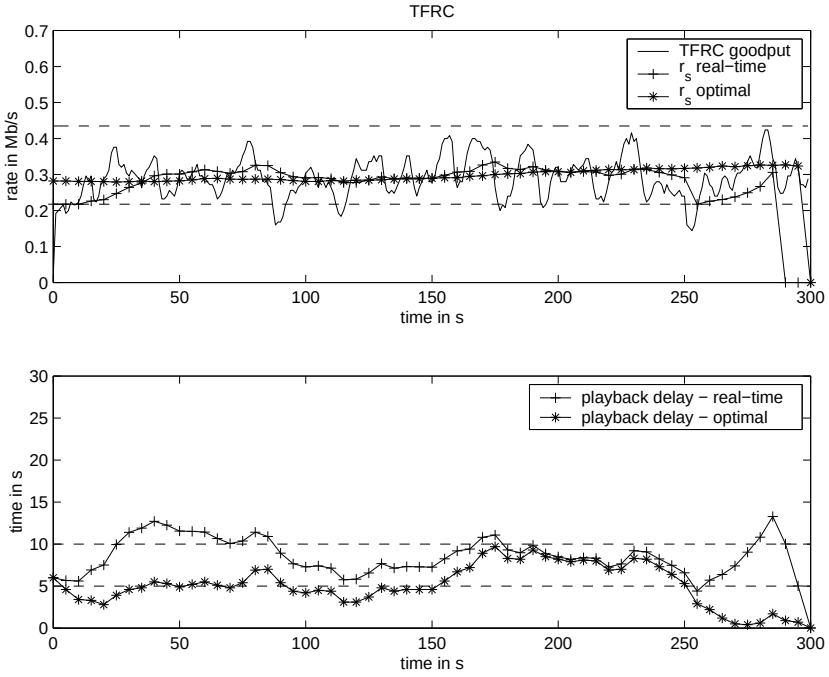


Figure 4.10: Rate adaptation for TFRC

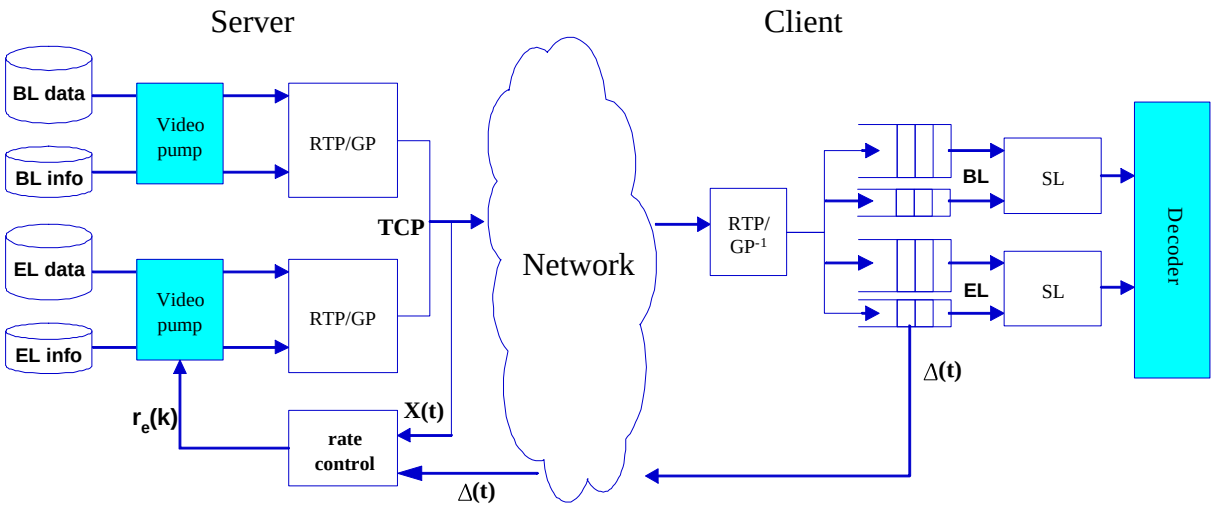


Figure 4.11: Architecture of the MPEG–4 streaming system

4.7.1 Architecture

The end–to–end architecture of our system is depicted in Figure 4.11. We use the terminology of the MPEG–4 system, as introduced in Section 2.2.2.

At the server, base layer and enhancement layer data are stored in separate files. Each data file is associated with a meta–file containing pointers to the individual Access Units (AUs), or video frames, as well as their composition and decoding timestamp. Video pumps push the AUs and their associated information to the network module. The server network module encapsulates the AUs into RTP packets in order to stay compatible with implementations over RTP/UDP[4]. The RTP payload format we used is the Group Payload (GP) format defined in [48]. Both BL and EL RTP packets are multiplexed over the same TCP connection to the client. Our server streams the video data at the maximum TCP available bandwidth $X(t)$ at time t. As in our framework, the server is required to send both layers of the same frame together.

The client extracts the AUs and their associated information from the incoming RTP packets and sends them into the corresponding playback buffers. In our implementation we used four playback buffers at the client: two for the base layer and enhancement layer data, and two for the base layer and enhancement layer meta–information. As we mentioned in Section 4.2, because the server streams both layers of the same frame together, BL and EL buffers always contain the same number of frames. The client sends back periodically to the server the value of the playback delay at time t, $\Delta(t)$. Individual

[4]Streaming RTP over TCP brings some overhead, although relatively small because AUs do not need to be fragmented into several RTP packets in the case of transmission over TCP.

for $k = 0$ *to* $N_{seg} - 1$ **do**

Retrieve/estimate the value of Δ_k from the client

Compute $X_{avg}(k-1)$

Compute the value of $r_e(k)$*:*

if $\Delta_k \leq C$ **then**

$r_e(k) = 0$

else if $C < \Delta_k \leq 2 \cdot C$ **then**

$r_e(k) = \alpha \cdot (X_{avg}(k-1) - r_b(k)) + (1-\alpha) \cdot r_e(k-1)$

else if $\Delta_k \geq 2 \cdot C$ **then**

$r_e(k) = \alpha \cdot r_e \cdot \frac{\Delta_k}{2 \cdot C} + (1-\alpha) \cdot r_e(k-1)$

Make sure $r_e(k)$ *stays within bounds:*

if $r_e(k) < 0$ **then** $r_e(k) = 0$

if $r_e(k) > r_e$ **then** $r_e(k) = r_e$

end

Algorithm 4.2: Implemented heuristic

AUs and their meta–information are then given to the decoder as SL packets, according to the MPEG–4 specification [6].

We study streaming of long videos (from tens of seconds to hours) that are composed of several segments. We assume that the stored video has been partitioned into N_{seg} video segments. In our experiments, the video has been segmented by hand so that all images within the same video segment have similar visual characteristics (e.g., same video shot, same motion, or same image complexity).

Unlike in the previous sections, we suppose that the base layer is VBR–encoded, with coding rate $r_b(t)$; the FGS enhancement layer is still CBR–encoded with coding rate r_e. According to the fine granularity property, the server can cut the enhancement layer bitstream anywhere. In our implementation, the rate control module fixes the coding rate of the enhancement layer to stream to the client for a given video segment k. This is denoted by $r_e(k) \in [0, r_e]$ (recall that in our previous framework $r_e(k) = K_s(k) \cdot r_e$). The video pump cuts the enhancement layer bitstream according to this rate. Note that, in this implementation, slot $[t_k, t_{k+1})$ now corresponds to the streaming of video segment k and, unlike in our previous simplified model, the successive slots can now have different length.

4.7.2 Simulations

We adapted the heuristic of Algorithm 4.1 to the case when the base layer is VBR–encoded and the server slots have variable length. The new heuristic is given in Algorithm 4.2. The constant C, which corresponds to the length of a server slot C_{slot} in the previous heuristic, is chosen empirically to trade off video quality with playout interactivity. It is expressed in seconds–worth of video data. We denote by $r_b(k)$ the average base layer coding rate of video segment number k (this information can be stored at the server because the video is pre–encoded).

We suppose that the client starts the playout of video data after having received $\Delta_0 = C$ seconds of data. As in Algorithm 4.1, the smoothing factor α aims to smooth the variations of $X_{avg}(k)$, so that the coding rate of successive video segments varies slowly (in order to minimize variations in quality between successive video segments). The value of α is chosen empirically to trade off small quality variability (small α) with better overall bandwidth utilization (high α). Similarly to Algorithm 4.1, when $\Delta_k > 2C$, the coding rate of the enhancement layer to send is proportional to the time–worth of video data stored in the playback buffers, Δ_k, in order to keep the playback buffers small.

For our experiments, we used a 4–mn video including both high and low motion scenes. We segmented the video into 54 segments of various length (in our tests, each segment actually corresponds to a short scene shot). The base layer was VBR–encoded, with average encoding rate of 384 kbps. The enhancement layer was coded at $r_e = 616$ kbps. The server and client were located at each end of a dedicated LAN, and a Linux router was used to limit the end–to–end available bandwidth: cross traffic was generated from the server to the client in order to make the available bandwidth for the streaming application vary with time.

Figure 4.12 shows the available bandwidth $X(t)$ given to the streaming application, as well as the choices made by the server for the enhancement layer coding rate $r_e(k)$ using our real–time heuristic with $C = 5$ s and $\alpha = 0.5$. We see that the variations of the enhancement layer coding rate streamed to the client roughly follow the variations of the available bandwidth[5]. Figure 4.13 depicts the variations of the content of the client playback buffers, i.e., $\Delta(t)$. We see that it never empties, i.e., no video data was lost. The application has stored up to 27 s of video into the client playback buffers. Finally, Figure 4.14 shows the image quality in PSNR after decoding video frames 525 to 1200 with our shot–based video segmentation (segm), together with the image quality obtained when the video segments are of arbitrary length of 200 frames (no segm). The dotted vertical lines show the boundaries of the video shots. We see that without shot–based segmentation the image quality can abruptly change within a same scene shot (e.g., around frame number 1000), which degrades the perceived quality (the difference in quality is around 1.1 dB in this example).

[5]Note that the coding rate of the VBR–encoded base layer should be added to the enhancement layer coding rate to make the total coding rate of the video.

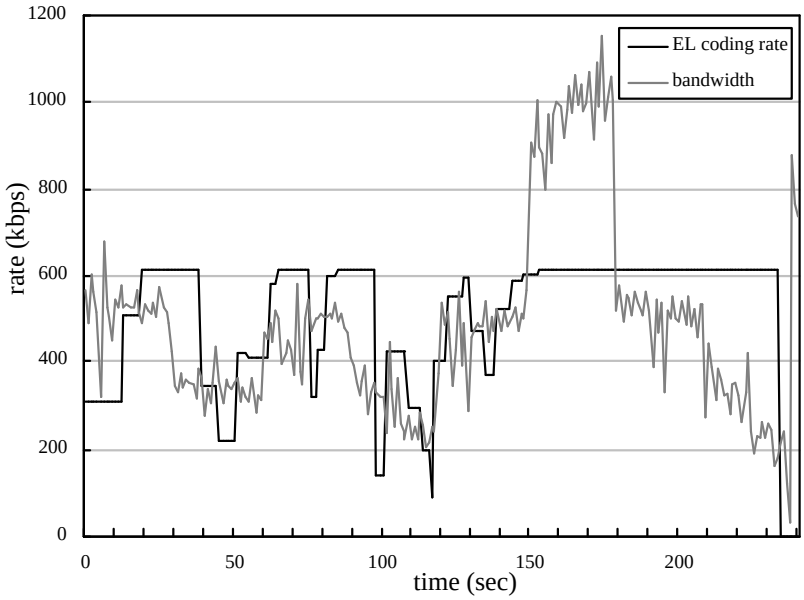


Figure 4.12: Evolution of the enhancement layer coding rate and available bandwidth

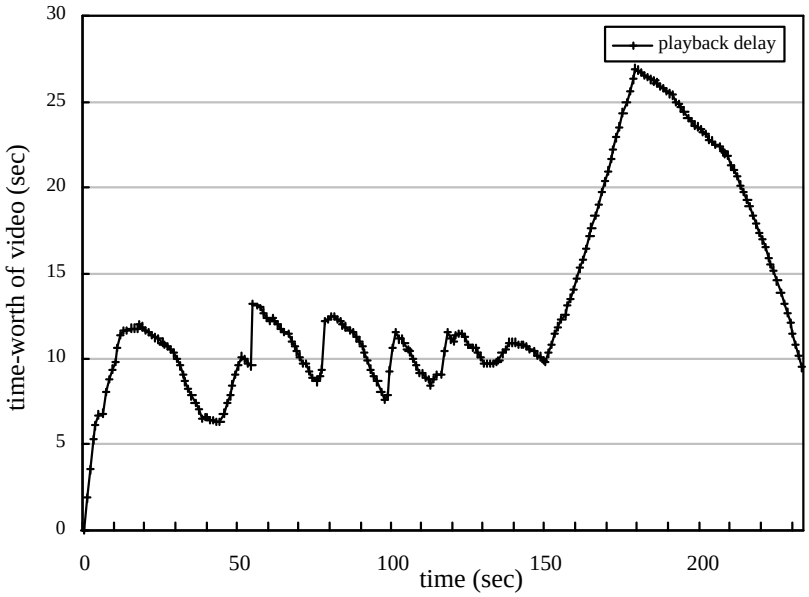


Figure 4.13: Evolution of the playback delay

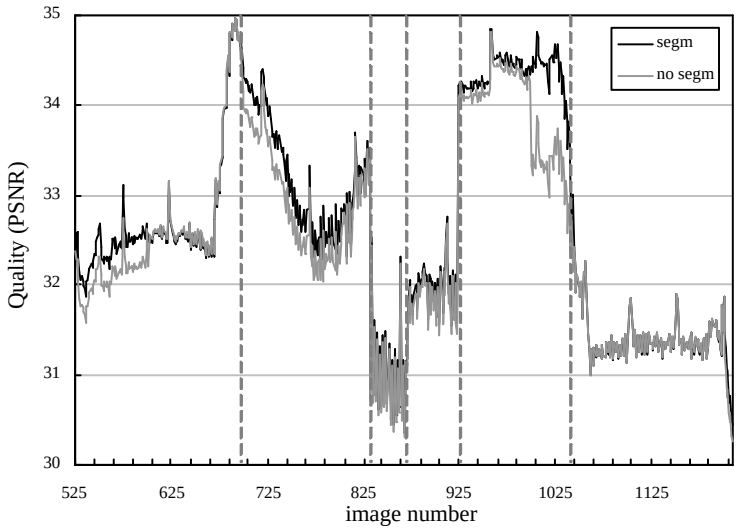


Figure 4.14: Quality in PSNR for images 525 to 1200

C (s)	Δ_{end} (s)	V_{seg} (dB)
1	0	2.40
5	8.8	1.95
15	45.7	1.92

Table 4.4: Performance for varying C

α	Δ_{end} (s)	V_{seg}
0.1	11.5	1.90
0.5	8.8	1.95
1.0	8.95	2.05

Table 4.5: Performance for varying α

We have run the same experiment with different parameters for our real–time adaptation heuristic. We denote by Δ_{end} the content of the client playback buffers when the server has finished to stream all AUs. Lower values of Δ_{end} result in higher total bit consumption, or bandwidth efficiency, and then potentially higher overall video quality, because this minimizes the bandwidth left unused at the end of the streaming. Similarly to V in (4.5), we define a metric that can account for the average variations in quality between successive video segments:

$$V_{seg} = \frac{1}{N_{seg} - 1} \sum_{k=2}^{N_{seg}} |PSNR(k) - PSNR(k-1)|, \tag{4.15}$$

where $PSNR(k)$ is the average PSNR of video segment number k (we give more details about MSE and PSNR quality measures in Chapter 5; this measure is similar to the $V(r)$ measure defined in Chapter 5). V_{seg} is expressed in dB.

Table 4.4 shows the performance of our real–time heuristic in terms of Δ_{end} and V_{seg} when C is changed, for a fixed $\alpha = 0.5$. On one hand, a low value of C causes higher variations in quality; indeed,

when $C = 1$ s, we observed the loss of several video frames because of playback buffer starvation. On the other hand, a high value for C causes low bandwidth efficiency, because the adaptation is not reactive enough with respect to the variations of the available bandwidth (since $\Delta_0 = C$, this also makes the user wait for a longer time before he can start watching the video).

In Table 4.5 we varied the smoothing parameter α for a fixed $C = 5$ s. As we can see, a high value of α gives better bandwidth efficiency, at the price of a relatively higher variability in quality.

4.8 Conclusions

We presented a new framework for streaming stored FGS encoded videos. We derived analytical results, and a method to find an optimal transmission policy that maximizes a measure of bandwidth efficiency and minimizes a measure of coding rate variability. We then presented a real–time algorithm for adaptive streaming of FGS video. Our simulations with TCP traces showed that our heuristic yields near–optimal performance in a wide range of bandwidth scenarios. In the context of streaming stored FGS video using client buffering, we have argued that streaming over TCP gives video quality results that are comparable to streaming over smoother TCP–friendly connections. Finally, we have presented an implementation of an end–to–end streaming application which uses MPEG–4 FGS videos to adapt in real–time and with low complexity to varying network conditions. Tests in realistic network situations have shown that our system gives good visual performance despite low efficiency of current FGS–encoding schemes.

In this chapter, we have introduced the concept of scene–based streaming: the server changes the encoding rate of the streamed FGS enhancement layer at scene boundaries, in order to maintain a constant rendering quality within a same video scene. In the next chapter, we study this concept of scene–based streaming in more details. We evaluate streaming at different aggregation levels (images, GoPs, scenes), based on the rate–distortion characteristics of FGS–encoded video.

Chapter 5

Rate–Distortion Properties of MPEG–4 FGS Video for Optimal Streaming

In this chapter, we analyze the rate–distortion properties of MPEG–4 FGS videos, in order to get insights for optimal rate–distortion streaming. We define performance metrics and build a library of rate–distortion traces for long videos. We use our traces to investigate rate–distortion optimized streaming at different video frame aggregation levels.

5.1 Introduction

In our previous approach for streaming FGS–encoded videos, we have considered the problem of maximizing the video rendered quality by using network–oriented metrics, such as the bandwidth efficiency. However, the rendered video quality is not directly proportional to the number of bits received. Receiving more bits results generally in better video quality, but all video packets do not have the same importance with respect to the final perceived quality of the video. Therefore, optimizing the rendered quality of streamed video requires accounting for the specificities of actual video codecs and video sequences.

As mentioned in Section 2.4.2, the goal of rate–distortion optimized streaming is to exploit the rate–distortion characteristics of the encoded video, in order to maximize the overall video quality at the receiver while meeting the constraints imposed by the underlying network. The maximization of the overall quality is generally achieved by maximizing the quality of the individual video frames and by minimizing the variations in quality between consecutive video frames [146]. The optimization procedure usually takes the rate–distortion functions of all individual video frames into account. With FGS–encoded video the optimization procedure at the server is to find the optimal number of enhancement layer bits to send for each image, subject to the bandwidth constraints.

In this chapter, we make two main contributions. First, we analyze FGS enhancement layer rate–

distortion curves for different videos. The provided rate–distortion curves make it possible to assess the quality of the decoded video over lossy network transport with good accuracy[1]. Our analysis gives several insights for rate–distortion optimized streaming systems; in particular, we find that the semantic content and that base layer coding have significant impact on the FGS enhancement layer properties.

Secondly, we examine rate–distortion optimized streaming on the basis of different frame aggregation levels. The optimization per video frame can be computationally demanding, which may reduce the number of simultaneous streams that a high–performing server can simultaneously support. We explore an alternative optimization approach where the server groups several consecutive frames of the video into sequences and performs rate–distortion optimization over the sequences. In this approach, each frame within a given sequence is allocated the same number of bits. We demonstrate that by exploiting the strong correlations in quality between consecutive images, this aggregation approach has the potential to decrease the computational requirement of the optimization procedure, and thereby the computational load on video servers.

5.1.1 Related Work

Significant efforts have gone into the development of the FGS amendment to the MPEG–4 standard, see for instance [71, 100] for an overview of these efforts. Following standardization, the refinement and evaluation of the FGS video coding has received considerable interest [55, 74, 101, 102, 126, 127, 135, 143]. Recently, the streaming of FGS video has been examined in a number of studies, all of which are complementary to our work (see Section 4.1.1 for related works on streaming of FGS–encoded videos). In Chapter 4, we proposed a real–time algorithm for adaptive streaming of FGS–encoded video and introduced the concept of scene–based streaming. However, the proposed algorithm does not take the rate–distortion characteristics of the encoded video into consideration, and the scene–based streaming approach was not evaluated with rate–distortion data.

This chapter is a follow–up of [38]. Fitzek and Reisslein [38] studied the traffic characteristics of single–layer (non–scalable) MPEG–4 and H.263 encoded video for different video quality levels. The quality level was controlled by the quantization scale of the encoder. However, neither the video quality nor the relationship between video traffic (rate) and video quality (distortion) were quantitatively studied. In [105], Reisslein et al. study the video traffic, quality, and rate–distortion characteristics of video encoded into a single layer and video encoded with the conventional temporal and spatial scalability modes. In contrast to [38] and [105], in this dissertation we consider the new Fine Granularity Scalability mode of MPEG–4 and study quantitatively the video traffic (rate), video quality (distortion), as well as their relationship (rate–distortion) for FGS encoded video.

[1] We note here that the subjectively perceived video quality is very complex to assess and the topic of ongoing research; our framework allows for complex metrics, but uses the PSNR for numerical studies.

This chapter is organized as follows. In Section 5.2 we present our framework for analyzing FGS video streaming. We define metrics based on individual video frames, and metrics based on aggregations of video frames (such as Groups of Pictures or visual scenes). In Section 5.3 we analyze the traces for a short video, and for a representative library of long videos from different genres[2]. Long traces are essential to obtain statistically meaningful performance results for video streaming mechanisms, and it is important to consider videos from a representative set of genres because the rate–distortion characteristics depend strongly on the semantic video content. In Section 5.4 we compare the rate–distortion optimized streaming at different video frame aggregation levels. We summarize our findings in Section 5.5.

5.2 Framework for Analyzing Streaming Mechanisms

In this section, we present our framework for analyzing streaming mechanisms for FGS–encoded video: we define metrics that characterize the traffic and quality on the basis of individual video frames and on the basis of scenes (or more generally any arbitrary aggregation of video frames), we explain how to use PSNR and MSE quality measures, and we detail our method for generating the rate–distortion traces.

5.2.1 Notation

We assume that the frame period (display time of one video frame) is constant and denote it by T_f seconds. Let N denote the number of frames in a given video and let n, $n = 1, \ldots, N$, index the individual video frames. Frame n is supposed to be decoded and displayed at the discrete instant $t = n \cdot T_f$. The base layer was encoded with fixed quantization scale, resulting in variable base layer frame sizes (as well as variable enhancement layer frame sizes). Let X_n^b denote the size of the base layer of frame n (in bit or byte), and $\bar{X}^b$ denote the average base layer frame size for the entire video. Let $X_{n,comp}^e$ denote the size of the complete FGS enhancement layer of frame n, i.e., the enhancement layer without any cuts. The base layer is transmitted with constant bit rate $r_n^b = X_n^b/T_f$ during the period from $t = (n-1) \cdot T_f$ to $t = n \cdot T_f$, $n = 1, \ldots, N$. Similarly, the complete enhancement layer would be streamed at the constant bit rate $r_{n,comp}^e = X_{n,comp}^e/T_f$ from $t = (n-1) \cdot T_f$ to $t = n \cdot T_f$. Recall that, according to the fine granularity property, the FGS enhancement layer can be truncated anywhere before (or during) the transmission through the network. The remaining — actually received — part of the FGS enhancement layer is added to the reliably transmitted base layer and decoded. We refer to the part of the enhancement layer of a frame that is actually received and decoded as *enhancement layer subframe*. More formally, we introduce the following terminology. We say that the enhancement layer subframe is encoded at rate r_n, $0 \leq r_n \leq r_{n,comp}^e$, when the first $r_n \cdot T_f$ bits of frame n are received and decoded

[2] All traces and statistics are made publicly available at `http://trace.eas.asu.edu/indexfgs.html`

together with the base layer. In other words, the enhancement layer subframe is said to be encoded with rate r_n when the last $(r^e_{n,comp} - r_n) \cdot T_f$ bits have been cut from the FGS enhancement layer and are not decoded.

For the scene based metrics the video is partitioned into consecutive scenes. Let S denote the total number of scenes in a given video. Let s, $s = 1, \ldots, S$, denote the scene index and N_s the length (in number of images) of scene number s. (Note that $\sum_{s=1}^{S} N_s = N$.) All notations that relate to video scenes can be applied to any arbitrary sequence of successive frames (e.g., GoP). In the remainder of the report, we explicitly indicate when the notation relates to GoPs rather than to visual scenes.

5.2.2 Image-based Metrics

Let $Q_n(r)$, $n = 1, \ldots, N$, denote the quality of the nth decoded image, when the enhancement layer subframe is encoded with rate r; for ease of notation we write here and for all image related metrics r instead of r_n. Let $Q^b_n = Q_n(0)$, denote the quality of the same image, when only the base layer is decoded. We define $Q^e_n(r) = Q_n(r) - Q^b_n$ as the improvement (increase) in quality which is achieved when decoding the enhancement layer subframe encoded with rate r together with the base layer of frame n.

The mean and sample variance of the image quality, are estimated as:

$$\bar{Q}(r) = \frac{1}{N}\sum_{n=1}^{N} Q_n(r), \tag{5.1}$$

$$\sigma^2_Q(r) = \frac{1}{N-1}\sum_{n=1}^{N}[Q_n(r) - \bar{Q}(r)]^2 = \frac{1}{N-1}\left\{\sum_{n=1}^{N}[Q_n(r)]^2 - [\bar{Q}(r)]^2\right\}. \tag{5.2}$$

The coefficient of quality variation is given by:

$$CoV_Q(r) = \frac{\sigma_Q(r)}{\bar{Q}(r)}. \tag{5.3}$$

The autocorrelation coefficient of the image qualities $\rho_Q(r,k)$ for lag k, $k = 1, \ldots, N$, is estimated as:

$$\rho_Q(r,k) = \frac{1}{N-k}\sum_{n=1}^{N-k}\frac{[Q_n(r) - \bar{Q}(r)] \cdot [Q_{n+k}(r) - \bar{Q}(r)]}{\sigma^2_Q(r)}. \tag{5.4}$$

Let $Q_{s,n}(r)$, $s = 1, \ldots, S$, $n = 1, \ldots, N_s$, denote the quality of the nth decoded image of scene s, when the enhancement layer subframe is encoded with rate r. Similar to $Q_n(r)$, we denote $Q_{s,n}(r) = Q^b_{s,n} + Q^e_{s,n}(r)$. The mean and sample variance of the qualities of the images within scene s are denoted by $\bar{Q}_s(r)$ and $\sigma^2_{Q_s}(r)$. They are estimated in the same way as the mean and sample variance of individual image quality over the entire video.

We denote the total size of image n by $X_n(r) = X^b_n + X^e_n(r)$, when the enhancement layer subframe is encoded with rate r, whereby $X^e_n(r) = r \cdot T_f$.

The key characterization of each FGS encoded frame is the rate–distortion curve of the FGS enhancement layer. This rate–distortion curve of a given frame n is a plot of the improvement in image quality Q_n^e as a function of the enhancement layer subframe bitrate r. This rate–distortion curve is very important for evaluating network streaming mechanisms for FGS encoded video. Suppose that for frame n the streaming mechanism was able to deliver the enhancement layer subframe at rate r. Then we can read off the corresponding improvement in quality as $Q_n^e(r)$ from the rate–distortion curve for video frame n. Together with the base layer quality Q_n^b, we obtain the decoded image quality as $Q_n(r) = Q_n^b + Q_n^e(r)$.

In order to be able to compare streaming mechanisms at different aggregation levels, we monitor the maximum variation in quality between consecutive images within a given scene s, $s = 1, \ldots, S$, when the enhancement layer subframes of all images in the considered scene are coded with rate r. We denote this *maximum variation in image quality* by $Var_s(r)$:

$$Var_s(r) = \max_{n=2,\ldots,N_s} \{|Q_{s,n}(r) - Q_{s,n-1}(r)|\}. \tag{5.5}$$

We define the *average maximum variation in image quality* of a video with S scenes as:

$$\overline{Var}(r) = \frac{1}{S} \sum_{s=1}^{S} Var_s(r). \tag{5.6}$$

We also define the *minimum value of the maximum quality variation* of a video with S scenes as:

$$minVar(r) = \min_{1 \le s \le S} Var_s(r). \tag{5.7}$$

5.2.3 Scene–based Metrics

Typically, long videos feature many different scenes composed of successive images with similar visual characteristics. Following Saw [115], we define a *video scene* as a sequence of images between two scene changes, where a scene change is defined as any distinctive difference between two adjacent images. (This includes changes in motion as well as changes in the visual content.)

In this section we define metrics for studying the quality of long videos scene by scene. We first note that the mean image quality of a scene, $\bar{Q}_s(r)$ defined in Section 5.2.2, may not necessarily give an indication of the overall quality of the scene. This is because the quality of individual images does not measure temporal artifacts, such as mosquito noise (moving artifacts around edges) or drifts (moving propagation of prediction errors after transmission). In addition, high variations in quality between successive images within the same scene may decrease the overall perceptual quality of the scene. For example, a scene with alternating high and low quality images may have the same mean image quality as when the scene is rendered with medium but constant image quality, but the quality perceived by the user is likely to be much lower.

For these reasons we let $\Theta_s(r)$ denote the *overall quality* of video scene number s, $s = 1, \ldots, S$, when the enhancement layer subframes have been coded at rate r for all images of the scene. Similar to the measure of quality of the individual images, we define $\Theta_s(r) = \Theta_s^b + \Theta_s^e(r)$, where $\Theta_s^b = \Theta_s(0)$ denotes the overall quality of scene s when only the base layer is decoded, and $\Theta_s^e(r)$ the improvement in quality achieved by the enhancement layer subframes coded at rate r. We analyze the mean $\bar{\Theta}(r)$, sample variance $\sigma_\Theta^2(r)$, coefficient of variation $CoV_\Theta(r)$, and the autocorrelation coefficients $\rho_\Theta(r,k)$ of the scene qualities. We denote the correlation coefficient between the base layer quality of a scene and the aggregate base and enhancement layers quality of a scene by $\rho_{\Theta^b,\Theta}(r)$. These metrics are estimated in analogous fashion to the corresponding image–based metrics.

Note that our measure for overall scene quality, $\Theta_s(r)$, does not account for differences in the length of the successive scenes. Our analysis with a measure that weighted the scene qualities proportionally to the scene length gave very similar results as the scene length independent metric $\Theta_s(r)$. We consider therefore the metric $\Theta_s(r)$ throughout this study. Moreover, it should be noted that the perception of the overall quality of a scene may not be linearly proportional to the length of the scene, but may also depend on other factors, such as the scene content (e.g., the quality of a high action scene may have higher importance than the quality of a very low action scene).

The rate–distortion characteristic of a given scene s is obtained by plotting the curve $\Theta_s(r)$, analogous to the rate–distortion curve of an individual image.

The mean and variance of the scenes' qualities give an overall indication of the perceived quality of the entire video. However, the variance of the scene quality does not capture the differences in quality between successive video scenes, which tend to cause a significant degradation of the perceived overall video quality. To capture these quality transitions between scenes, we introduce a new metric, called *average scene quality variation*, which we define as:

$$V(r) = \frac{1}{S-1} \sum_{s=2}^{S} |\Theta_s(r) - \Theta_{s-1}(r)|. \tag{5.8}$$

Also, we define the *maximum scene quality variation* between two consecutive scenes as:

$$V_{\max}(r) = \max_{2 \le s \le S} |\Theta_s(r) - \Theta_{s-1}(r)|. \tag{5.9}$$

Finally, we monitor the length (in video frames) of the successive scenes N_s, $s = 1, \ldots, S$. We denote the mean and sample variance of N_s as $\bar{N} = N/S$ and σ_N^2.

Table 5.1 summarizes the notations used in this chapter[3].

[3]Note that, in all notations, r denotes the encoding bitrate of the enhancement layer subframe.

T_f	Frame period
N	Total number of frames in the video
X_n^b	Size of the base layer for frame n
$\bar{X}^b$	Average size of base layer frames
$X_{n,comp}^e$	Size of the enhancement layer for frame n
$r_{n,comp}^e$	Total bitrate of the enhancement layer for frame n
S	Number of scenes in the video
N_s	Number of frames in scene s
$Q_n(r)$	Quality of the nth decoded image when the EL is encoded with rate r
$\bar{Q}(r)$	Mean of the image quality
$\sigma_Q^2(r)$	Variance of the image quality
$CoV_Q(r)$	Coefficient of quality variation
$\rho_Q(r,k)$	Autocorrelation coefficient of the image qualities for lag k
$Q_{s,n}(r)$	Quality of the nth decoded image of scene s
$\bar{Q}_s(r)$	Mean quality of the images within scene s
$\sigma_{Q_s(r)}^2$	Variance in quality of the images within scene s
$Var_s(r)$	Maximum variation in image quality within scene s
$\overline{Var}(r)$	Average maximum variation in image quality throughout the video
$minVar(r)$	Minimum value of the maximum quality variation for all scenes of the video
$\Theta_s(r)$	Overall quality of video scene s for EL subframes coded at rate r
$\Theta_s^e(r)$	Improvement in scene quality achieved by the enhancement layer
$\bar{\Theta}(r)$	Mean of the scene quality
$CoV_\Theta(r)$	Coefficient of variation of the scene quality
$\rho_\Theta(r,k)$	Autocorrelation coefficient of the scene quality at lag k
$\rho_{\Theta^b,\Theta}(r)$	Corr. coeff. between the BL and the aggregate BL+EL quality of a scene
$V(r)$	Average scene quality variation
$V_{\max}(r)$	Maximum scene quality variation between two consecutive scenes

Table 5.1: Summary of notations for Chapter 5

5.2.4 MSE and PSNR Measures

The evaluation metrics defined in the previous sections are general in that any specific quality metric can be used for the image quality $Q_n(r)$ and the overall scene quality $\Theta_s(r)$. In this section we explain how to use the Peak Signal–to–Noise Ratio (PSNR) (derived from the Mean Square Error (MSE)) as an instantiation of these general metrics. The choice of PSNR (MSE) is motivated by the recent Video Quality Expert Group (VQEG) report [133]. This report describes extensive experiments that compared several different objective quality measures with subjective quality evaluations (viewing and scoring by humans). It was found that none of the objective measures (some of them quite sophisticated and computationally demanding) performed better than the computationally very simple PSNR (MSE) in predicting (matching) the scores assigned by humans.

For video images of size $X \times Y$ pixels, the PSNR of the video sequence between images n_1 and $n_2 \geq n_1$ is defined by:

$$\text{PSNR}(n_1, n_2) = 10 \cdot \log \frac{M^2}{\text{MSE}(n_1, n_2)}, \tag{5.10}$$

where M is the maximum value of a pixel (255 for 8–bit grayscale images), and $\text{MSE}(n_1, n_2)$ is defined as:

$$\text{MSE}(n_1, n_2) = \frac{1}{X \cdot Y \cdot (n_2 - n_1 + 1)} \sum_{n=n_1}^{n_2} \sum_{y=1}^{Y} \sum_{x=1}^{X} [I(x, y, n) - \tilde{I}(x, y, n)]^2, \tag{5.11}$$

where $I(x, y, n)$ and $\tilde{I}(x, y, n)$ are the gray–level pixel values of the original and decoded frame number n, respectively. The PSNR and MSE are well–defined only for luminance values, not for color [140]. Moreover, as noted in [133], the Human Visual System (HVS) is much more sensitive to the sharpness of the luminance component than to the sharpness of the chrominance component. Therefore, we consider only the luminance PSNR.

To use the PSNR as an instantiation of the generic image quality $Q_n(r)$ and scene quality $\Theta_s(r)$, we set:

$$Q_n(r) = \text{PSNR}(n, n), \tag{5.12}$$

$$Q_{s,n}(r) = \text{PSNR}(\sum_{j=1}^{s-1} N_j + n, \sum_{j=1}^{s-1} N_j + n), \tag{5.13}$$

$$\Theta_s(r) = \text{PSNR}(1 + \sum_{j=1}^{s-1} N_j, \sum_{j=1}^{s} N_j). \tag{5.14}$$

Equation (5.14) assumes that all enhancement layer subframes within scene s are encoded with constant bitrate r. We mention again that we use the MSE and PSNR as an instantiation of our general metrics.

Our general evaluation metrics defined in Sections 5.2.2 and 5.2.3 accommodate any other quality metric, e.g., the ANSI metrics motion energy difference and edge energy difference [5], in a similar manner. (See [90] for an overview of existing objective quality metrics.)

5.2.5 Generation of Traces and Limitations

In our experiments, we used the Microsoft MPEG–4 software encoder/decoder [86] with FGS functionality. The Group of Pictures (GoP) structure of the base layer is set to IBBPBBPBBPBB. We encoded the videos using 2 different sets of quantization parameters for the base layer: the high quality base layer was obtained with the quantization parameters $(4, 4, 4)$ for (I,P,B) frames; the low quality base layer was obtained with the quantization parameters $(10, 14, 16)$. For each base layer quality, we cut the corresponding FGS enhancement layer at the increasing and equally spaced bitrates $r = 200,\ 400,\ 600,\ \ldots$ kbps. The frame period is $T_f = 1/30$ s throughout.

Due to a software limitation in the encoder/decoder, some PSNR results (particularly at some low enhancement layer bitrates) are incoherent (outliers). This has a minor impact for the short videos, because the trend of the rate–distortion curves for all individual images and video scenes is clear enough to estimate the quality that will be reached without considering the outliers. However, for the long videos, only the high quality base layer encoding gave valid results for most enhancement layer bitrates; thus, we only consider the high base layer quality for long videos.

Since the automatic extraction of scene boundaries is still a subject of ongoing research [30, 54, 79, 134] (see [65] for a survey on existing techniques for video segmentation), we restricted the segmentation of the video to the coarser segmentation into *shots* (also commonly referred to as *scene shots*). A shot is the sequence of video frames between two director's cuts. Since shot segmentation does not consider other significant changes in the motion or visual content, a shot may contain several distinct scenes (each in turn delimited by any distinctive difference between two adjacent frames). Nevertheless, distinct scene shots are still likely to have distinct visual characteristics, so we believe that performing shot–segmentation instead of a finer scene segmentation does not have a strong effect on the conclusions of our analysis. A finer segmentation would only increase the total number of distinct video scenes, and increase the correlation between the qualities of the frames in a scene. Many commercial applications can now detect shot cuts with good efficiency. We used the MyFlix software [83], an MPEG–1 editing software which can find cuts directly in MPEG–1 compressed videos. (For the shot segmentation we encoded each video into MPEG–1, in addition to the MPEG–4 FGS encoding.)

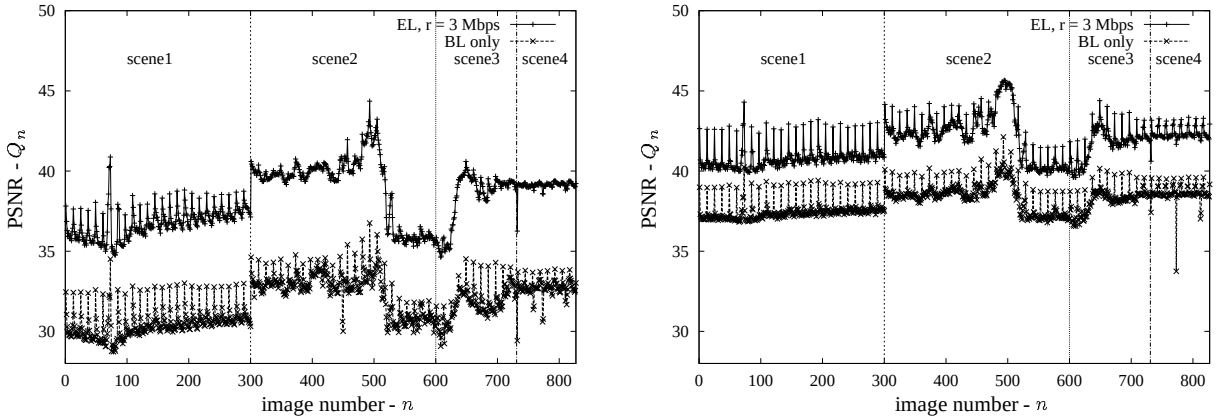
5.3 Analysis of Rate–Distortion Traces

5.3.1 Analysis of Traces from a Short Clip

In this section, we present the analysis of a short video clip of 828 frames encoded in the CIF format. This clip, which we denote by *Clip* throughout, was obtained by concatenating the well–known sequences *coastguard*, *foreman*, and *table* in this order. We segmented (by hand) the resulting clip into 4 scenes ($N_1 = 300$, $N_2 = 300$, $N_3 = 131$, $N_4 = 96$) corresponding to the 4 shots of the video (the *table* sequence is composed of 2 shots).

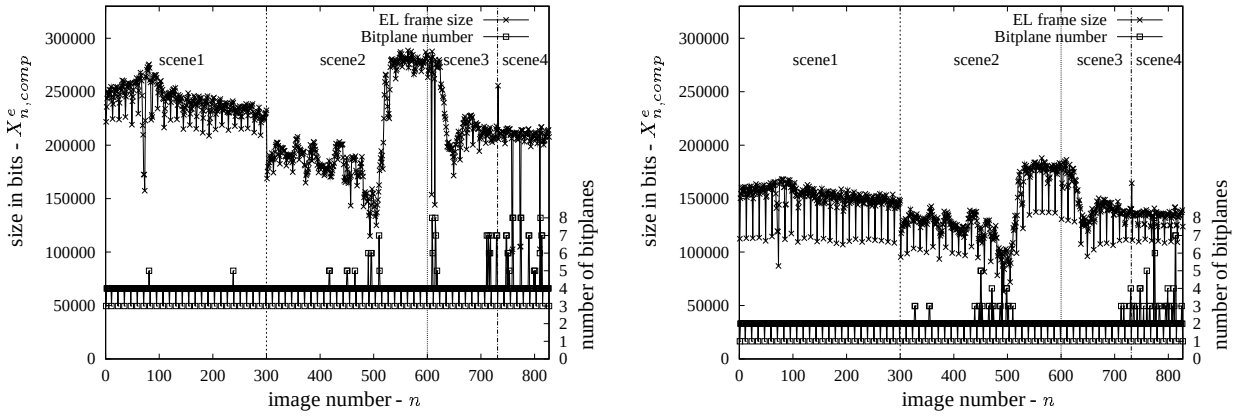
Figure 5.1 shows the quality of the successive images Q_n when only the base layer is decoded and when FGS enhancement layer subframes of rate $r = 3$ Mbps are added to the base layer. We make the following observations for both low and high base layer qualities: (i) First, the average image quality changes from one scene to the other for both base layer–only and EL–enhanced streaming. (ii) For a given scene, we see that for the base layer there are significant differences in the quality of successive images. Most of these differences are caused by the different types of base layer images (I, P, B) — the frames with the highest quality correspond to I–frames. When adding a part of the enhancement layer (at rate $r = 3$ Mbps in the figures), we see that these differences are typically still present, but may have changed in magnitude. This suggests to distinguish between the different types of images in order to study the rate–distortion characteristics of the FGS enhancement layer. (iii) We notice that scenes 2 and 3 feature high variations of image quality even for a given frame type. Scene 2 corresponds to the *foreman* sequence in which the camera pans from the foreman's face to the building. A finer scene segmentation than shot–based segmentation would have segmented scene 2 into two different scenes, because the foreman's face and the building have different visual complexities.

Figure 5.2 shows the size of the complete enhancement layer, $X^e_{n,comp}$, and the number of bitplanes needed to code the enhancement layer of each image (we refer the reader to Section 2.2.4 for details about FGS coding). First, we focus on a given scene. We observe that, in general, I images have fewer bitplanes than P or B images and that the total number of bits for the enhancement layer images is larger for P and B images than for I images. This is because I images have higher base layer quality. Therefore, fewer bitplanes and fewer bits are required to code the enhancement layer of I images. For the same reason, when comparing different high and low base layer qualities, we see that the enhancement layer corresponding to the high base layer quality needs, for most images, fewer bitplanes than the enhancement layer corresponding to the low base layer quality. For low base layer quality, the enhancement layer contains, for most images, 4 bitplanes, whereas, for the high base layer quality, it usually contains 2 bitplanes.



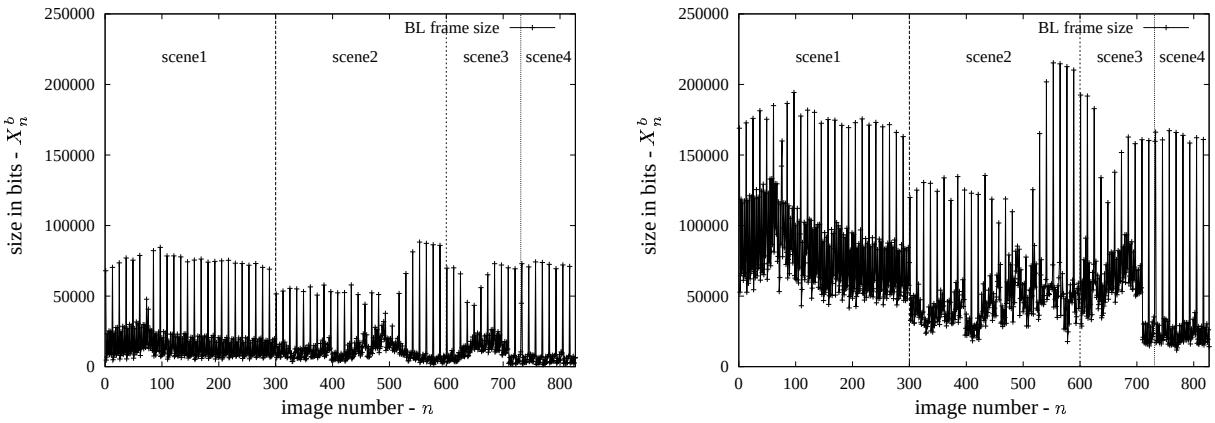
(a) low quality base layer (b) high quality base layer

Figure 5.1: Image quality in PSNR for all images of *Clip*



(a) low quality base layer (b) high quality base layer

Figure 5.2: Size of complete EL frames and number of bitplanes for all frames of *Clip*



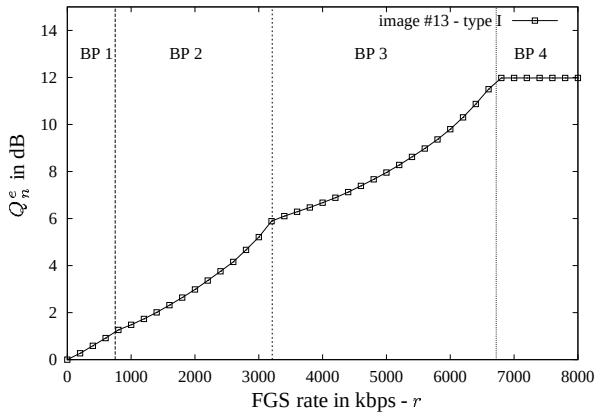
(a) low quality base layer

(b) high quality base layer

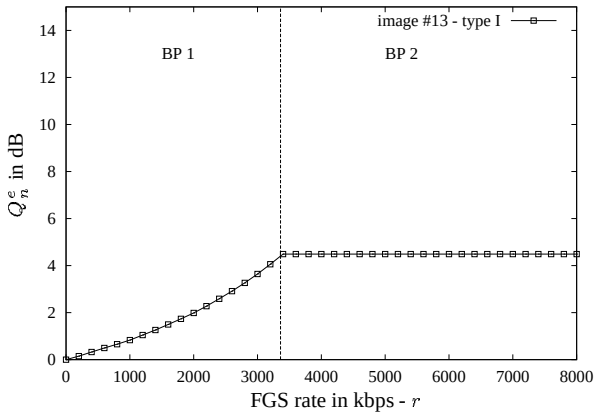
Figure 5.3: Size of base layer images for *Clip*

Next, we conduct comparisons across different scenes. Figure 5.3 shows the size of the base layer frames, X_n^b. When comparing the average size of the enhancement layer frames for the individual scenes (Figure 5.2) with the average size of the corresponding base layer frames (Figure 5.3), we see that the larger the average base layer frame size of a scene the larger the average enhancement layer frame size of the scene. This can be explained by the different complexities of the scenes. For example, for a given base layer quality, we see that it requires more bits to code I images in scene 1 than in the first part of scene 2. This means that the complexity of scene 1 is higher than the complexity of scene 2. Therefore, the average number of bits required to code the enhancement layer of scene 1 images is larger than for the first part of scene 2.

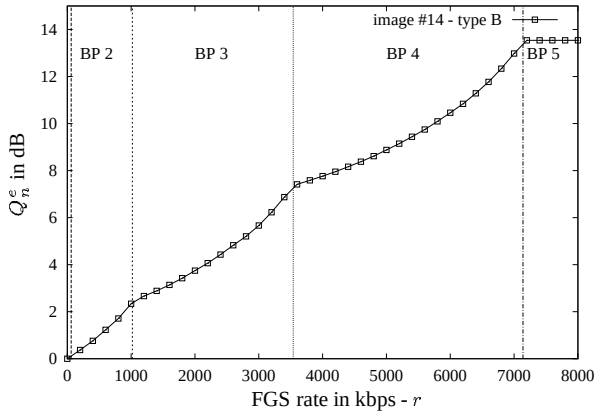
In Figure 5.4 we plot the rate–distortion functions $Q^e_{1,13}(r)$ and $Q^e_{1,14}(r)$ (improvement in quality brought by the enhancement layer as a function of the encoding rate of the FGS enhancement layer) for different types of images within the same GoP. These plots give rise to a number of interesting observations, which, in turn, have important implications for FGS video streaming and its evaluation. First, we observe that the rate–distortion curves are different for each bitplane. The rate–distortion curves of the lower (more significant) bitplanes tend to be almost linear, while the higher (less significant) bitplanes are clearly non–linear. (Note that the most significant bitplane (BP1) for image 14 with low quality base layer has a very small size.) More specifically, the rate–distortion curves of the higher bitplanes tend to be convex. In other words, the closer we get to the end of a given bitplane, the larger the improvement in quality for a fixed amount of additional bandwidth. This appears to be due to the bitplane headers. Indeed, the more bits are kept in a given bitplane after truncation, the smaller the share of the bitplane header in the total data for this bitplane. As a result, when designing streaming mechanisms, it may be worthwhile to prioritize the enhancement layer cutting toward the end of the bitplanes.



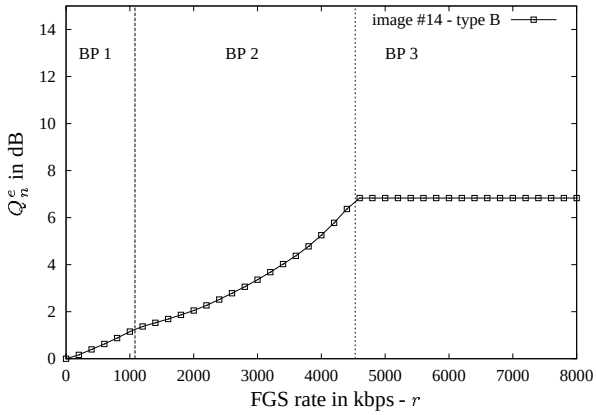
(a) image 13 — low quality base layer



(b) image 13 — high quality base layer



(c) image 14 — low quality base layer

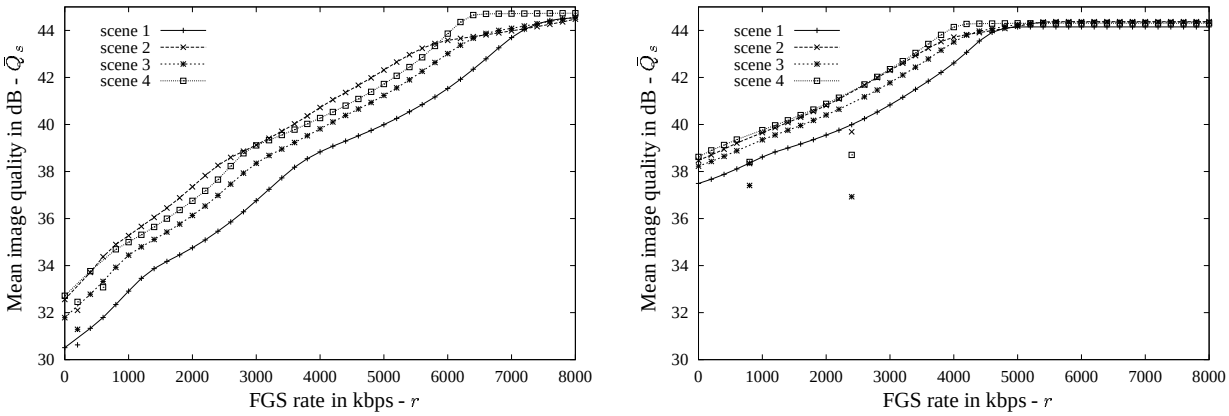


(d) image 14 — high quality base layer

Figure 5.4: Improvement in PSNR as function of the FGS bitrate for scene 1 images of *Clip*

Recall that the plots in Figure 5.4 are obtained by cutting the FGS enhancement layer every 200 kbps. We observe from these plots that a piecewise linear approximation of the curve using the 200 kbps spaced sample points gives an accurate characterization of the rate–distortion curve. We also observe that approximating the rate distortion–curves of individual bitplanes by straight lines (one for each bitplane) can result in significant errors (typically in the range from 0.5 – 0.75 dB and up to 1 dB). It is therefore recommended to employ a piecewise linear approximation based on the 200 kbps spaced sample points. An interesting avenue for future work is to fit analytical functions to our empirically measured rate–distortion curves.

So far, we have considered the rate–distortion curves of individual frames. We now aggregate the frames into scenes and study the rate–distortion characteristics of the individual scenes. Figure 5.5 shows the average image quality (from base plus enhancement layer) of the individual scenes in the



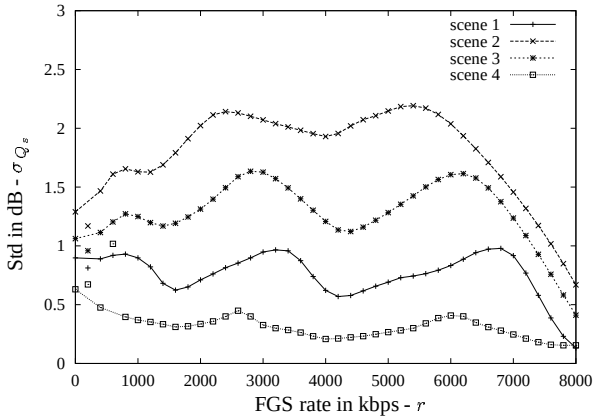
(a) low quality base layer

(b) high quality base layer

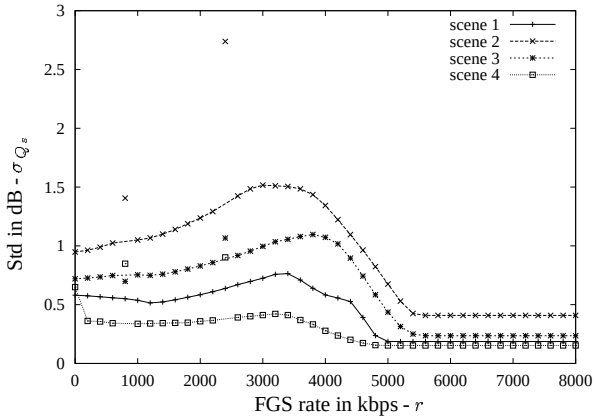
Figure 5.5: Average image quality by scene as a function of the FGS–EL bitrate for *Clip*

Clip as a function of the FGS enhancement layer rate. (The outliers at low FGS bitrates are due to the software limitation discussed in Section 5.2.5.) We observe that the scenes differ in their rate–distortion characteristics. For the low quality base layer version, the PSNR quality of scene 1 (*coastguard*) is about 2 dB lower than the PSNR quality of scene 2 (*foreman*) for almost the entire range of enhancement layer rates. This quality difference falls to around 1 dB for the high quality base layer video. This appears to be due to the higher level of motion in *coastguard*. Encoding this motion requires more bits with MPEG–4 FGS, because there is no motion compensation in the enhancement layer. Overall, the results indicate that it is prudent to (i) analyze FGS encoded video on a scene by scene basis (which we do in the next section for long video with many scenes), and (ii) to take the characteristics of the individual scenes into consideration when streaming FGS video (which we examine in more detail in Section 5.4).

As noted in the introduction, the perceived video quality depends on the qualities of the individual frames as well as on the variations in quality between successive frames. To examine the quality variations, we plot in Figure 5.6 the standard deviation of the image quality σ_{Q_s} for the different scenes. For both base layer qualities, we observe that overall scene 2 (*foreman*) is the scene with the largest variance. This is due to the change of visual complexity within the scene as the camera pans from the foreman's face to the building behind him. We also observe that for a given scene, the variance in quality can change considerably with the FGS enhancement layer rate. To examine the cause for these relatively large and varying standard deviations, we plot in Figure 5.7 the standard deviation of both image quality σ_Q and GoP quality σ_Θ for the entire video clip. We see that the standard deviation of the GoP quality is negligible compared to the standard deviation of the image quality. This indicates that most of the variations in quality are due to variations in image quality between the different types of images (I, P, and B) within a given GoP. Thus, it is, as already noted above, reasonable to take the frame type into

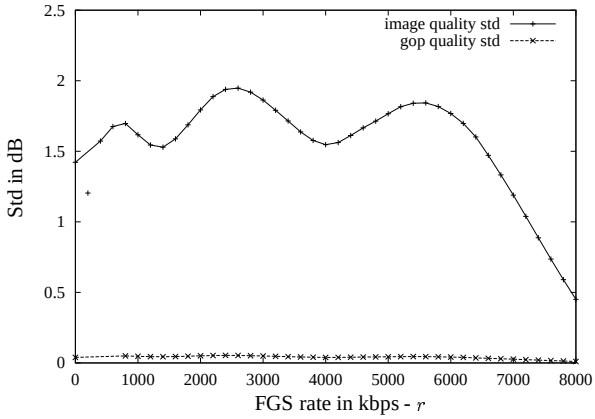


(a) low quality base layer

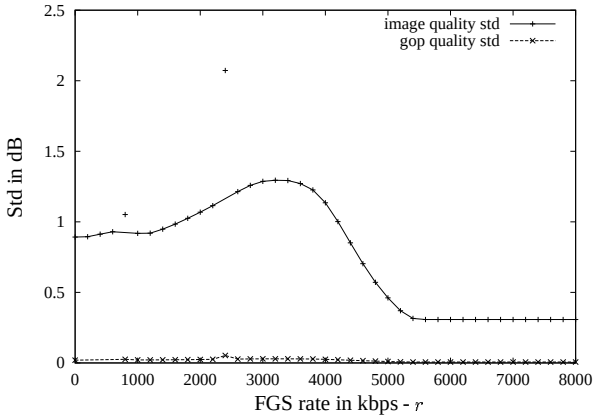


(b) high quality base layer

Figure 5.6: Std of image quality for individual scenes as a function of the FGS bitrate for *Clip*

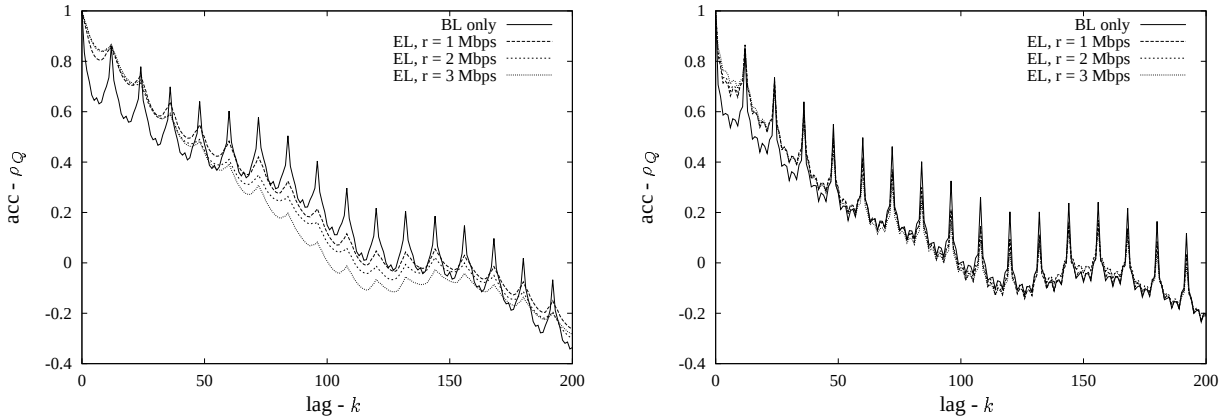


(a) low quality base layer



(b) high quality base layer

Figure 5.7: Std of image quality and GoP quality as a function of the FGS bitrate for *Clip*



(a) low quality base layer

(b) high quality base layer

Figure 5.8: Autocorrelation coefficient of image quality for *Clip*

consideration in the streaming.

To take a yet closer look at the quality variations, we plot in Figure 5.8 the autocorrelation function ρ_Q of the image quality for the base layer and the FGS enhancement layer coded at rates $r = 1, 2$ and 3 Mbps. We observe periodic spikes which correspond to the GoP pattern. We verify that, at small lags there are high correlations (i.e., relatively smooth transitions) in quality for the different types of images, especially for high FGS enhancement layer rates. This means that a higher FGS enhancement layer rate smoothes the difference in quality between near images. Indeed, for the same number of FGS enhancement layer bits added to the base layer, the gain in quality is different for consecutive I, P, and B frames. In general, the gain in quality for I frames is smaller than the gain in quality for P or B frames. As indicated earlier, the base layer has higher quality for I frames; therefore, the enhancement layer bits provide higher (less visible) spatial frequencies for the I frames than for the P and B frames.

5.3.2 Analysis of Traces from Long Videos

In this section we analyze the traces of long videos. All videos have been captured and encoded in QCIF format (176×144 pixels), except for the movie *Silence* (for *Silence of the Lambs*), which has been captured and encoded in CIF format (352×288 pixels). All videos have been encoded with high base layer quality. The image–based metrics defined in Section 5.2.2 lead to similar insights for long videos as found in Section 5.3.1. In contrast to *Clip*, the long videos contain many different scenes and thus allow for a statistically meaningful analysis at the scene level.

Table 5.2 gives the scene shot length characteristics of the long videos. We observe that the scene lengths differ significantly among the different videos. *Toy Story* has the shortest scenes, with an average

	run time	S	$\bar{N}$	CoV_N	$N_{\max}/\bar{N}$
The Firm	1h	890	121	0.94	9.36
Oprah+com	1h	621	173	2.46	39.70
Oprah	38mn	320	215	1.83	23.86
News	1h	399	270	1.67	9.72
Star Wars	1h	984	109	1.53	19.28
Silence (CIF)	30mn	184	292	0.96	6.89
Toy Story	1h	1225	88	0.95	10.74
Football	1h	876	123	2.34	31.47
Lecture	49mn	16	5457	1.62	6.18

Table 5.2: Scene shot length characteristics for the long videos

scene length of just about 2.9 seconds (= 88 frames/30 frames per second). Comparing Oprah with commercials (*Oprah + com*) with *Oprah* (same video with commercials removed), we observe that the commercials significantly reduce the average scene length and increase the variability of the scene length in the video. The *lecture* video, a recording of a class by Prof. M. Reisslein, has by far the longest average scene length, with the camera pointing to the writing pad or blackboard for extended periods of time. The scene length can have a significant impact on the required resources (e.g., client buffer) and the complexity of streaming mechanisms that adapt on a scene by scene basis. (In Section 5.4 we compare scene by scene based streaming with other streaming mechanisms from a video quality perspective.)

Table 5.3 gives elementary base layer traffic statistics of our long videos. The base layer statistics are quite typical for encodings with fixed quantization scales (4,4,4)[4]. Note that *Oprah* with and without commercials have the highest base layer average bitrate among QCIF movies, with commercials increasing the bitrate by around 60%; *Star Wars* has the lowest base layer bitrate, which is probably due to the high number of scenes with dark backdrops or with little contrast in this type of movies.

Table 5.4 presents the average scene quality statistics for our long videos. Table 5.4 indicates that the average scene PSNR is very different from one video to the other. In particular, while *Oprah with commercials* and *Oprah* have the highest base layer encoding rates, the average overall PSNR quality achieved for the base layer for both videos is low compared to the average PSNR quality achieved by the other videos. This appears to be due to the high motion movie trailers featured in the show as well as noise from the TV recording, both of which require many bits for encoding. We observe that for a given video, each additional 1 Mbps of enhancement layer increases the average PSNR by roughly 3–4 dB. (The relatively large bitrate and low PSNR for the *Lecture* video are due to the relatively noisy copy of

[4]We refer the interested reader to [38] for a detailed study of these types of traffic traces.

	$\bar{r}_b$ (Mbps)	$\bar{X}^b$ (bits)	CoV_{X^b}	$X^b_{\max}/\bar{X}^b$
The Firm	0.65	21765	0.65	6.52
Oprah+com	2.73	91129	0.14	1.94
Oprah	1.69	56200	0.19	2.33
News	0.74	24645	0.54	5.30
Star Wars	0.49	16363	0.65	6.97
Silence (CIF)	1.74	57989	0.72	7.85
Toy Story	1.08	36141	0.49	5.72
Football	0.97	32374	0.53	3.90
Lecture	1.54	51504	0.29	2.72

Table 5.3: Base layer traffic characteristics for the long videos

	BL only		$r = 1$ Mbps			$r = 2$ Mbps		
	$\bar{\Theta}$ (dB)	CoV_Θ	$\bar{\Theta}$ (dB)	CoV_Θ	$\rho_{\Theta^b,\Theta}$	$\bar{\Theta}$ (dB)	CoV_Θ	$\rho_{\Theta^b,\Theta}$
The Firm	36.76	0.013	40.10	0.017	0.92	43.70	0.003	-0.18
Oprah+com	35.71	0.015	38.24	0.013	0.99	42.30	0.010	0.83
Oprah	35.38	0.003	38.18	0.003	0.84	42.84	0.007	0.48
News	36.66	0.018	39.65	0.027	0.83	43.76	0.021	0.25
Star Wars	37.48	0.025	41.14	0.031	0.97	43.83	0.013	0.67
Silence (CIF)	37.88	0.015	NA	NA	NA	39.70	0.020	0.53
Toy Story	36.54	0.021	39.57	0.029	0.98	43.95	0.013	0.90
Football	37.42	0.034	40.69	0.041	0.97	43.97	0.018	0.81
Lecture	35.54	0.001	38.48	0.002	0.52	43.64	0.007	-0.17

Table 5.4: Scene quality statistics of long videos for the base layer and FGS enhancement layer

	base layer only		$r = 1$ Mbps		$r = 2$ Mbps	
	V	$V_{\max}$	V	$V_{\max}$	V	$V_{\max}$
The Firm	0.06	1.83	0.16	2.37	0.00	1.26
Oprah+com	0.04	12.15	0.05	11.31	0.03	7.32
Oprah	0.00	0.36	0.00	0.42	0.00	1.13
News	0.11	3.15	0.29	3.17	0.12	2.36
Star Wars	0.29	8.25	0.57	8.83	0.13	6.28
Silence (CIF)	0.05	1.42	NA	NA	0.25	4.45
Toy Story	0.19	9.77	0.40	11.25	0.12	6.23
Football	0.51	9.79	0.72	10.12	0.19	6.36
Lecture	0.00	0.14	0.00	0.20	0.00	0.64

Table 5.5: Average scene quality variation and maximum scene quality variation of long videos

the master tape.) We also observe from Table 5.4 that the coefficient of variation of the scene qualities is relatively small. This is one reason why we defined the average scene quality variation V in (5.8) and maximum scene quality variation $V_{\max}$ in (5.9), which focus more on the quality change from one scene to the next (and which we will examine shortly). The other point to keep in mind is that these results are obtained for fixed settings of the FGS enhancement layer rate r. When streaming over a real network, the available bandwidth is typically variable and the streaming mechanism can exploit the fine granularity property of the FGS enhancement layer to adapt to the available bandwidth, i.e., the enhancement layer rate r will become a function of time as in Chapter 4.

In Table 5.5, we first observe that *Oprah* has the smallest average scene quality variation V at all FGS rates, whereas the *Football* video has the largest average scene quality variation for the base layer and $r = 1$ Mbps. For most videos, V and $V_{\max}$ are both minimum at $r = 2$ Mbps. We see from $V_{\max}$ that the difference in quality between successive scenes can be as high as 12 dB and is typically larger than 2 dB at all FGS rates for most videos. This indicates that there are quite significant variations in quality between some of the successive video scenes, which may visibly affect the video quality.

Figures 5.9 and 5.10 give, for *The Firm* and *News* respectively, the scene quality as a function of the scene number ($\Theta_s(r)$) for the base layer and FGS cutting rates $r = 1$ and 2 Mbps, as well as the average encoding bitrates for the base layer and the base layer plus complete enhancement layer. The plots illustrate the significant variations in scene quality for an enhancement layer rate of r = 1 Mbps. The quality variations are less pronounced for the base layer, while, for $r = 2$ Mbps, the quality is almost constant for most scenes. With r = 2 Mbps, most scenes are encoded at close to maximum achievable quality.

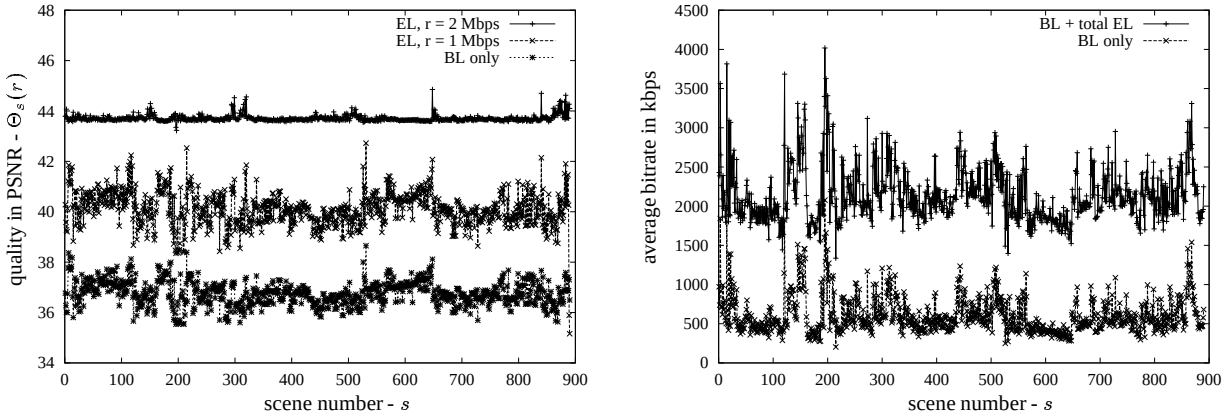


Figure 5.9: Scene PSNR (left) and average encoding bitrate (right) for all scenes of *The Firm*

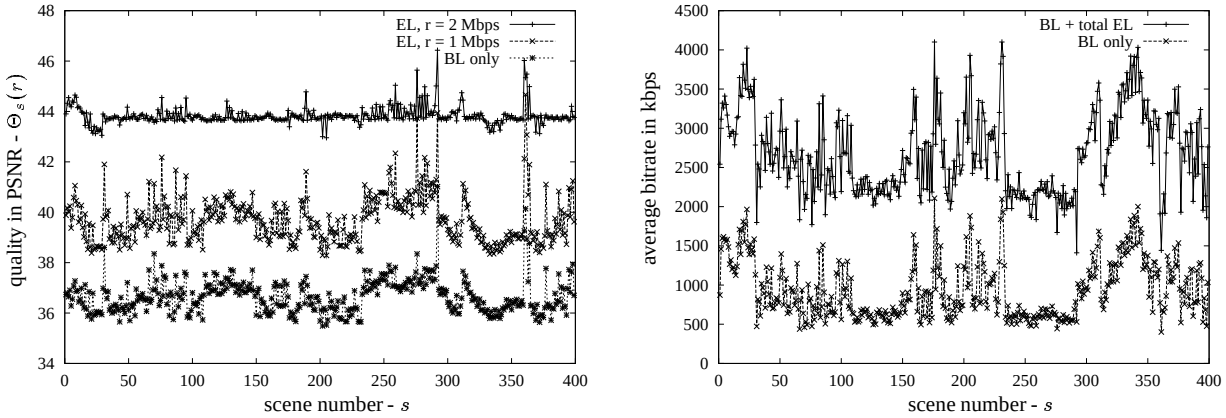


Figure 5.10: Scene PSNR (left) and average encoding bitrate (right) for all scenes of *News*

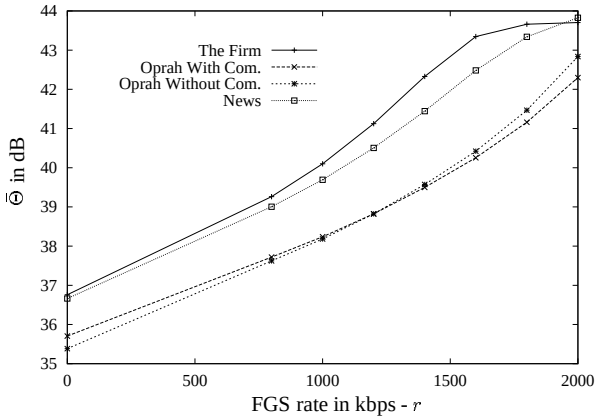


Figure 5.11: Average scene quality as a function of the FGS bitrate

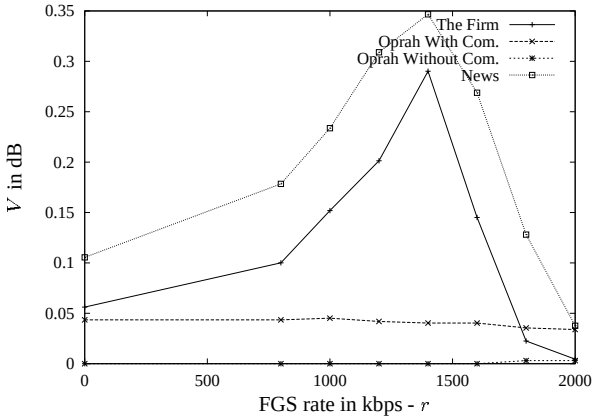


Figure 5.12: Average scene quality variability as a function of the FGS bitrate

Figure 5.11 shows the average scene quality for *The Firm*, *Oprah*, *Oprah with commercials* and *News*, as a function of the FGS rate ($\bar{\Theta}(r)$). We observe that the slope of the quality increase with increasing FGS enhancement layer rate is about the same for all considered videos. We also observe that there is a difference of around 1 dB between the average base layer quality for *The Firm* or *News* and the average base layer quality for *Oprah* or *Oprah with commercials*; this difference roughly remains constant at all FGS rates. This indicates that the average quality achieved by a video at all FGS rates strongly depends on the visual content of the videos and on the average quality of the base layer. This is confirmed in Figure 5.13 which shows the coefficient of scene correlation between the base layer, and the aggregate base and enhancement layer quality as a function of the FGS rate ($\rho_{\Theta^b,\Theta}(r)$). The correlation decreases slightly with the FGS rate but stays high at all rates (see Table 5.4 for complete statistics for all videos).

Figure 5.12 shows the average scene quality variation as a function of the FGS rate ($V(r)$). As we see, the difference in quality between the successive scenes first increases with the FGS rate for *The Firm* and *News*. This is probably because some scenes can achieve maximum quality with a small number of enhancement layer bits (low complexity scenes), while other scenes require a higher number of bits to achieve maximum quality (high complexity scenes). At high FGS rates the variability starts to decrease because all scenes tend to reach the maximum quality (as confirmed in Table 5.5 for most videos). For *Oprah* and *Oprah with commercials*, the variability stays very low at all FGS rates, which is mainly due to the fact that the VBR–base layer encoder has been able to smooth the differences in scene quality effectively.

Finally, Figure 5.14 shows, for each video, the autocorrelation in scene quality ρ_Θ for the base layer and FGS rates $r = 0$, 1, and 2 Mbps. For the four videos, we observe that the autocorrelation functions drop off quite rapidly for a lag of a few scene shots, indicating that there is a tendency of abrupt changes in quality from one scene to the next. Also, for a given video, the autocorrelation function for the aggregate

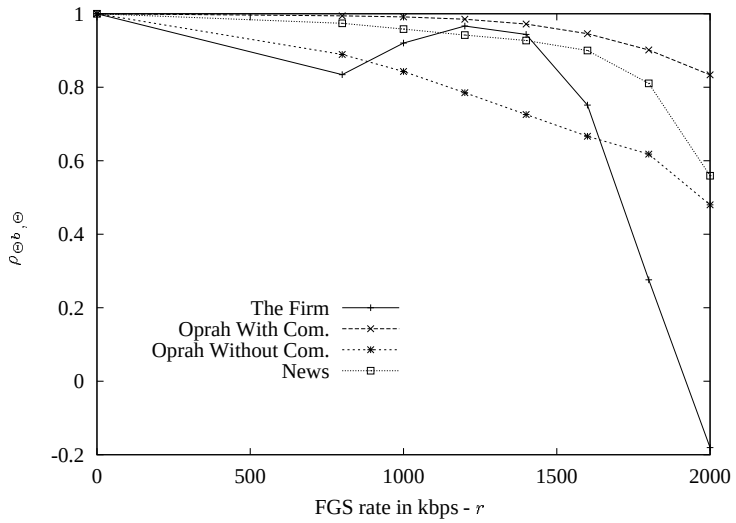


Figure 5.13: Coeff. of correlation between BL and overall quality of scenes, as a function of the FGS bitrate

base and enhancement layers follows closely the autocorrelation function for the base layer only, except for *Oprah with commercials* at $r = 2$ Mbps. The difference in autocorrelation at low lags between *Oprah* and *Oprah with commercials* can be explained by the higher diversity of successive scene types when adding commercials.

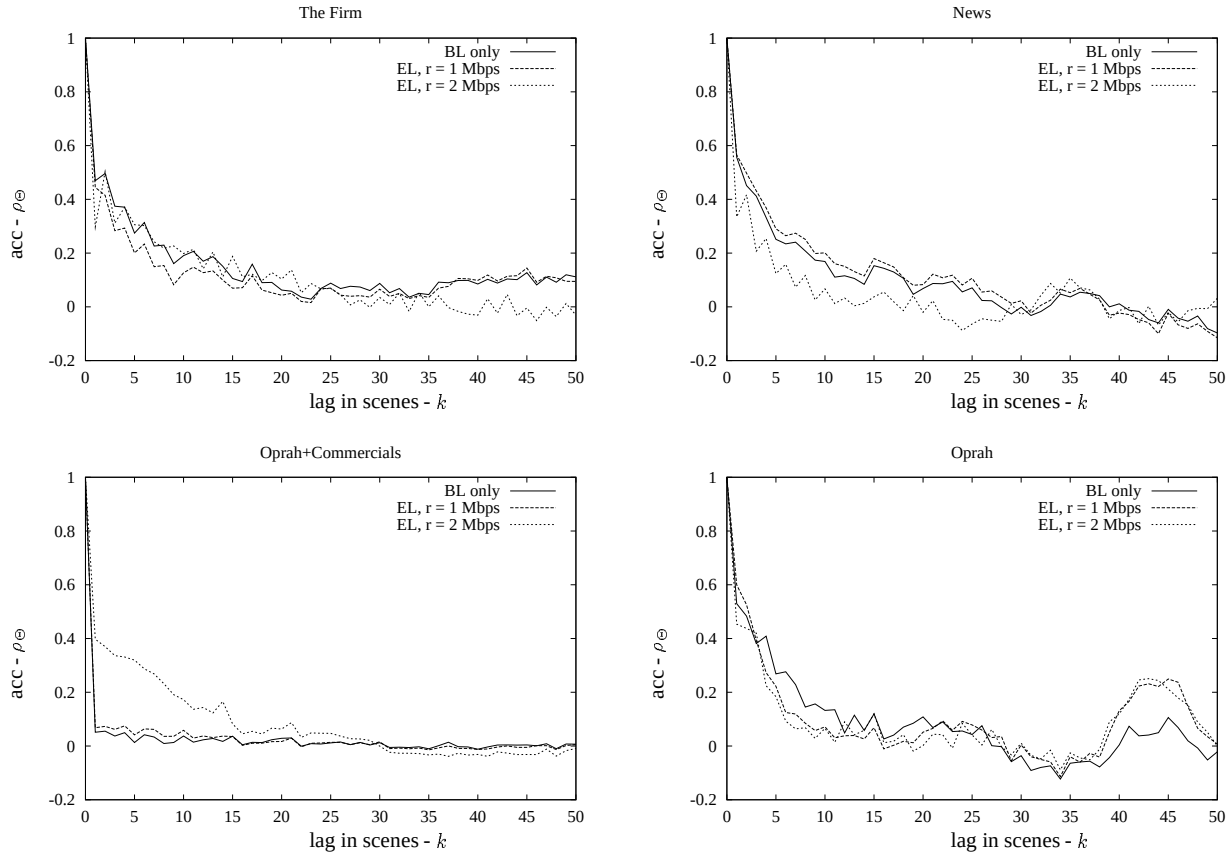


Figure 5.14: Autocorrelation in scene quality for videos encoded with high quality base layer

5.4 Comparison of Streaming at Different Image Aggregation Levels

In this section we use our traces and metrics to compare rate–distortion optimized streaming of FGS–encoded video at different levels of image aggregation.

5.4.1 Problem Formulation

We suppose that the transmission of the base layer is made reliable, and we focus on the streaming of the enhancement layer. When streaming video over the best–effort Internet, the available bandwidth typically fluctuates over many time–scales. However, as we explained in Section 2.4.1, for non real–time applications such as streaming stored video, the user can usually tolerate an initial playback delay, during which some initial part of the video is stored into the client buffers before the start of the playback. Maintaining a sufficient playback delay throughout the rendering allows the application to accommodate future bandwidth variations, as we showed in Chapters 3 and 4.

To account for bandwidth variability, we model bandwidth constraints and client buffering resources as follows. As shown on Figure 5.15, the video is partitioned into M *allocation segments*, with each allocation segment m containing the same number of frames $\lfloor N/M \rfloor$. While the server is streaming the video, for each allocation segment m, the server assigns a maximum bandwidth budget $B_{\max} = R_{\max} \cdot \lfloor N/M \rfloor \cdot T_f$ bits to be allocated across all the frames in the segment, where the maximum average bitrate $R_{\max}$ varies from one segment to the next. In this section, we focus on the allocation of the bandwidth budget to the individual frames within a segment. In our experiments, we use allocation segments consisting of 1000 frames, which correspond to about 30 seconds of a 30 f/s video.

Due to the client buffering, the server has great flexibility in allocating the given bandwidth budget to the frames within a segment. Given the rate–distortion functions of all images, the server can optimize the streaming within the segment by allocating bits from the bandwidth budget to the individual frames so as to maximize the video quality. Alternatively, the server can group several consecutive images of an allocation segment into sub–segments, referred to as *streaming sequences*, and perform rate–distortion optimization on the granularity of streaming sequences. In this case, each frame in a streaming sequence (that is sub–segment) is allocated the same number of bits. We denote by S_m the number of streaming sequences in a given allocation segment m, $m = 1, \ldots, M$.

We consider five aggregation cases for streaming sequences:

– **image**: each image from the current allocation segment forms a distinct streaming sequence ($S_m = \lfloor N/M \rfloor$).

– **gop**: we group all images from the same GoP into one streaming sequence. In this case, the

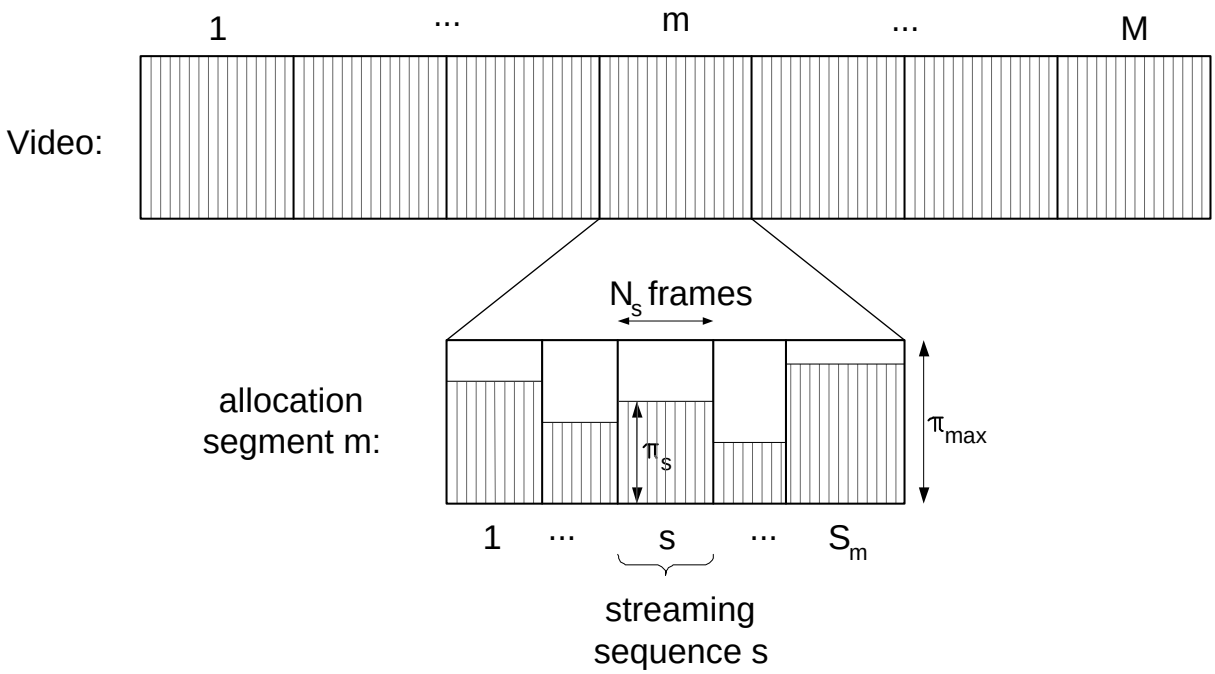


Figure 5.15: Partitioning of the video into allocation segments and streaming sequences

number of streaming sequences in allocation segment m is equal to the number of distinct GoPs in the allocation segment ($S_m = \lfloor N/(12 \cdot M) \rfloor$ with the 12 image GoP structure used in this study).

– **scene**: we group all images from the same video scene into one streaming sequence. In this case $S_m = S_m^{scene}$, where S_m^{scene} denotes the number of distinct scenes in allocation segment m, according to the initial segmentation of the video (shot–based segmentation in this study).

– **constant**: allocation segment m is divided into $S_m^{const} = S_m^{scene}$ streaming sequences, each containing the same number of frames. Consequently, each streaming sequence contains a number of frames equal to the average scene length of the allocation segment.

– **total**: all the images from allocation segment m form one streaming sequence ($S_m = 1$).

In the following, we focus on the streaming of a particular allocation segment m. In order to simplify the notation, we remove the index m from all notations whenever there is no ambiguity. Let S be the number of streaming sequences in the current allocation segment. Let N_s be the number of frames in streaming sequence s, $s = 1, \ldots, S$ (see Figure 5.15). For a given allowed average rate $R_{\max}$, let π_s denote the number of bits allocated to each of the N_s images in streaming sequence s. Define $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_S)$ as the streaming policy for the current allocation segment. We denote by $\pi_{\max}$ the

maximum number of enhancement layer bits that can be allocated to any image of the video (this depends on the total coding rate of the enhancement layer).

Extending the scene quality metric defined in Section 5.2.3 to allocation segments, we define $\Theta(\boldsymbol{\pi})$ as the overall quality of the current allocation segment under the streaming policy $\boldsymbol{\pi}$. We denote $D(\boldsymbol{\pi})$ for the corresponding average distortion and $B(\boldsymbol{\pi})$ for the total number of bits to stream under this policy. As explained in Section 5.2.4, the distortion of a sequence of successive frames is measured in terms of the average MSE, which is obtained by averaging the MSEs of individual frames. The overall quality of a sequence is measured in terms of the PSNR and computed directly from the average MSE of the sequence. We denote by $d_{s,n}(\pi)$ the distortion (in terms of MSE) of image $n,\ n = 1, \ldots, N_s$, of streaming sequence s, when its enhancement layer subframes are encoded with π bits. We denote by $D_s(\pi) = \frac{1}{N_s}\sum_{i=1}^{N_s} d_{s,i}(\pi)$, the average distortion of streaming sequence s when all enhancement layer subframes contain π bits. With these definitions we can formulate the rate–distortion streaming optimization problem as follows:

Problem 5.1

For the current allocation segment, a given bandwidth constraint $R_{\max}$, and a given aggregation case, the optimization procedure at the server consists of finding the policy $\boldsymbol{\pi}^* = (\pi_1^*, \ldots, \pi_S^*)$ that optimizes:

$$\textit{minimize } D(\boldsymbol{\pi}) = \sum_{s=1}^{S}\left(\frac{1}{N_s}\sum_{i=1}^{N_s} d_{s,i}(\pi_s)\right)$$

$$\textit{subject to: } B(\boldsymbol{\pi}) = N_1 \cdot \pi_1 + \cdots + N_S \cdot \pi_S \leq B_{\max},$$

$$\pi_s \leq \pi_{\max}, s = 1, \ldots, S.$$

We denote $D^* = D(\boldsymbol{\pi}^*)$ (respectively Θ^*) for the minimum total distortion (maximum overall quality) achieved for the current allocation segment. Our problem is a resource allocation problem, which can be solved by dynamic programming[5] [32]. The most popular technique to solve such an optimization problem is called *recursive fixing*. This consists in evaluating recursively the optimal decisions from the ending state to the starting state of the system. This is similar to the well–known Dijkstra algorithm which is used to solve shortest–path problems (this is the method we used in the previous chapter for Theorem 4.3).

As we have observed in Figure 5.4, the rate–distortion curves cannot easily be modeled by a simple function. Therefore, we implemented recursive fixing by sampling the possible values of enhancement

[5]Dynamic programming is a set of techniques that are used to solve various decision problems. In a typical decision problem, the system transitions from state to state according to the decision taken for each state. Each transition is associated with a profit. The problem is to find the optimal decisions from the starting state to the ending state of the system, i.e., the decisions that maximize the total profit, or in our context minimize the average distortion.

layer bits per image in steps of 833 bytes ($=$ 200 kbps $\times$ 0.033 s$/8$), with a maximum of $\pi_{\max} = 8333$ bytes per image. Recall that our long traces have been obtained by cutting the enhancement layer bitstream at 200 kbps, 400 kbps, 600 kbps, ..., 2 Mbps. A finer granularity solution to the optimization problem could be obtained by interpolating the rate-distortion curve between the 200 kbps spaced points and using a smaller sampling step size in the recursive fixing, which in turn would increase the required computational effort.

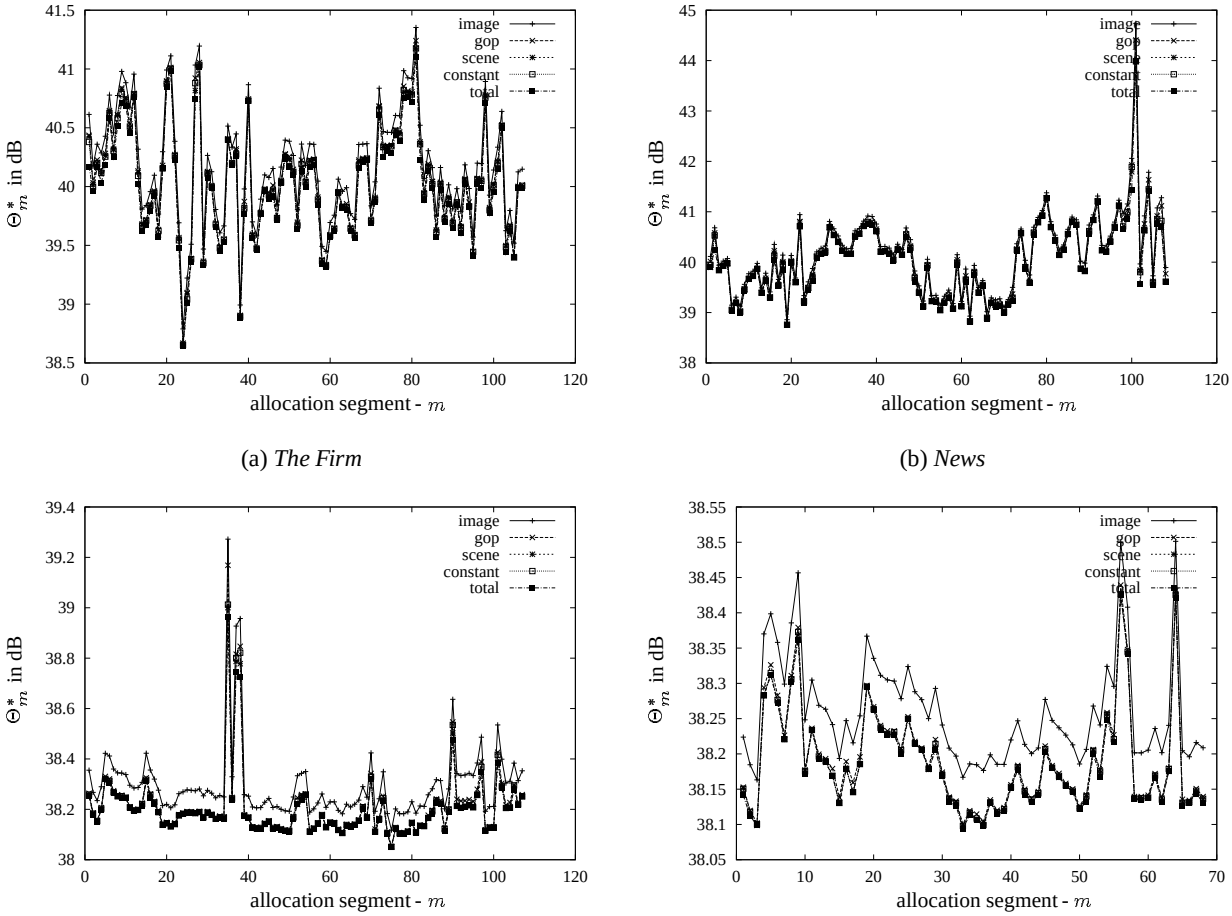
The computational effort required for resolving our problem depends on the aggregation case which is considered (image, gop, scene, constant, or total), on the length of an allocation segment, as well as on the number of scenes within a given allocation segment. Since scene shots are usually composed of tens to thousands of frames, the reduction in computational complexity when aggregating frames within a shot (scene case) or aggregating an arbitrary number of frames (constant case) is typically significant. For instance, in *The Firm*, with an encoding at 30 frames per second, there are on average only around 8 scene shots in one allocation segment of 1000 frames.

5.4.2 Results

Figure 5.16 depicts the maximum overall quality Θ_m^* as a function of the allocation segment number m, for an average target rate of $R_{\max} = 1000$ kbps for *The Firm*, *Oprah*, *Oprah with commercials* and *News*. Not surprisingly, for all allocation segments, the overall quality with image–by–image streaming optimization is higher than for the other aggregation cases. This is because the image–by–image optimization is finer. However, the overall quality achieved by the other aggregation cases is very close to that of image–by–image streaming: the difference is usually less than 1 dB in PSNR. This is due to the high correlation between the enhancement layer rate–distortion functions of successive frames.

We show in Figure 5.17 the optimal quality averaged over all allocation segments for the entire videos as a function of the target rate constraint $R_{\max}$. We observe that the gop, scene, constant and total aggregation cases give about the same optimal quality for all target rates. Figure 5.18 presents the results obtained for low and high base layer quality versions of *Clip*. For both base layer versions, we see that the best quality is achieved for the image aggregation followed by gop, scene, constant and total aggregation in this order. We observed in other experiments, which are not presented here, that the qualities are very similar when the allocation segments contain more than 1000 frames.

As we have seen from the average MSE–based metrics plotted in the previous figures, there seems to be very little difference between the maximum quality achieved for all allocation segments when aggregating over scenes or arbitrary sequences. However, the actual perceived quality may be different. The reason is that the MSE does not account for temporal effects, such as the variations in quality between consecutive images: two sequences with a same average MSE may have different variations



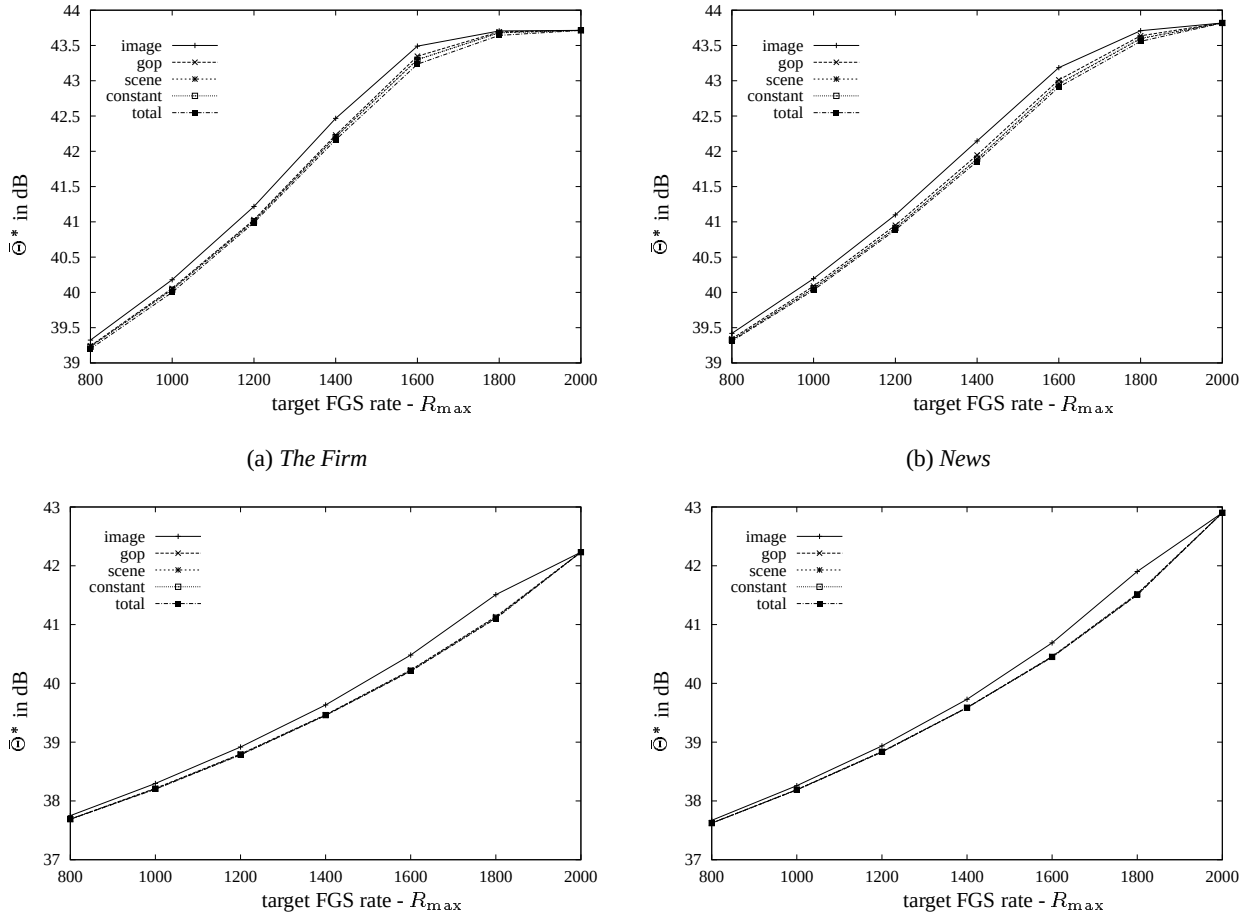
(a) *The Firm*

(b) *News*

(c) *Oprah with commercials*

(d) *Oprah*

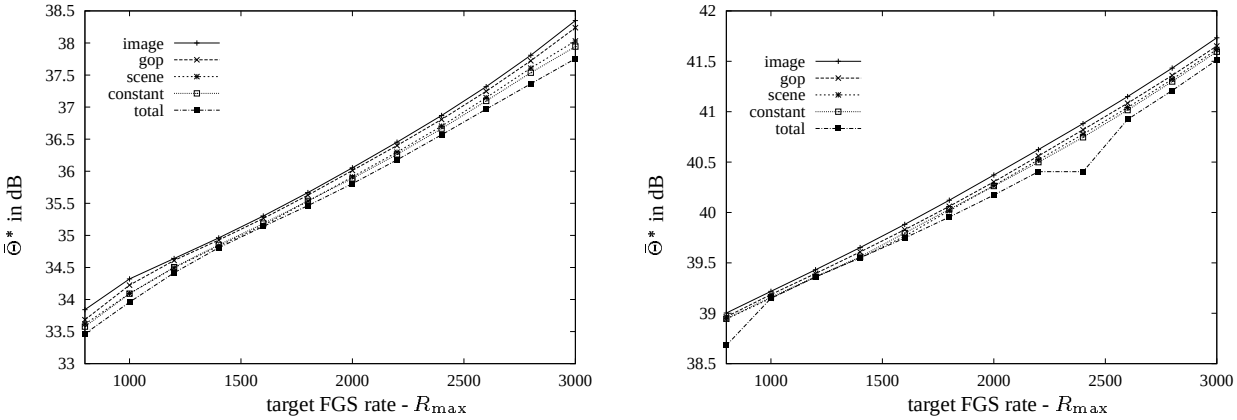
Figure 5.16: Maximum overall quality as a function of allocation segment number



(a) *The Firm* (b) *News*

(c) *Oprah with commercials* (d) *Oprah*

Figure 5.17: Average maximum quality as a function of the enhancement layer (cut off) bitrate



(a) low quality base layer

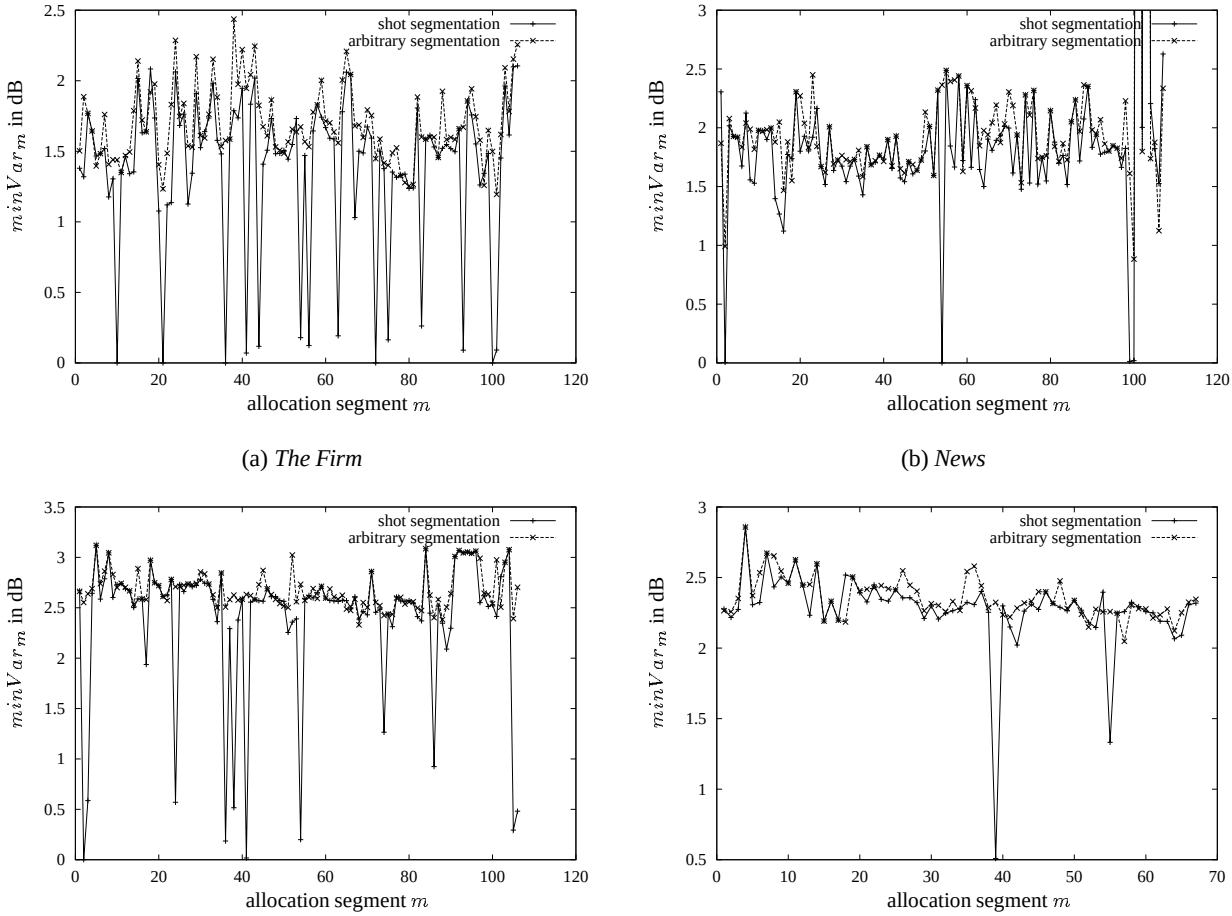
(b) high quality base layer

Figure 5.18: Average maximum quality as a function of the enhancement layer bitrate for *Clip*

in image MSE, and thus different perceived quality. To illustrate this phenomenon, we monitor the maximum quality variation between consecutive images Var_s of a given streaming sequence (defined in (5.5)). For the streaming sequence s, $s = 1, \ldots, S_m$, with each enhancement layer subframe encoded with π bits, the maximum variation is $Var_s(\pi/T) = \max_{i=2,\ldots,N_s}\{|Q_{s,i}(\pi/T) - Q_{s,i-1}(\pi/T)|\}$.

For allocations segments of 1000 frames, Table 5.6 shows the average maximum variation in quality $\overline{Var}$ (defined in (5.6)) for different FGS rates. We observe that the average maximum variation in quality for a given FGS rate is always smaller with scene aggregation than with constant or total aggregation. This means that selecting a constant number of bits for the enhancement layer of all images within a given video shot yields on average a smaller maximum variation in image quality than selecting a constant number of bits for an arbitrary number of successive images. Therefore, it is preferable to choose streaming sequences that correspond to visual shots rather than segmenting the video arbitrarily. This result is intuitive; frames within a given shot are more likely to have similar visual complexity, and thus similar rate–distortion characteristics, than frames from different shots. This is confirmed in Figure 5.19, which shows the minimum value of the maximum variations in quality over all scenes of a given allocation segment $minVar$ (defined in (5.7)), for $R_{max} = 800$ kbps. We observe that the min–max variation in image quality is typically larger for arbitrary segmentation. This indicates that the minimum jump in quality in the streaming sequences of a given allocation segment is larger for arbitrary segmentation. In the case of shot segmentation the minimum jumps in quality are smaller; when the shot consists of one homogeneous video scene, $minVar_m$ is close to 0 dB. As shown in Figure 5.19, for some allocation segments, the difference with arbitrary segmentation can be more than 1 dB.

More generally, we expect the difference in rendered quality between shot–based segmentation and arbitrary segmentation to be more pronounced with a scene segmentation that is finer than shot–based



(a) *The Firm*

(b) *News*

(c) *Oprah with commercials*

(d) *Oprah*

Figure 5.19: Min–max variations in quality as a function of the allocation segment number

	$R_{\max} = 800$ kbps			$R_{\max} = 1600$ kbps		
	scene	const.	total	scene	const.	total
The Firm	1.84	1.99	2.51	0.81	0.92	1.44
OprahWith	2.64	2.77	2.99	2.68	2.76	2.93
Oprah	2.43	2.47	2.60	2.60	2.64	2.75
News	2.22	2.55	3.71	1.43	1.64	2.89
StarWars	1.90	2.11	3.44	0.85	0.97	1.84
Silence (CIF)	1.33	1.37	1.82	1.37	1.40	1.88
Toy Story	2.34	2.54	3.54	1.46	1.74	2.96
Football	2.21	2.56	4.92	1.09	1.38	3.49
Lecture	2.64	2.69	2.73	2.06	2.08	2.12
Clip high BL	2.27	2.44	4.75	1.67	1.88	2.72
Clip low BL	2.14	2.49	3.87	1.76	2.23	3.67

Table 5.6: Average maximum variation in quality for long videos and *Clip*

segmentation. A finer segmentation would further segment sequences with varying rate–distortion characteristics, e.g., sequences with changes in motion or visual content other than director's cuts. This would increase the correlation between the qualities of the frames in a same scene, thereby further reducing the quality degradation due to scene–based streaming over image–based streaming.

5.5 Conclusions

In this chapter we have analyzed rate–distortion traces of MPEG–4 FGS videos based on several performance metrics. The defined metrics capture the quality of the received and decoded video both at the level of individual video frames (images) as well as aggregations of images (GoP, scene, etc). The rate–distortion traces provide the rate–distortion characteristics of the FGS enhancement layer for a set of long videos from different genres. Our analysis of the traces provides a number of insights that are useful for the design of streaming mechanisms for FGS–encoded video. First, the convex form of the rate–distortion curves for individual bitplanes suggests to prioritize the cutting of the bitstream close to the end of the bitplanes. (Note however that cutting the enhancement layer bitstream only at bitplane boundaries would provide coarser granularity in adapting video quality to varying network conditions.) Secondly, the base layer frame types (I, P, and B) and, in general, the base layer coding tend to have a significant impact on the total quality obtained from the base layer plus FGS enhancement layer stream. We observed that, for fixed FGS enhancement layer cut–off rates, significant variations in the base layer quality correspond to significant variations in the total (base + enhancement layer) quality. This suggests

to take the different base layer frame types into consideration in the streaming of the FGS enhancement layer frames. We also observed that, for fixed FGS enhancement layer cut–off rates, the total video quality tends to vary according to the different semantic content of the video scenes. This suggests to take the scene structure into consideration in the enhancement layer streaming.

We have used our traces to investigate rate–distortion optimized streaming at different image aggregation levels. We have found that the optimal scene–by–scene adjustment of the FGS enhancement layer rate reduces the computational complexity of the optimization significantly compared to image–by–image optimization, while having only a minor impact on the video quality. We have observed that reducing the computational optimization effort by aggregating the images arbitrarily (without paying attention to the scene structure) tends to result in quality deteriorations.

The goal of rate–distortion optimal streaming is to maximize the quality of the received video, by evaluating the expected rendered quality at the server, as a function of network conditions and rate–distortion properties of each video packet. However, current optimization approaches from the literature do not account for the possibility of error concealment at the decoder. In the next chapter, we present a unified framework for rate–distortion optimal streaming with accounting for decoder error concealment.

Chapter 6

Unified Framework for Optimal Streaming using MDPs

In this chapter we consider streaming layered video (live and stored) over a lossy packet network. We propose an end–to–end unified framework in which packet scheduling and error control decisions at the sender explicitly account for the error concealment mechanism at the receiver. We show how the theory of infinite–horizon, average–reward Markov decision processes with average–cost constraints, can be applied to find optimal transmission policies. We demonstrate the framework and solution procedure using MPEG–4 FGS video traces.

6.1 Introduction

In a typical streaming application, the sender *schedules* the transmission of media packets in order to maximize the rendered video quality. The sender may choose not to transmit some media packets, thereby not sending some layers in some frames (this is also called quality adaptation [108]). Over lossy channels, scheduling is followed by *error correction* in order to mitigate the effects of packet loss on the rendered video. Error correction for streaming media typically consists in the retransmission of lost packets that can arrive at the receiver before their decoding deadlines, or the transmission of redundant forward error correction packets (also called error protection). Both scheduling and error correction should jointly adapt to the variations of network conditions, such as the available bandwidth and packet loss rate of the connection. In our framework, error correction is provided by Reed–Solomon (RS) FEC codes (see Section 2.3.2).

At the receiver, some of the media packets are available on time, that is, before their decoding deadlines. Other packets are not available, either because they were transmitted and lost, or simply because the sender never scheduled them for transmission. At the time of rendering to the user, the decoder typ-

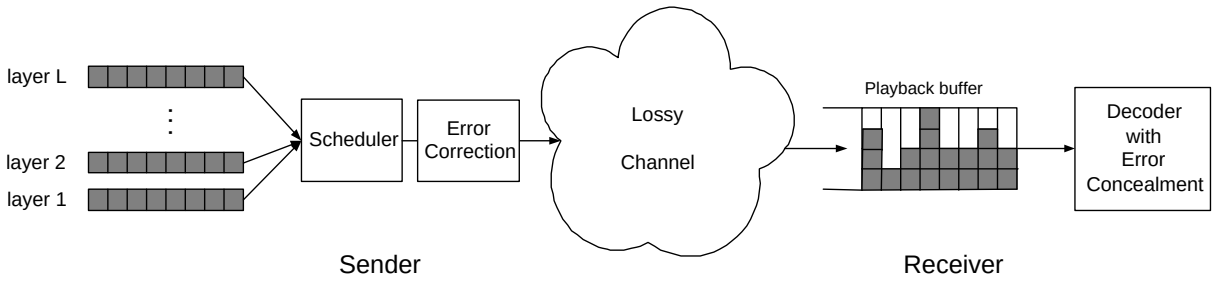


Figure 6.1: Video streaming system with decoder error concealment

ically applies several methods of *error concealment* in order to best conceal the missing packets. Error Concealment (EC) consists in exploiting the spatial and temporal correlations of audio or video to interpolate missing packets from the surrounding available packets [137]. For video, a simple and popular method for temporal error concealment is to display, instead of the missing macro block from the current frame, the macro block at the same spatial location from the previous frame.

Packet scheduling, error correction and error concealment are fundamental components in an end–to–end video streaming system. Figure 6.1 illustrates their respective functions. At the sender, the scheduler determines the layers that should be sent to the receiver for each frame of the video; the error protection component determines the amount of FEC packets to be sent with each layer. At the receiver, incoming packets are stored temporarily in the playback buffer; before rendering the media, the decoder performs error concealment from the available layers. Traditionally, scheduling and error correction transmission policies are normally optimized without taking into account the presence of error concealment at the receiver [27, 85, 97].

In this dissertation, we argue that the scheduling and error protection components of a video streaming system should be designed jointly with decoder error concealment. In particular, when designing a scheduling and error correction transmission policy, not only should we account for the layered structure of the media, the channel characteristics, and the effects of missing packets on distortion, but we should also explicitly account for error concealment at the receiver. Thus, we argue for a more unified, end–to–end approach for designing video streaming systems.

In this chapter we make several contributions. We present a new unified optimization framework for joint packet scheduling and error correction with considering temporal error concealment in the optimization process. We show how the theory of infinite–horizon, average–reward Markov Decision Processes (MDPs) *with average–cost constraints* can be applied to this optimization problem. To our knowledge, infinite–horizon constrained MDPs have not been applied yet to video streaming. We show that constrained MDPs can be used for a wide variety of quality metrics, including metrics that take quality

variation into account; infinite–horizon MDPs also permit to find optimal policies with computationally tractable procedures. Using simulations from MPEG–4 FGS videos, we show that accounting for decoder error concealment during the joint optimization of scheduling and error protection can enhance the quality of the received video significantly. We find that policies with static error protection strategy give near–optimal performance. Finally, we find that degradations in quality for a channel with imperfect state information are small; thus our MDP approach is suitable for networks with long end–to–end delays.

6.1.1 Related Work

In [26, 27], Chou and Miao have considered scheduling packetized media over a packet erasure channel in order to minimize an additive combination of distortion and average rate. They develop a heuristic algorithm for finding a sub–optimal scheduling policy, whose performance may be significantly below the truly optimal scheduling policy; decoder error concealment is not a central part of their framework. Previous works that considered decoder error concealment for optimal streaming include [46,147]. Frossard and Verscheure [46] study optimal FEC allocation for non–layered video; they consider the problem of minimizing the PDM (Perceptual Distortion Metric), which can be simply expressed as a function of the video source rate and a constant that depends on the error concealment scheme. The approach from Zhang et al. [147] relies on a simple linear estimate for the expected distortion of GoPs after decoding, that can account for decoder error concealment.

Unlike in these approaches, decoder error concealment is a central part of our framework, and our constrained MDP approach provides a tractable means for determining the truly optimal transmission policy. The framework provided in this chapter can also handle quality variability metrics in addition to average distortion metrics.

Other closely related works on optimal streaming of media using a feedback channel include [97, 117]. These works do not consider error concealment. Podolsky et al. [97] study optimal retransmission strategies of scalable media. Their analysis is based on Markov chains with a state space that grows exponentially with the number of layers. Servetto [117] studies scheduling of complete GoPs encoded in multiple description codes. The sender adapts the number of descriptions sent to the receiver as a function of the network state, which is modeled as a HMM (Hidden Markov Model).

Finally, streaming layered video with unequal error protection (UEP) through FEC has been presented in [52, 121, 128, 146]. None of these approaches consider decoder error concealment in the optimization process.

6.1.2 Benefits of Accounting for EC during Scheduling Optimization

In this section we provide a simple example to highlight the benefits of accounting for error concealment during the scheduling optimization. We consider a video segment composed of five frames, each of which is encoded into three layers. We suppose in this example that each frame is independently encoded (there is no motion–compensation); the only dependencies are due to layered encoding, i.e., a given layer of a video frame needs all the lower layers of the same frame to be decoded. We suppose that each layer fits exactly into one packet, all packets are the same size, and all frames have the same rate–distortion functions.

On the left of Figure 6.2, we give the distortion values for each frame before EC at the decoder, as a function of the number of layers which are available for the frame (the available layers at the decoder are represented in grey on the figure). These are distortion values expected by the sender without accounting for temporal EC at the receiver. On the right of Figure 6.2, we show the distortion values for frame n after EC from the previous frame $n-1$ is used. These are the distortion values which are actually obtained after decoding.

Figure 6.3 shows four possible scheduling policies at the sender (A, B, C, and D), when we require each policy to send exactly nine packets. Initially, we suppose that there is no packet loss. For each scheduling policy, we give the total distortion before and after EC at the decoder, which corresponds to the distortion expected by the server without and with accounting for EC, respectively. Now consider the optimal policy without accounting for EC and the optimal policy with accounting for EC. The optimal policy without accounting for EC is policy B, which minimizes the distortion at the receiver before EC (distortion before EC $= 7$). After applying error concealment to policy B, the resulting distortion is 6. Hence, the optimal policy without accounting for EC has a distortion of 6. But the optimal policy with accounting for EC is policy A, which has a lower rendered distortion than policy B after EC. Therefore, not considering decoder EC during the optimization at the server can result in choosing a sub–optimal policy (i.e, policy B instead of policy A in this example).

This chapter is organized as follows. In Section 6.2, we formulate our optimization problem. Section 6.3 gives the experimental setup of our simulations with MPEG–4 FGS videos. In Section 6.4 we show how our optimization problem can be solved by using results from MDPs. In Section 6.5 we investigate how to incorporate additional quality metrics in our framework. Section 6.6 presents the more general case with delayed receiver state information. Finally, we conclude in Section 6.7.

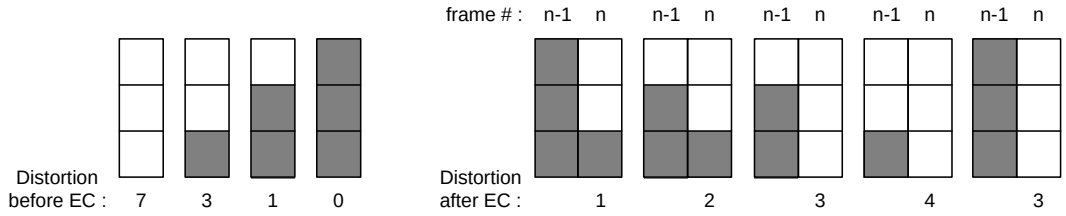


Figure 6.2: Example of distortion values for a video encoded with 3 layers

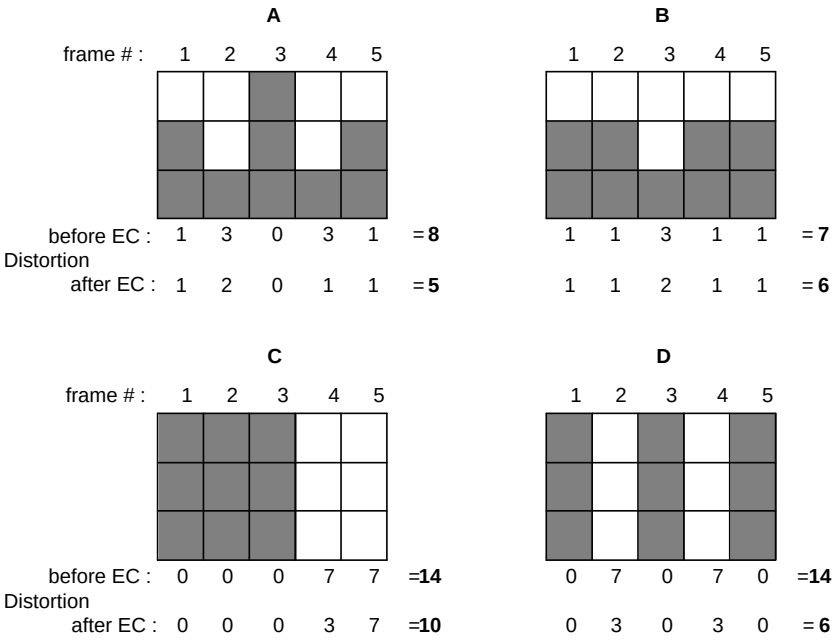


Figure 6.3: Example of scheduling policies transmitting 9 packets

6.2 Problem Formulation

In this chapter, we consider video streaming, live or stored. When streaming layered–encoded video, the reception of the base layer provides minimum acceptable quality. So, the base layer should be transmitted with high reliability. This can be achieved by sufficient playback buffering at the client to allow for the retransmission of most lost video packets before their decoding deadline expires [76], or by protecting the base layer with a high amount of FEC codes. Additionally, transmitting the base layer with high reliability permits the use of highly bit–rate efficient — despite poorly error resilient — encoding methods such as motion–compensation. We suppose that the base layer is transmitted to the client without loss, and we focus on determining optimal policies for the transmission of the enhancement layers.

The video at the sender is encoded into L enhancement layers. Recall that the main property of layered–encoded video is that layer l of a given frame cannot be decoded unless all lower layers $1, \ldots, l-1$ are also available at the decoder. Let N be the number of frames in the video.

We suppose that the L enhancement layers are not motion–compensated, i.e., the decoding of layer l of frame n does not depend on the decoding of the enhancement layers for previous frames. As we explain in the next section, this assumption corresponds particularly to the case of the FGS enhancement layer defined in the MPEG–4 standard [8]. However, our unified framework stays valid for any highly error resilient layering scheme that does not encode the enhancement layers with motion–compensation. For simplicity of the analysis, we suppose that all enhancement layers have the same size. Each layer contains exactly U source packets. We consider that a given layer is useful at the decoder only if all U source packets of the layer are available.

We suppose that the additional quality brought by a given layer is roughly constant for all frames of the video (i.e., layer l of frame n brings roughly the same amount of quality to frame n as does layer l of frame $n+m$ to frame $n+m$). More generally, for long videos containing multiple scenes with different visual characteristics, the quality brought by a layer is likely to vary for different parts of the video (see Chapter 5). In this case, we suppose that the video has been previously segmented into homogeneous segments of video frames, such that the quality brought by each layer is roughly constant throughout the segment. Therefore, throughout, we consider a single homogeneous segment containing N frames. In the case of longer videos, we would apply our optimization framework to each separate segment.

Throughout this chapter, we suppose that the transmission channel is a packet–erasure channel. The channel has a probability of success of q.

At the decoder, we suppose that, in order to conceal loss of packets for frame number n, only information from previous frame $n-1$ is used. However, information from frame $n-1$ does not necessarily fully conceal loss of packets from frame n. Note that, in practice, information from a set of consecutive

previous frames, and even from subsequent frames, can also be used to perform temporal error concealment for the current frame at the decoder. This has the potential to increase the accuracy in predicting any missing packet, but at the cost of an increase in run–time complexity of the decoder [137]. The theory presented here can be extended to handle these more sophisticated forms of error concealment; however, in order to see the forest through the trees, we focus on using only the previous frame in error concealment.

For a given scheduling and error correction transmission policy σ, let $rate_{avg}(\sigma)$ denote the average transmission rate for the video. It is defined as the average number of packets sent for a frame, normalized by the total number of source packets for the frame (i.e., $L \times U$). Let $dist_{avg}(\sigma)$ denote the average distortion of the rendered video after error concealment. A typical problem formulation of rate–distortion optimized streaming is the following [27, 146]:

Problem 6.1
Find an optimal transmission policy σ that

$$\textit{minimizes } dist_{avg}(\sigma) \textit{ subject to } rate_{avg}(\sigma) \leq \rho,$$

where ρ is the maximum (normalized) transmission rate that is allowed by the network connection, or alternatively, the rate budget that is allocated to the streaming. We denote by $dist^{*}_{avg}$, the minimum distortion achieved by an optimal policy for Problem 6.1.

It may be misleading to solely use the average image distortion, usually expressed in terms of average MSE (Mean Squared Error), to account for the quality of the rendered video. As we mentioned in Section 5.2.3, the average image distortion does not measure temporal artifacts; in particular, high variations in quality between successive images may decrease the overall perceptual quality of the video. Therefore, the formulation of our problem should incorporate additional quality constraints. As in the previous chapter, we treat as an example the case of variations in quality between consecutive images. For a given transmission policy σ, let $var_{avg}(\sigma)$ denote the average variation in distortion between two consecutive images. We can now formulate the following problem:

Problem 6.2
Find an optimal transmission policy σ that

$$\textit{minimizes } dist_{avg}(\sigma) \textit{ subject to } rate_{avg}(\sigma) \leq \rho \textit{ and } var_{avg}(\sigma) \leq \gamma,$$

where γ is the maximum average variation in distortion that is allowed. (Its value can be found from subjective tests.)

Let A_n denote the joint scheduling and error correction action that the sender takes for frame n. This is defined as the total number of packets (source + FEC packets) to send for all layers of frame n: $A_n = \boldsymbol{a} = (a_1, \ldots, a_L)$ where $a_l \in \{0, U, U+1, \ldots, 2U-1\}$ is the total number of packets to send for layer l. (We restrict the number of FEC packets for each layer to be less than the number of source packets, i.e., $a_l < 2U$). Note that the decision $a_l = 0$ means that the sender does not send layer l at all. In particular, this should imply that $a_{l+1} = \ldots = a_L = 0$, because higher layers $l+1, \ldots, L$ will never be decoded if the sender does not send layer l. Because of this hierarchy, our system should also give more protection to lower layers than to higher layers (UEP). Therefore, we should have $a_1 \geq a_2 \geq \ldots \geq a_L$. Let $\mathcal{A}$ denote the set of all possible decisions $(a_1, \ldots, a_L)$ for any frame[1].

Let $X_n \in \mathcal{X} = \{0, 1, \ldots, L\}$ denote the state at the receiver for previous frame $n-1$, i.e., the number of successive layers that are available at the decoder for frame $n-1$. Let D_n denote the distortion of frame n after decoding.

We denote by d_l, the distortion of a frame containing only the first l layers before temporal EC. (Without loss of generality, we take $d_L = 0$ and $d_0 = 1$.) We have $d_L < d_{L-1} < \ldots < d_1 < d_0$. For $0 \leq i, j \leq L$, we denote by d_{ij} the distortion of a frame after temporal error concealment, when i layers of the previous frame and j layers of the current frame were received by the decoder. Whenever $i \leq j$, the decoder cannot conceal lost layers of the current frame from the previous frame; therefore $d_{ij} = d_j$ when $i \leq j$. We denote by *distortion matrix*, matrix $[d_{ij}]_{0 \leq i,j \leq L}$. Table 6.1 summarizes the notations used in this chapter.

In our system, we suppose that the sender knows the distortion matrix $[d_{ij}]$ of the current video segment. When streaming stored video, the distortion matrix can be computed off–line. It can be stored at the sender, together with the video file. When streaming live video, the sender needs to estimate the value of the distortion matrix before starting the encoding and transmission of the current video segment. This estimate can be based on the previous video segments that have been already encoded and sent to the receivers. Since in most applications of live video streaming, such as streaming of sporting events or videoconferences, the consecutive video segments usually have recurrent or similar visual characteristics, we expect that the distortion matrix of an upcoming segment can be estimated sufficiently accurately.

6.3 Experimental Setup

In order to illustrate our results, we use MPEG–4 FGS videos. As we mentioned in Section 2.2.4, Fine Granularity Scalability has been specifically standardized for transmission of video over the best–effort Internet [8]. We suppose that the FGS enhancement layer has been divided into L layers for the current video segment. Recall that there is no motion compensation in the MPEG–4 FGS enhancement layer,

[1]Note that our system does not allow for retransmission of lost enhancement layer packets. This is a reasonable assumption for live streaming. It is also reasonable for stored video systems with short playback delays and high VCR–like interactivity.

N	Number of frames in the video
L	Number of enhancement layers
U	Number of source packets per layer
$rate_{avg}(\sigma)$	Average (normalized) transmission rate of an image under transmission policy σ
$dist_{avg}(\sigma)$	Average distortion of an image under transmission policy σ
$var_{avg}(\sigma)$	Average variation in distortion between two images under policy σ
$dist^*_{avg}$	Minimum average distortion of an image
ρ	Maximum target average transmission rate
γ	Maximum target average variation in distortion
A_n	Joint scheduling and error protection action for frame n
$\mathcal{A}$	Set of possible actions for a frame
$\mathcal{X}$	Set of possible receiver states
X_n	Receiver state for frame $n-1$
D_n	Distortion of frame n after decoding
d_{ij}	Distortion of frame n when $X_n = i$ and $X_{n+1} = j$

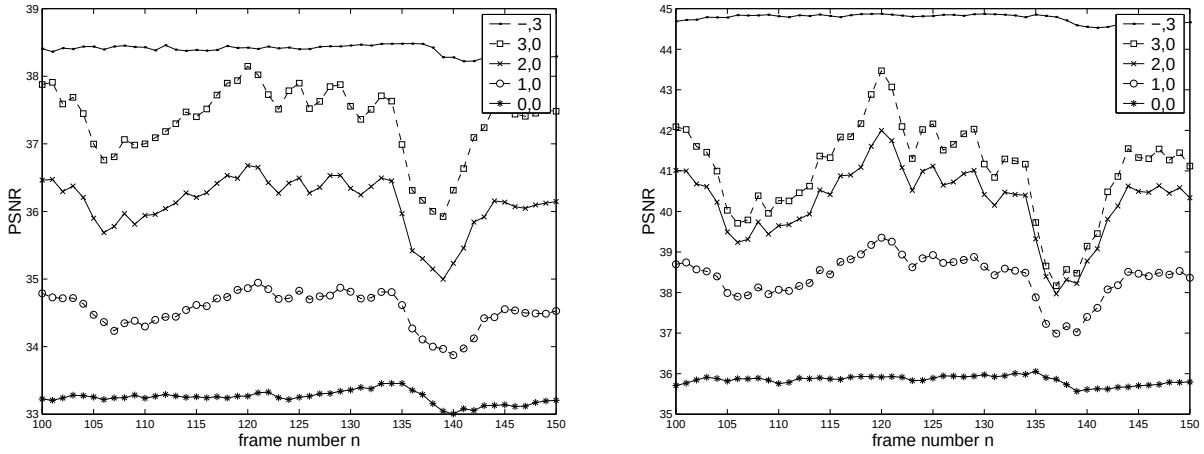
Table 6.1: Summary of notations for Chapter 6

which makes it highly resilient to transmission errors. Therefore, our unified framework is particularly well suited to the transmission of MPEG–4 FGS encoded video over the best–effort Internet. We apply our framework to the L enhancement layers extracted from the FGS enhancement layer[2].

In our experiments, we choose the simplest strategy for temporal error concealment, which consists in replacing the missing layers in the current frame by the corresponding layers in the previous frame. During our experiments, we have noticed that this strategy performs well for low motion video segments but poorly for segments with high motion. Video segments with a high amount of motion, such as *Coast-guard* or *Foreman*, would require an error concealment strategy which also compensates for motion. For example, [23] presents a scheme for error–concealment in the FGS enhancement layer, which uses, along with the layers from the previous frame, the motion information contained in the base layer of the current frame. Since we suppose that the base layer is transmitted without loss, such a strategy would be easily applicable to our system.

We present experiments with the low motion segment *Akiyo*. As in Chapter 5, we use the Microsoft

[2]Cutting the FGS enhancement layer into a fixed number of layers contrasts with the framework that we have introduced in the previous chapters, in which the server cuts the FGS enhancement layer at the granularity of bits. However, this makes our framework in this chapter applicable to both regular layered–encoded video and FGS–encoded video; the fine granularity property of FGS videos can still be exploited by choosing a high value for the number of layers L (clearly, the higher the number of layers extracted from the FGS enhancement layer bitstream, the finer the adaptation to bandwidth variations).



(a) low quality

(b) high quality

Figure 6.4: PSNR of frames 100 to 150 of *Akiyo* after EC for different values of $(X_n, X_{n+1}) = (y, z)$

MPEG–4 software encoder/decoder [86] with FGS functionality. We encode the video using two target qualities (low and high qualities), which can be used for different network capacities. Both low and high quality videos are encoded into a VBR base layer, with average bitrate of 36 kbps and 47 kbps, respectively, and a complete FGS enhancement layer with average bitrate of 900 kbps and 3 Mbps, respectively. For each quality version, we cut the FGS–EL into 3 layers of equal size ($L = 3$). The video segment is encoded into the CIF format (352×288 pixels), at a frame rate of 30 f/s. It contains $N = 300$ frames. In order to prevent too much fluctuations in quality between successive frames, the first base layer frame is encoded as an I–picture and all following frames as P–pictures[3].

Figure 6.4 shows, for a given frame n of the video, the quality in PSNR after error concealment when $X_n = y$ layers have been received for the previous frame $n-1$ and $X_{n+1} = z$ layers have been received for the current frame n. According to our simple temporal error concealment scheme, when more layers have been received for frame $n-1$ than for frame n, i.e. $y > z$, the decoder uses the additional $y - z$ enhancement layers from frame $n-1$ for decoding frame n. We verify on the figures that, when no layers have been received for frame n, i.e. $z = 0$, the PSNR of frame n after error concealment increases with the number of received layers for frame $n-1$, y. This shows that temporal error concealment is effective in increasing the quality of the rendered video. The increase in quality can be substantial. For example, for frame 120 of the low quality version of *Akiyo*, simple error concealment from the first enhancement layer of the previous frame can improve the quality of the current frame by almost 2 dB (when z = 0, the PSNR of frame 120 goes from 33.2 dB when $y = 0$ to 35 dB when $y = 1$). Note that the upper graph on

[3] As observed in Chapter 5, our VBR base layer encoder gives important variations in quality between the different types of frames.

Figure 6.5: Frame 140 of *Akiyo* (low quality) when (left) ($X_{140} = 0, X_{141} = 0$) – $PSNR = 33$ dB, (middle) ($X_{140} = 3, X_{141} = 0$) – $PSNR = 36.3$ dB, (right) ($X_{141} = 3$) – $PSNR = 38.3$ dB.

Figure 6.4 shows the maximum quality for a given frame n, which corresponds to the case when all the layers of frame n have been received ($z = 3$).

Figure 6.5 shows a zoomed–in part of decoded frame 140 after error concealment when no enhancement layer was received for frame 140 nor for frame 139 (left), no enhancement layer was received for frame 140 but all 3 layers of previous frame 139 were received (middle), and when all 3 layers of frame 140 were received (right). As we can see, the overall quality of frame 140 is better when all layers of the previous frame have been received (middle picture) than when no layer is available at the receiver for the previous frame (left picture). However, the quality is still lower than when all layers of frame 140 have been received and decoded (right picture).

We compute the average distortion over all frames of the video segment for all possible receiver states. After normalizing, we obtain the following distortion matrices for high and low quality versions of *Akiyo* :

$$[d_{ij}]_{\text{high}} = \begin{bmatrix} 1 & 0.34 & 0.09 & 0 \\ 0.53 & 0.34 & 0.09 & 0 \\ 0.31 & 0.18 & 0.09 & 0 \\ 0.25 & 0.14 & 0.06 & 0 \end{bmatrix}, \qquad [d_{ij}]_{\text{low}} = \begin{bmatrix} 1 & 0.57 & 0.20 & 0 \\ 0.64 & 0.57 & 0.20 & 0 \\ 0.33 & 0.52 & 0.20 & 0 \\ 0.15 & 0.32 & 0.03 & 0 \end{bmatrix}. \tag{6.1}$$

Note from (6.1) that, for the low quality version, $d_{21} = 0.52 > d_{20} = 0.33$ and $d_{31} = 0.32 > d_{30} = 0.15$. This means that replacing all available layers from the current frame by the corresponding layers from the previous frame achieves a lower distortion (better quality) than using the first layer of the current frame and the subsequent layers of the previous frame. This is due to our simple temporal EC strategy. Since we did not implement any motion compensation for EC, the replacement of layers of the

current frame by layers of the previous frame create some visual impairments. These impairments are usually minor for low–motion video segments. However, for some frames that are significantly different from the previous frames, the resulting increase in distortion can be slightly higher than the decrease in distortion brought by error concealment. As shown in (6.1), this does not occur for the high quality version of the video.

6.4 Optimization with Perfect State Information

In this section we suppose that the sender can observe state X_n when choosing the action A_n. This implies a reliable feedback channel from the receiver to the sender, and a connection RTT that is less than one frame time. This assumption is reasonable for interactive video applications, such as videoconferencing, that require short end–to–end transmission delays. It is also reasonable for live streaming and stored video systems with short playback delays and high VCR–like interactivity.

We show that Problem 6.1 can be formulated as a constrained MDP, which can in turn be solved by linear programming [33, 58]. The problem is naturally formulated as a finite–horizon MDP with N steps, where N is the number of frames in a video segment. However, the computational effort associated with a finite–horizon MDP can be costly when N is large [11]. This may be a serious impediment for real–time senders. Therefore, we instead use infinite–horizon constrained MDPs. They have optimal stationary policies and have lower computational cost. The infinite–horizon assumption corresponds to considering infinite–length video segments ($N = \infty$). Throughout this study, the values $rate_{avg}(\sigma)$, $dist_{avg}(\sigma)$ and $var_{avg}(\sigma)$ will be long–run averages.

6.4.1 Analysis

We consider the Markov Decision Process $\{X_n, A_n, n = 0, \dots\}$. Recall that D_n denotes the distortion for frame n after decoder error concealment. We define the reward when the receiver is in state $X_n = i$ and action $A_n = \boldsymbol{a} = (a_1, \dots, a_L)$ is chosen as:

$$\begin{aligned} r(i, \boldsymbol{a}) &= -E[D_n | X_n = i, A_n = \boldsymbol{a}] \\ &= -\sum_{j=0}^{L} d_{ij} P(X_{n+1} = j | A_n = \boldsymbol{a}). \end{aligned} \tag{6.2}$$

The cost is defined as:

$$c(i, \boldsymbol{a}) = \frac{1}{U \cdot L} \sum_{l=1}^{L} a_l. \tag{6.3}$$

From these definitions, and given that $E[r(X_n, A_n)] = -E[D_n]$, Problem 6.1 can be rewritten as

finding an optimal policy $\boldsymbol{\sigma}^*$ which maximizes the long–run average reward:

$$\lim_{n\to\infty} \frac{1}{n} E_{\boldsymbol{\sigma}}[\sum_{m=1}^{n} r(X_m, A_m)] \quad \text{s.t.} \quad \lim_{n\to\infty} \frac{1}{n} E_{\boldsymbol{\sigma}}[\sum_{m=1}^{n} c(X_m, A_m)] \leq \rho, \tag{6.4}$$

which falls into the general theory of constrained MDPs.

For a given layer, we denote by $q(u)$ the probability that the layer is successfully transmitted to the receiver, when $u \in \{0, U, \ldots, 2U-1\}$ is the total number of packets that have been sent for this layer. $q(u)$ is computed as the probability to transmit successfully at least U packets out of the u packets sent for the layer. Assuming that the transmission channel is a packet erasure channel with success probability q, we have:

$$q(u) = \begin{cases} \sum_{i=0}^{u-U} \binom{u}{U+i} q^{U+i}(1-q)^{u-U-i} & \text{for } u \geq U \\ 0 & \text{for } u = 0. \end{cases} \tag{6.5}$$

The reward can be expressed as:

$$r(i, \boldsymbol{a}) = -\sum_{j=0}^{L-1} d_{ij} q(a_1) \cdots q(a_j)(1 - q(a_{j+1})) \tag{6.6}$$

(because we took the convention that $d_{iL} = d_L = 0$). Recall that, in our model, a given layer is useful at the decoder only if all U source packets of the layer are available.

For a randomized stationary policy $\boldsymbol{\sigma}$, let $\sigma_{i\boldsymbol{a}} = P_{\boldsymbol{\sigma}}(A_n = \boldsymbol{a} | X_n = i)$. We denote by $P_{i\boldsymbol{a}j} = P(X_{m+1} = j | X_m = i, A_m = \boldsymbol{a})$ for the law of motion of the MDP. It is given by:

$$P_{i\boldsymbol{a}j} = \begin{cases} q(a_1) \cdots q(a_j)(1 - q(a_{j+1})) & \text{when } j < L \\ q(a_1) \cdots q(a_L) & \text{otherwise.} \end{cases} \tag{6.7}$$

This MDP is clearly a unichain MDP[4]. It therefore follows that the optimal policy for the constrained MDP is a randomized stationary policy. Furthermore, randomization occurs in at most one state [111]. An optimal stationary policy $\boldsymbol{\sigma}^*$ may be obtained from the following procedure:

Step 1. Find an optimal solution $z^* = \{z^*_{i\boldsymbol{a}}, (i, \boldsymbol{a}) \in \mathcal{X} \times \mathcal{A}\}$ to the linear program (LP):

$$\max \sum_{i\in\mathcal{X}} \sum_{\boldsymbol{a}\in\mathcal{A}} r(i, \boldsymbol{a}) z_{i\boldsymbol{a}} \quad \text{s.t.} \quad \begin{cases} \sum_{i\in\mathcal{X}} \sum_{\boldsymbol{a}\in\mathcal{A}} c(i, \boldsymbol{a}) z_{i\boldsymbol{a}} \leq \rho, \\ \sum_{i\in\mathcal{X}} \sum_{\boldsymbol{a}\in\mathcal{A}} (\delta_{ij} - P_{i\boldsymbol{a}j}) z_{i\boldsymbol{a}} = 0, \text{ for all } j \in \mathcal{X}, \\ \sum_{i\in\mathcal{X}} \sum_{\boldsymbol{a}\in\mathcal{A}} z_{i\boldsymbol{a}} = 1, \\ z_{i\boldsymbol{a}} \geq 0, \text{ for all } i \in \mathcal{L}, \boldsymbol{a} \in \mathcal{A}, \end{cases} \tag{6.8}$$

[4]An MDP is said *unichain* if the Markov Chain induced by any pure (stationary and non–randomized) policy has one recurrent class and a (perhaps empty) set of transient states [58, 111]. For a lossy channel ($q < 1$), Markov Chains induced by our MDP have a unique recurrent class containing state $X = 0$, which is accessible from any other state.

with $\delta_{ij} = 1$ if $i = j$; otherwise $\delta_{ij} = 0$.

Let $\mathcal{X}^* := \{i \in \mathcal{X} : z^*_{i\boldsymbol{a}} > 0 \text{ for some } \boldsymbol{a} \in \mathcal{A}\}$.

Step 2. Determine an optimal policy $\boldsymbol{\sigma}^*$ as follows:

$$\begin{cases} \text{for } i \in \mathcal{X}^*, & \sigma^*_{i\boldsymbol{a}} = \frac{z^*_{i\boldsymbol{a}}}{\sum_{\boldsymbol{a}\in\mathcal{A}} z^*_{i\boldsymbol{a}}} \\ \text{for } i \notin \mathcal{X}^*, & \sigma^*_{i\boldsymbol{a}} = 1 \text{ for some arbitrary } \boldsymbol{a} \in \mathcal{A}. \end{cases} \tag{6.9}$$

Note that there are several algorithms to solve LPs. The most popular is the simplex algorithm (Dantzig, 1947). It has exponential worst–case complexity, but requires a small number of iterations in practice. There are other more elaborate algorithms which have polynomial complexity, such as the projective algorithm by Karmarkar [60].

6.4.2 Case of 1 Layer Video with No Error Protection

As an example, first consider the particular case with 1 layer ($L = 1$), no error protection and $U = 1$ (i.e., 1 packet per layer). In this situation, the transmission action consists in deciding for each frame whether to send the single layer or send nothing at all. For this special case, we can actually derive a closed–form expression for the optimal policy (thereby circumventing linear programming). After analysis (see Appendix), the optimal transmission policy can be expressed as:

$$\sigma^*_{01} = \rho/(1-\rho q), \quad \sigma^*_{11} = 0 \qquad \text{if } \rho \le 1/(1+q); \tag{6.10}$$

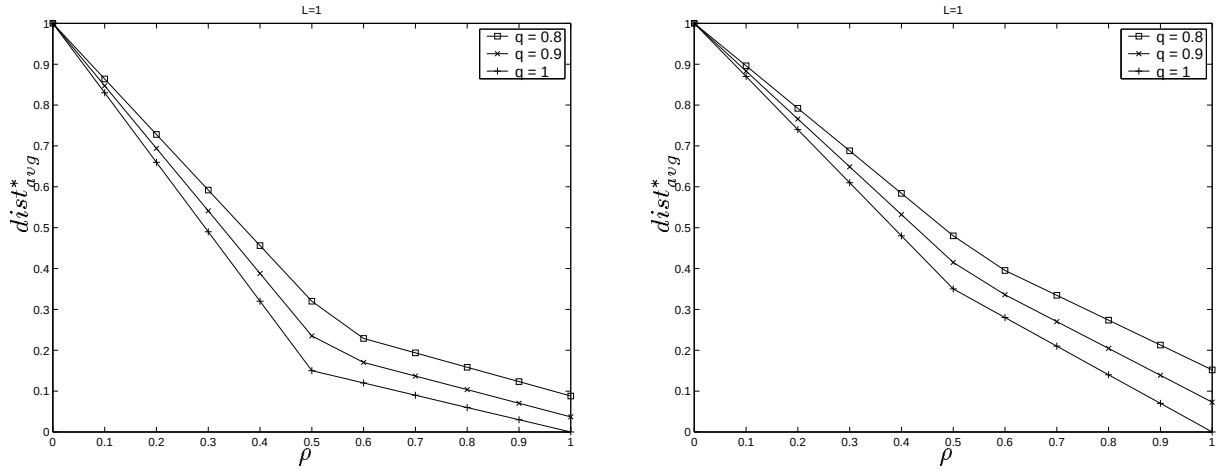
$$\sigma^*_{01} = 1, \qquad \sigma^*_{11} = 1 + (\rho - 1)/(\rho q) \quad \text{otherwise.} \tag{6.11}$$

The optimal average transmission rate and distortion are given by:

$$rate^*_{avg} = \rho \tag{6.12}$$

$$dist^*_{avg} = \begin{cases} 1 - \rho q(2 - d_{10}) & \text{if } \rho \le 1/(1+q), \\ (1-\rho q)(1 - q(1-d_{10})) & \text{otherwise.} \end{cases} \tag{6.13}$$

In Figure 6.6, we plot the minimum average distortion $dist^*_{avg}$ as a function of the maximum average transmission rate ρ, for selected values of the channel success rate q and two different values of d_{10} (normalized distortion when replacing the layer of a frame by the layer from the previous frame). For a given d_{10}, we observe that the difference between the values of $dist^*_{avg}$ for different channel success rates increases with ρ. Indeed, for low values of ρ, optimal policies are likely to send very few frames ($\sigma^*_{01} \approx 0$ and $\sigma^*_{11} = 0$ in (6.10)), so channel losses do not have much effect; for higher values of ρ, optimal policies send a large number of frames ($\sigma^*_{01} = 1$ and $\sigma^*_{11} \approx 1$ in (6.11)), so the value of the



(a) $d_{10} = 0.3$ (b) $d_{10} = 0.7$

Figure 6.6: Minimum average distortion for the case of 1 layer video

channel success rate q has higher importance. Comparing Figures 6.6(a) and 6.6(b), we verify that for a given q the minimum achievable distortion decreases with d_{10}; a low d_{10} corresponds to a highly efficient temporal error concealement, which is usually obtained with a video with high temporal redundancy.

Throughout the remaining of this chapter, we present simulations with the MPEG–4 FGS videos *Akiyo* described in Section 6.3, i.e., with $L = 3$.

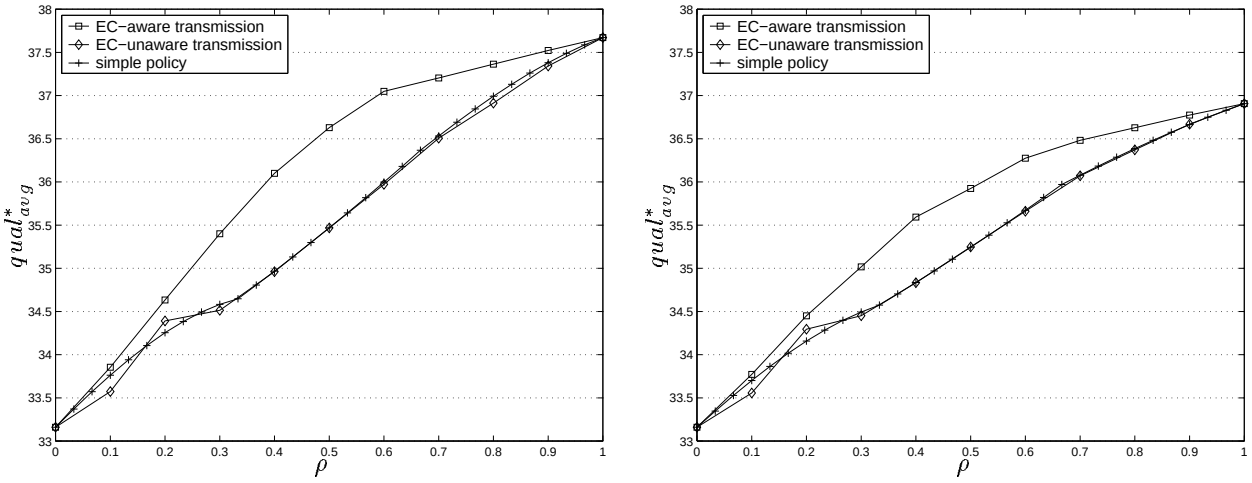
6.4.3 Comparison between EC–aware and EC–unaware Optimal Policies

We compare the scheduling and error protection optimization with accounting for error concealment, to the optimization without accounting for error concealment:

- **EC–unaware transmission:** The sender determines and employs the optimal transmission policy, which is obtained without accounting for error concealment at the receiver. Nevertheless, the receiver applies error concealment before rendering the video.

- **EC–aware transmission:** The sender determines and employs the optimal transmission policy, which accounts for error concealment. The receiver applies error concealment before rendering the video.

It is important to notice that both schemes employ error concealment at the decoder, so that when comparing the rendered video quality of the two schemes, we are indeed making a fair comparison. Let $qual^*_{avg}$ denote the maximum quality of the video, i.e., the quality given by the optimal transmission policy.



(a) $q = 0.9$ (b) $q = 0.8$

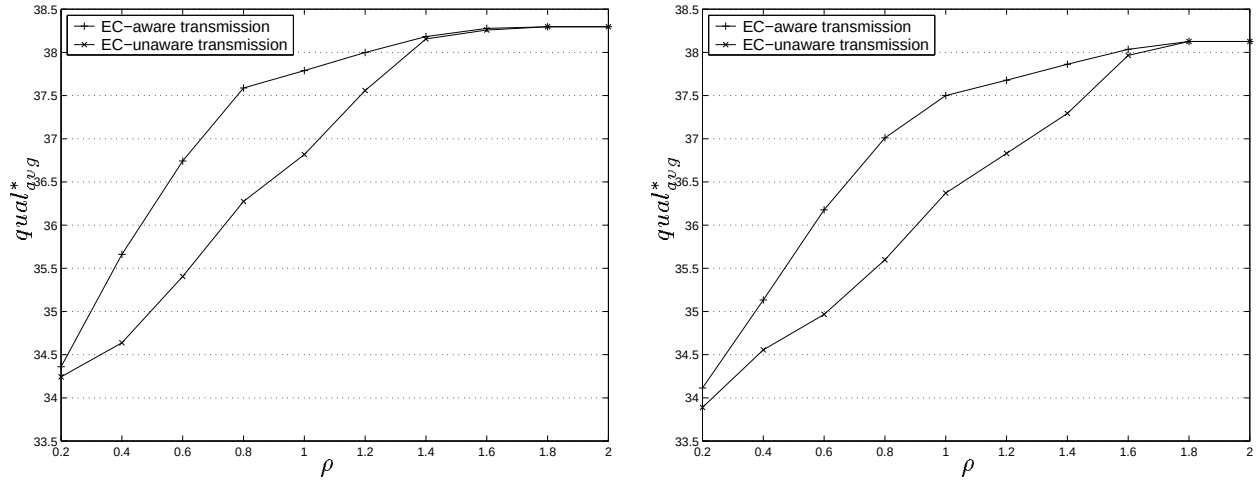
Figure 6.7: Comparison between EC–aware, EC–unaware and simple optimal policies without FEC for *Akiyo* (low quality)

We first suppose no error protection (no FEC) and $U = 1$. We study the comparison between optimal dynamic transmission policies (EC–aware and EC–unaware) as given by our optimization framework, and some simple static scheduling policies, i.e., that do not depend on the receiver state. We denote by *simple policy* (l, p), the policy that sends alternatively l layers, with probability p and $l - 1$ layers with probability $1 - p$. Simple policy $(l, 1)$ corresponds to a static non–randomized policy that sends l layers for all frames of the video. Note that a simple policy that is optimal for Problem 6.1 among all simple policies, should verify:

$$rate_{avg} = l \cdot p + (l - 1) \cdot (1 - p) = \rho. \tag{6.14}$$

Figure 6.7(a) and Figure 6.7(b) show, for Problem 6.1, the value of $qual^*_{avg}$ in PSNR as a function of the target transmission rate ρ, for EC–unaware and EC–aware transmission optimal policies, as well as optimal simple policies. We used the low quality version of *Akiyo*. We consider channel success rates of $q = 0.9$ and $q = 0.8$, which correspond to typical values in today's Internet (the packet loss rate is usually between 5% and 20%). We see on both figures that the maximum quality achieved by EC–unaware optimal policies and optimal simple policies is similar, while EC–aware optimal policies achieve the best quality for all target rates. The gain brought by optimizing scheduling with considering error concealment is up to 1.1 dB[5] for a channel with a high success rate ($q = 0.9$). These results indicate that optimizing the transmission without considering decoder error concealment in the optimization process

[5]We expect the difference in quality between EC–unaware and EC–aware optimal policies to be even higher with error concealment schemes that compensate for motion, notably by using the base layer information.



(a) $q = 0.9$ (b) $q = 0.8$

Figure 6.8: Comparison between EC–aware and EC–unaware optimal policies with FEC for *Akiyo* (low quality)

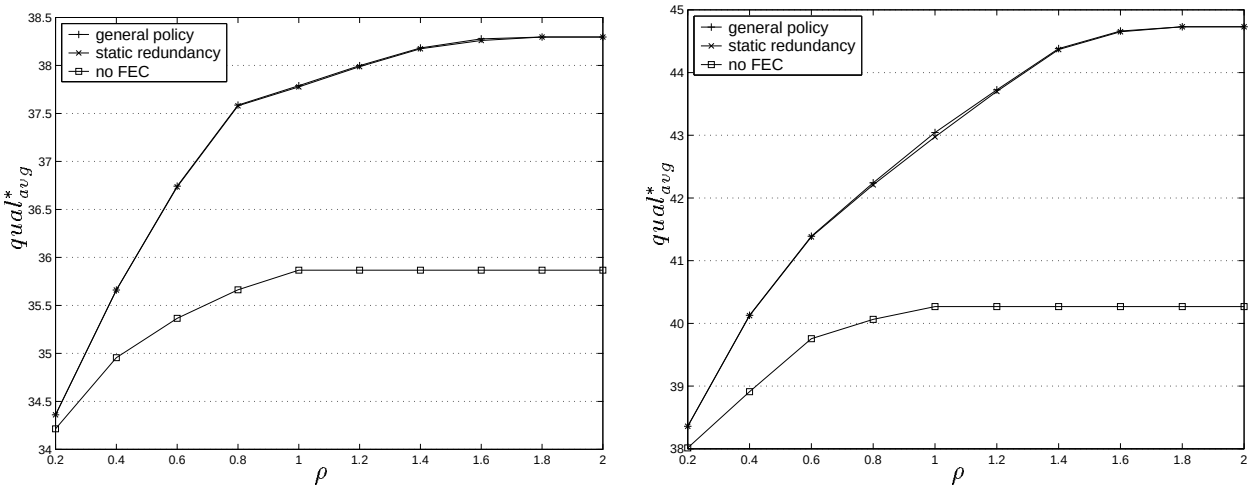
can lead to an end–to–end performance that is no much better than very simple static policies.

We now consider joint scheduling and error protection through FEC. We suppose that $U = 4$. Figure 6.8(a) and Figure 6.8(b) show curves similar to Figures 6.7 but with FEC. We verify on both figures that the maximum quality achieved by EC–aware optimal policies is significantly higher than that of EC–unaware optimal policies (for both values of q the difference in quality is up to 1.5 dB). This confirms the need to account for decoder error concealment during joint scheduling and error protection optimization. Simulations with the high quality version of *Akiyo* also give differences in quality that exceed 1 dB. Note that for high values of ρ both schemes achieve the same performance. This corresponds to the extreme case when the average bandwidth of the connection is much higher than the source bitrate of the video ($\rho >> 1$). In this situation, both EC–aware and EC–unaware optimal policies transmit all layers with additional FEC packets, thereby achieving maximum performance.

Throughout the rest of this study, we only consider EC–aware transmission policies.

6.4.4 Comparison between Dynamic and Static FEC

We also investigate solutions of Problem 6.1 for the particular case when the amount of FEC codes added to each layer is constant throughout the video sequence. For this case, let $0 \leq f_l \leq U - 1$ denote the number of FEC packets added to layer l for all frames of the current video sequence. The transmission decision to take for frame n is still expressed as $A_n = (a_1, \ldots, a_L)$, but now with $a_l \in \{0, U + f_l\}$. We denote the corresponding transmission policies by *static redundancy policies* (in contrast to dynamic



(a) low quality

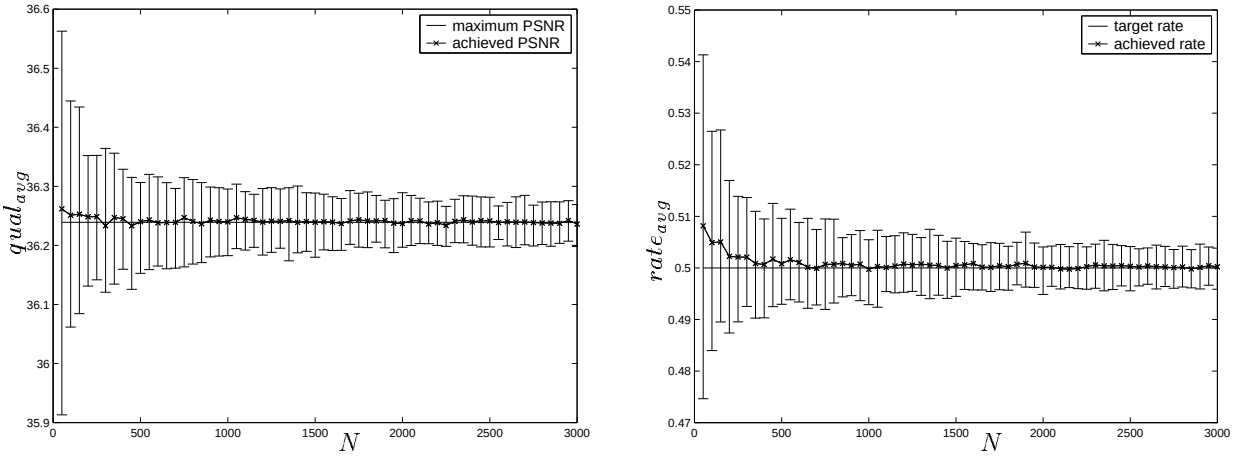
(b) high quality

Figure 6.9: Comparison between general and static redundancy optimal policies for *Akiyo*

redundancy policies in the general case). Optimal static redundancy policies can be found by solving LP (6.8) with the new set of possible actions $\mathcal{A}_{static} \subset \mathcal{A}$, for all possible sets $(f_1, \ldots, f_L)$ (brute–force algorithm).

Figure 6.9 shows the maximum average quality $qual^*_{avg}$ for the low and high quality versions of *Akiyo*, as a function of ρ, for a transmission channel with $q = 0.9$. We first compare optimal general policies with optimal static redundancy policies. We can see that, for both quality versions of the video, the maximum quality for the optimal general policy and for the optimal static redundancy policy is almost the same for all ρ. (We noticed that both optimal policies are indeed identical for most values of ρ.) This indicates that we can restrict our optimization problem to static redundancy policies. Simulations for other values of q lead to the same conclusion.

We compare optimal general and static redundancy policies with FEC to optimal policies without FEC. We see that the gain in quality achieved with FEC can be substantial. When $q = 0.9$, for both versions of the video, the difference in quality achieved by the optimal policy with FEC and without FEC is more than 1 dB for all values of $\rho \geq 0.6$. Note that when $\rho \geq 1$, the maximum quality achieved by the optimal policy without FEC stays constant, while the quality achieved with FEC still increases with ρ. Indeed, when $\rho \geq 1$, the channel can accommodate the transmission of all video source packets plus some additional packets. So, the optimal policy without FEC can only send all source packets, whereas the optimal policy with FEC can send additional FEC packets, which enhances the quality of the rendered video.



(a) average quality in PSNR

(b) transmission rate

Figure 6.10: Simulations for video segments containing up to 3000 frames.

6.4.5 Performance of Infinite–Horizon Optimization

We study the performance of our EC–aware optimal transmission policies, obtained by our optimization framework over an infinite–horizon, in the practical case when the number of frames of the video sequence, N, is finite. We used the average distortion matrix $[d_{ij}]_{low}$ given in (6.1) for all N frames of the video. We show simulations for a target transmission rate of $\rho = 0.5$, over a channel with success rate $q = 0.9$. We averaged our results over 100 channel realizations.

Figure 6.10(a) and Figure 6.10(b) plot the achieved average quality and average transmission rate, respectively, as a function of the number of frames of the video (up to 3000 frames). We plot confidence intervals that represent 95% of the channel runs. As we can see on both figures, as the number of frames increases, the achieved transmission rate and quality averaged over all channel realizations converge towards the target rate ρ and the maximum quality $qual^*_{avg}$, respectively. For a 50 frame segment, the convergence errors are only of 0.03 dB for the quality and 2% for the transmission rate. However, the confidence intervals can be large for segments with a low number of frames: for a 50 frame segment, the transmission rate achieved for a given channel realization can be up to 8% higher than ρ, and the quality up to 0.35 dB lower than the target quality. For a 500 frame segment, these errors come down to 3% and 0.1 dB, respectively.

Since, in common videos, most homogeneous segments are composed of tens to thousands of frames (homogeneous segments usually correspond to video scenes as we mentioned in Chapter 5), we expect that our optimization framework over an infinite–horizon will achieve a good operational performance in most cases. For video segments composed of a few frames only, it may be more appropriate to use finite–horizon linear programming in order to find optimal policies for each separate frame, as mentioned

	quality in PSNR			transmission rate		
	target	avg.	min.	target	avg.	max.
$\rho = 0.33$	35.69	35.71	35.48	1.00	1.01	1.05
$\rho = 0.50$	36.63	36.64	36.36	1.50	1.51	1.62
$\rho = 0.66$	37.15	37.14	36.93	2.00	2.01	2.12

Table 6.2: Simulations for *Akiyo* (low quality)

at the beginning of Section 6.4.

Finally, Table 6.2 presents the results for the 300 frames of the original low quality *Akiyo* sequence, without FEC and for $U = 1$. We show, for different values of the average target transmission rate ρ, the achieved quality in PSNR and the transmission rate, averaged over all channel realizations. We also show the minimum quality and maximum transmission rate, which are achieved by one channel realization. As we can see, the average achieved values are very close to the target values in all cases. This shows that applying our infinite–horizon optimization framework to finite–length videos gives very good performance. In this example, the difference between the minimum achieved PSNR and the target quality is always lower than 0.3 dB. The difference between the maximum achieved transmission rate and the target transmission rate is always lower than 8%.

6.5 Additional Quality Constraint

In Problem 6.2, we added a new quality constraint to our optimization framework. Specifically, besides minimizing the average distortion, $dist_{avg}$, the optimal transmission policy should also maintain an average variation in distortion between consecutive images, var_{avg}, below a maximum sustainable value γ. As in Problem 6.1, we consider that the video has infinite length. For a given transmission policy σ, $var_{avg}(\sigma)$ is the long–run average defined by:

$$var_{avg}(\sigma) := \lim_{n\to\infty} \frac{1}{n-1} E_\sigma[\sum_{i=2}^{n} |D_i - D_{i-1}|] \tag{6.15}$$

As for Problem 6.1, we analyze Problem 6.2 with a Markov Decision Process over an infinite–horizon. We suppose that the sender can observe the state of the receiver as in Section 6.4. The expected average distortion of a given frame n depends only on action A_n and on the state for the previous frame $n-1$, i.e., X_n. However, the expected average variation in distortion for frame n depends also on the value of the state for frame $n-2$, i.e., X_{n-1}. Indeed, from (6.15), we have $var_{avg}(\sigma) = E_\sigma[|D_n - D_{n-1}|]$, where D_{n-1} is the distortion for frame $n-1$, which depends on the number of layers that have been received for frames $n-1$ and $n-2$, i.e, X_n and X_{n-1} respectively.

We consider the MDP $\{X_{n-1}, X_n, A_n, n = 0, \dots\}$, where $\{X_{n-1}, X_n\}$ and $\{A_n\}$ are the state and

action processes, respectively. We define the reward and cost functions, when the receiver is in state $(X_{n-1} = i, X_n = j)$ and action $A_n = \boldsymbol{a}$ is taken, as:

$$r(i, j, \boldsymbol{a}) = -E[D_n | X_n = j, A_n = \boldsymbol{a}], \tag{6.16}$$

$$c(i, j, \boldsymbol{a}) = \frac{1}{U \cdot L} \sum_{l=1}^{L} a_l, \tag{6.17}$$

$$c^{'}(i, j, \boldsymbol{a}) = E[|D_n - D_{n-1}||X_{n-1} = i, X_n = j, A_n = \boldsymbol{a}]. \tag{6.18}$$

From these definitions, Problem 6.2 can be rewritten as finding an optimal policy $\boldsymbol{\sigma}^*$ which maximizes the long–run average reward:

$$\lim_{n\to\infty} \frac{1}{n} E_{\boldsymbol{\sigma}}[\sum_{m=1}^{n} r(X_{m-1}, X_m, A_m)] \quad \text{s.t.} \begin{cases} \lim_{n\to\infty} \frac{1}{n} E_{\boldsymbol{\sigma}}[\sum_{m=1}^{n} c(X_{m-1}, X_m, A_m)] \leq \rho, \\ \lim_{n\to\infty} \frac{1}{n} E_{\boldsymbol{\sigma}}[\sum_{m=1}^{n} c^{'}(X_{m-1}, X_m, A_m)] \leq \gamma, \end{cases} \tag{6.19}$$

which falls into the general theory of Markov Decision Processes with multiple constraints. The optimal policy can be found from a linear program similar to (6.8), but with a higher number of variables and one additional constraint.

Note that the additional cost is expressed as follows:

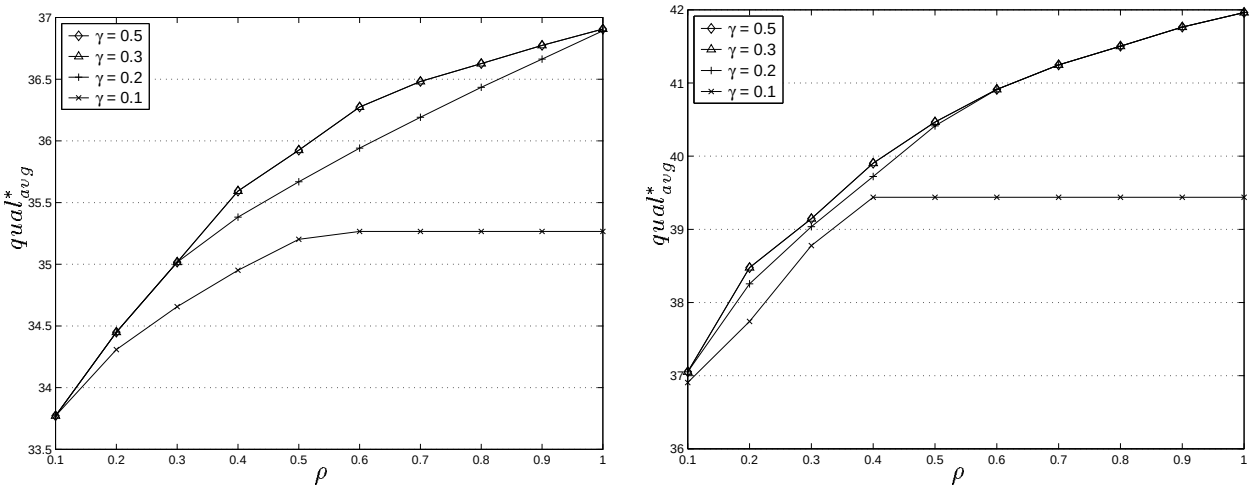
$$\begin{aligned} c^{'}(i, j, \boldsymbol{a}) = &\sum_{k=0}^{L-1} |d_{jk} - d_{ij}| q(a_1) \cdots q(a_k)(1 - q(a_{k+1})) \\ &+ |d_{jL} - d_{ij}| q(a_1) \cdots q(a_L) \end{aligned} \tag{6.20}$$

Figure 6.11 shows the optimal quality achieved as a function of ρ, for different values of the maximum variation in distortion γ. We consider optimal EC–aware transmission policies without FEC, with $U = 1$, over a channel with $q = 0.8$. As we can see, the constraint on the variation in distortion comes with a penalty in average quality, for $\gamma \leq 0.2$. For higher values of γ, the quality is the same as without the constraint on the variation in distortion because we have reached the variation in distortion of the optimal transmission policies for Problem 6.1.

6.6 Optimization with Imperfect State Information

In this section we suppose that the sender cannot, in general, observe X_n when choosing the action A_n. In this case MDP $\{X_n, A_n\}$ is a Partially–Observable MDP (POMDP), i.e., an MDP with imperfect state information. POMDPs are notoriously difficult[6], but our POMDP is tractable due to its special structure.

[6]Solutions of unconstrained POMDPs with delayed state information have been obtained [34]. However, to our knowledge, there is no theory for solving constrained POMDPs.



(a) low quality

(b) high quality

Figure 6.11: Maximum quality achieved for different values of the maximum quality variability for *Akiyo*

We can suppose that the sender observes the state of a previous frame $n-\tau_n$ for which it has received a feedback, i.e., we suppose that the sender can observe $X_{n-\tau_n+1}$ when choosing the action A_n ($\tau_n \geq 1$). This corresponds to a RTT of less than τ_n frame time for transmission of frame $n-\tau_n$.

When the state of reception for frame $n-1$, X_n, is not immediately available ($\tau_n > 1$), the transmitter can still take the decision for frame n, i.e. A_n, from the history of past state observations and past actions. Let $H_n = (X_{n-\tau_n+1}, A_{n-\tau_n+1}, \ldots, A_{n-1})$ denote the state and action history when the transmitter takes action A_n. Consider the case when $\tau_n = \tau > 1$ for all n, i.e., the maximum feedback delay for all frames is constant. Now, $\{H_n, A_n\}$ is a MDP with perfect state information. We define the associated reward and cost, when the receiver state is $H_n = \boldsymbol{h} = (x_{n-\tau+1}, \boldsymbol{a}_{n-\tau+1}, \ldots, \boldsymbol{a}_{n-1})$ and action $A_n = \boldsymbol{a}$ is chosen as:

$$r(\boldsymbol{h}, \boldsymbol{a}) = -E[D_n | H_n = \boldsymbol{h}, A_n = \boldsymbol{a}] \tag{6.21}$$

$$= -E[D_n | A_{n-1} = \boldsymbol{a}_{n-1}, A_n = \boldsymbol{a}], \tag{6.22}$$

$$c(\boldsymbol{h}, \boldsymbol{a}) = \frac{1}{L \cdot U} \sum_{l=1}^{L} a_l. \tag{6.23}$$

The reward only depends on A_{n-1} and A_n, because the distortion of frame n only depends on A_n and X_{n-1}, which in turn only depends on A_{n-1}. Subsequently, our MDP is equivalent as MDP $\{A_{n-1}, A_n\}$. (This is because, in our framework, we consider temporal error concealment from the previous frame only.) Therefore, our optimization framework does neither depend on the maximum feedback delay, τ, nor on the reception of the feedback. It is particularly well suited to applications where a feedback

channel cannot be used, for example to applications that have strict delay requirements, such as video-conferencing.

When $A_{n-1} = \boldsymbol{a}$ and $A_n = \boldsymbol{a}'$, the reward and cost of MDP $\{A_{n-1}, A_n\}$ for a packet erasure channel are given by:

$$r(\boldsymbol{a}, \boldsymbol{a}') = -\sum_{i=0}^{L}\sum_{j=0}^{L} d_{ij} \cdot P(X_n = i | A_{n-1} = \boldsymbol{a}) \cdot P(X_{n+1} = j | A_n = \boldsymbol{a}'), \tag{6.24}$$

$$c(\boldsymbol{a}, \boldsymbol{a}') = \frac{1}{L \cdot U}\sum_{l=1}^{L} a_l'. \tag{6.25}$$

Figure 6.12(a), (b), (c) and (d) show, for both quality versions of *Akiyo* and different values of q, the difference in performance between a channel model with perfect state information (immediate feedback) and imperfect state information (delayed feedback), with and without FEC. On the figures, we see that the difference in quality for the optimal policies with FEC is small (always less than 0.2 dB). Without FEC, the difference in quality between both channel models is larger. For $q = 0.9$ it is around 0.5 dB for most values of ρ. Indeed, adding FEC increases the effective packet transmission success rate, which, in turn, increases the knowledge of the sender about the actual receiver state.

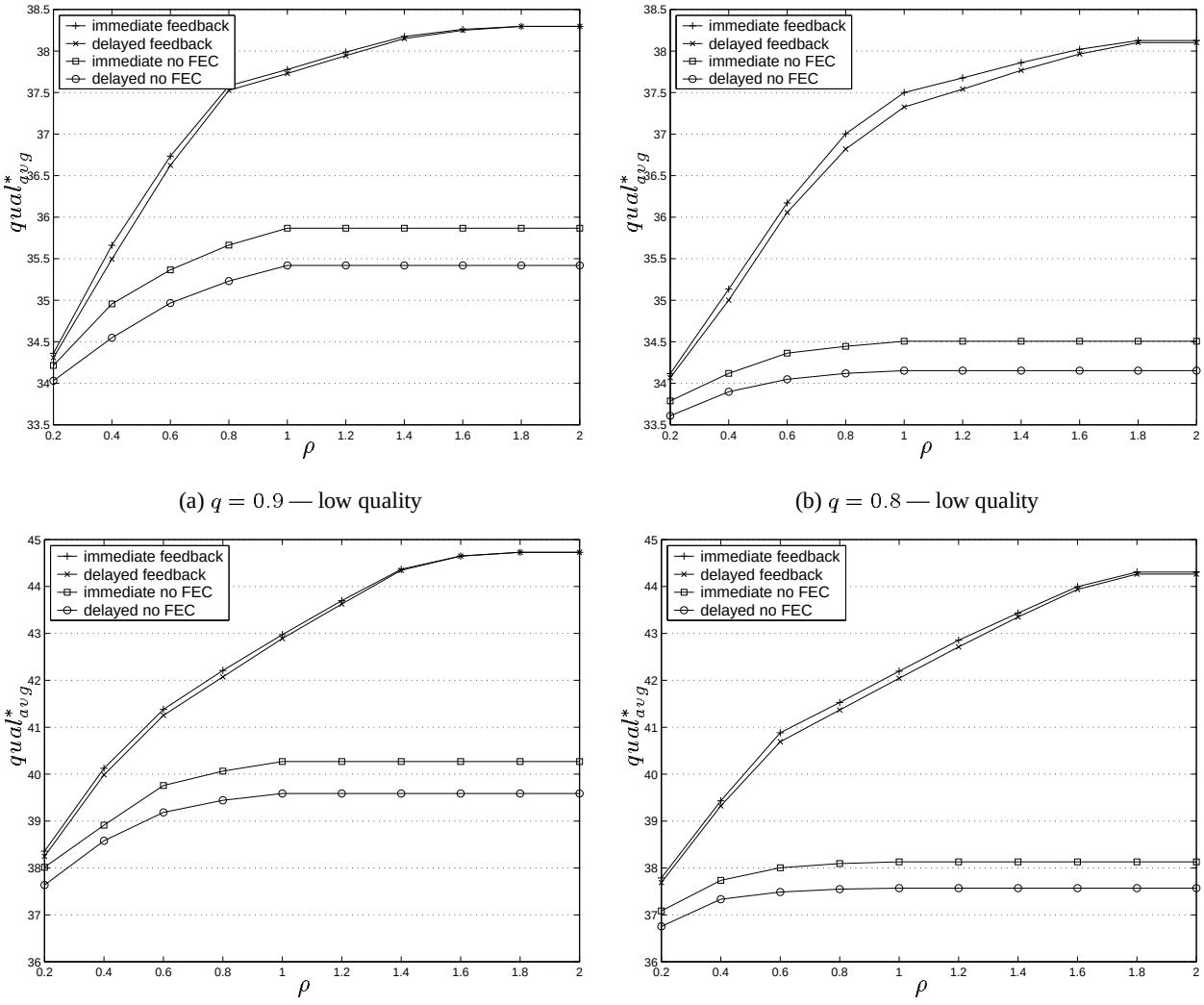
These results indicate that our framework for joint scheduling and error control optimization can achieve very good performance, even in the case when the receiver state can not be fully observed when making new decisions. This corresponds to the usual situation of video streaming over the best–effort Internet, where the feedback channel is unreliable and the connection has an average RTT which is higher than the video frame rate.

6.7 Conclusions

We have proposed a unified optimization framework that combines packet scheduling, error control and decoder error concealment. We used results on constrained Markov Decision Processes over an infinite–horizon, to compute optimal transmission policies for a wide range of quality metrics.

We have analyzed the problem of minimizing the average distortion under a limited transmission rate. Our analysis leads to a low–complexity algorithm, based on Linear Programming. We have evaluated the performance of our optimization framework in the context of streaming MPEG–4 FGS videos.

We first considered a packet–erasure channel with perfect receiver state information. We showed the potential quality gains brought by EC–aware transmission optimization over EC–unaware optimization. Our simulations indicate that complex scheduling optimization procedures that do not consider decoder error concealment in the optimization process can achieve results that are significantly lower than optimal results. We have seen, through numerical simulations, that our infinite–horizon optimization framework gives good performance for finite–length video segments composed of hundreds of video frames. We



(a) $q = 0.9$ — low quality

(b) $q = 0.8$ — low quality

(c) $q = 0.9$ — high quality

(d) $q = 0.8$ — high quality

Figure 6.12: Comparison between channels with perfect and imperfect state information for *Akiyo*

showed that our framework allows to accommodate additional quality metrics other than the average distortion, such as the variation in distortion between consecutive images.

Finally, we have shown that our optimization problem could be limited to static redundancy transmission policies, and that our methodology can achieve good performance in the case when the receiver state information is not available at the sender.

Chapter 7

Conclusions

In this dissertation, we have formulated new frameworks and solved new optimization problems for network and content adaptive streaming of layered–encoded video (regular and FGS). We have analyzed optimal transmission policies and evaluated their performance based on simulations with network traces and real videos.

7.1 Summary

After giving an overview of Internet video streaming in Chapter 2, in Chapter 3 we have designed equivalent adaptive streaming policies for adding/dropping layers and for switching among versions, in the context of reliable TCP–friendly transmission. Our simulations showed that, for low values of the layering overhead, the enhanced flexibility provided by layering over versions (i.e., immediate enhancement) allows adding/dropping layers to achieve similar performance to switching versions in terms of high quality viewing time, but at the expense of higher fluctuations in quality. For moderate values of the layering bitrate overhead (higher than 10%), switching versions reaches better performance than adding/dropping layers in all bandwidth situations.

In contrast with regular layered encoding, using FGS–encoded video allows the streaming application to adapt finely to changing network bandwidth. In Chapter 4 we have presented a new framework for adaptive–streaming of stored FGS–encoded video. We derived optimal transmission policies that simultaneously maximize a measure of bandwidth efficiency and minimize a measure of coding rate variability. We developed a real–time heuristic that is inspired from the optimal solution. Simulations with Internet traces showed that our heuristic yields near–optimal performance in a wide range of bandwidth scenarios. We have compared streaming over TCP with streaming over a reliable TFRC connection. Simulations showed that, while TFRC goodput is smoother than that of TCP, our adaptive streaming policies can reach similar performance over both protocols. This indicates that streaming stored FGS–encoded video

with sufficient client buffering does not require smooth–rate TCP–friendly algorithms to achieve good video quality. Finally, we have presented an implementation of our framework and our heuristic in an end–to–end system for streaming MPEG–4 FGS videos.

In Chapter 5 we have focused on the content characteristics of MPEG–4 FGS videos. We analyzed the rate–distortion traces of long videos, using performance metrics that capture the quality of the received and decoded video both at the level of individual video frames (images) as well as aggregations of images (GoP, scene, etc). Our analysis suggests to prioritize the cutting of the FGS enhancement layer bitstream close to the end of the bitplanes, and to take into consideration the different base layer frame types and the scene structure of the video. We have investigated rate–distortion optimized streaming at different image aggregation levels. Simulations from our traces have shown that aggregating successive frames for the optimal adjustment of the FGS enhancement layer rate reduces significantly the computational complexity of the optimization, at the cost of a small loss in overall quality. However, aggregating images arbitrarily tends to result in quality deterioration, compared to aggregating images on the basis of visual scenes. Therefore, we advocate streaming video at the granularity of visual scenes.

Finally, in Chapter 6 we have studied rate–distortion optimized streaming of layered video over lossy connections. We have proposed a unified optimization framework that combines packet scheduling, error control and decoder error concealment. We solved the problem of minimizing the average distortion under a limited transmission rate, using the theory of average–reward MDPs with infinite–horizon. We did simulations with MPEG–4 FGS videos. Considering a packet–erasure channel with perfect receiver state information, we showed that optimization procedures that do not consider decoder error concealment in the optimization process can achieve results that are significantly lower than truly optimal results. We demonstrated that our infinite–horizon optimization framework gives good performance for finite–length video segments composed of hundreds of video frames, and that it can be limited to static redundancy transmission policies. We extended our optimization problem by adding a constraint on the variation in distortion between consecutive images; this illustrates that our framework permits to accommodate additional quality metrics other than the average distortion. In the case when the receiver state information is not always available at the sender, we showed that our framework still achieves good performance; thus, our MDP approach is also suitable for networks with long end–to–end delays.

7.2 Areas of Future Work

We can continue the work presented in this dissertation in several directions both in the areas of network adaptation and content adaptation.

In the area of network adaptation, our frameworks could be first enriched by specifically accounting

for retransmissions of lost video packets in the optimization procedure. While there are many studies that propose mechanisms for delay–constrained retransmission of audio or video, optimal streaming with retransmission has not received significant attention (see Section 2.4.2). Modeling and solving of such problems is indeed particularly difficult, in part because of the correlation between losses and delays in the best–effort Internet. Network adaptation could also benefit from models of long–term variations of the TCP–friendly bandwidth given to an application. This would lead to a real–time streaming mechanism that performs closely to optimal policies obtained when the complete evolution of bandwidth is known a priori, such as in Chapter 4. Unlike voice traffic over the POTS[1], the Internet traffic cannot be easily modeled by Poisson distributions; therefore specific models need to be derived for the Internet. The availability of a QoS–enabled Internet, such as Diffserv or Intserv, would certainly be favorable to video streaming applications. In particular, more work is needed on how to optimally combine layered encoding with a packet marking strategy for streaming video over DiffServ. Finally, our frameworks and algorithms could be adapted to other configurations than the simple client–server configuration. For instance, in current CDNs (Content Distribution Networks) and peer–to–peer networks, different parts of the content can be served simultaneously from different servers. In this case, we should determine a global optimal transmission policy and we should find efficient ways to synchronize the transmission from each server.

There are other avenues for future work in the domain of content adaptation. First, as most rate–distortion optimized streaming algorithms presented in the literature, our algorithms would need to be assessed with objective quality metrics other than the average distortion. Unfortunately, as of today, there is no universal reliable objective metric for evaluating the quality of streamed videos, so extensive experiments with actual users are generally required. Quality metrics that account for the variability in image quality would be particularly relevant for our study. Our work would also benefit from finer temporal segmentation than shot–based segmentation; the segmentation of scenes is still an active research area. Finally, our schemes for content adaptation could be extended by complementing scene–based temporal segmentation with spatial segmentation of the video into objects. Image spatial segmentation on videos is another active research area in multimedia signal processing, which has been put forward by MPEG–4. Combining object–based scalability, layered encoding and scene–based segmentation would lead to an extended framework, in which the server could add/drop layers of individual objects, based on the semantic importance of each object in the scene, as well as the rate–distortion properties of each layer.

[1] Plain Old Telephone System

Appendix A:
Optimal EC–Aware Transmission of 1 Layer

In this appendix, we derive the closed–form expression of the optimal policy for Problem 6.1 that is given in Section 6.4.2. We have $L = 1$, $U = 1$ and no error protection. In this case $X_n, A_n \in \{0, 1\}$, and the distortion matrix has the following form:

$$[d_{ij}] = \begin{bmatrix} 1 & 0 \\ d_{10} & 0 \end{bmatrix} \tag{A.1}$$

Let $d = d_{10} \in [0, 1]$. When $U = 1$ (1 packet per layer), we simply have $q(0) = 0$ and $q(1) = q$. The reward and cost functions given in (6.6) and (6.3) are expressed as:

$$\begin{cases} r(0,0) = -1 \\ r(1,0) = -d \\ r(0,1) = -(q \times 0 + (1-q) \times 1) = -(1-q) \\ r(1,1) = -(q \times 0 + (1-q) \times d) = -d(1-q) \end{cases} \quad \text{and} \quad \begin{cases} c(0,0) = 0 \\ c(1,0) = 0 \\ c(0,1) = 1 \\ c(1,1) = 1 \end{cases} . \tag{A.2}$$

LP (6.8) can be written as:

$$\max \left\{ -(z_{00} + d \cdot z_{10} + (1-q) z_{01} + d(1-q) z_{11}) \right\} \tag{A.3}$$

$$\text{s.t.} \begin{cases} z_{01} + z_{11} \leq \rho, \\ (1 - P_{000}) z_{00} + (1 - P_{010}) z_{01} - P_{100} z_{10} - P_{110} z_{11} = 0 \quad (j = 0), \\ -P_{001} z_{00} - P_{011} z_{01} + (1 - P_{101}) z_{10} + (1 - P_{111}) z_{11} = 0 \ \ (j = 1), \\ z_{00} + z_{01} + z_{10} + z_{11} = 1, \\ z_{ia} \geq 0, \ \forall i, a \in \{0, 1\}. \end{cases} \tag{A.4}$$

After replacing the laws of motion P_{iaj} by their values from (6.7), we have:

$$\min\Big\{ z_{00} + d \cdot z_{10} + (1-q)z_{01} + d(1-q)z_{11} \Big\} \text{ s.t. } \begin{cases} z_{01} + z_{11} \leq \rho, \\ q \cdot z_{01} - z_{10} - (1-q)z_{11} = 0, \\ z_{00} + z_{01} + z_{10} + z_{11} = 1, \\ z_{ia} \geq 0 \text{ for } i, a \in \{0,1\}. \end{cases} \tag{A.5}$$

We replace z_{00} and z_{10} in the objective function by their expression from the constraint equations. This yields:

$$\min\Big\{ 1 - q(2-d)z_{01} - q \cdot z_{11} \Big\} \text{ s.t. } \begin{cases} z_{01} + z_{11} \leq \rho, \\ z_{10} = q \cdot z_{01} + (q-1)z_{11}, \\ z_{00} = 1 - (1+q)z_{01} - q \cdot z_{11}, \\ z_{ia} \geq 0 \text{ for } i, a \in \{0,1\}. \end{cases} \tag{A.6}$$

We pose $x = z_{01}$ and $y = z_{11}$, and obtain the following bi–dimensional LP:

$$\max\Big\{ (2-d)x + y \Big\} \text{ s.t. } \begin{cases} x + y \leq \rho, \\ q \cdot x + (q-1)y \geq 0, \\ (1+q)x + q \cdot y \leq 1, \\ x \geq 0,\ y \geq 0. \end{cases} \tag{A.7}$$

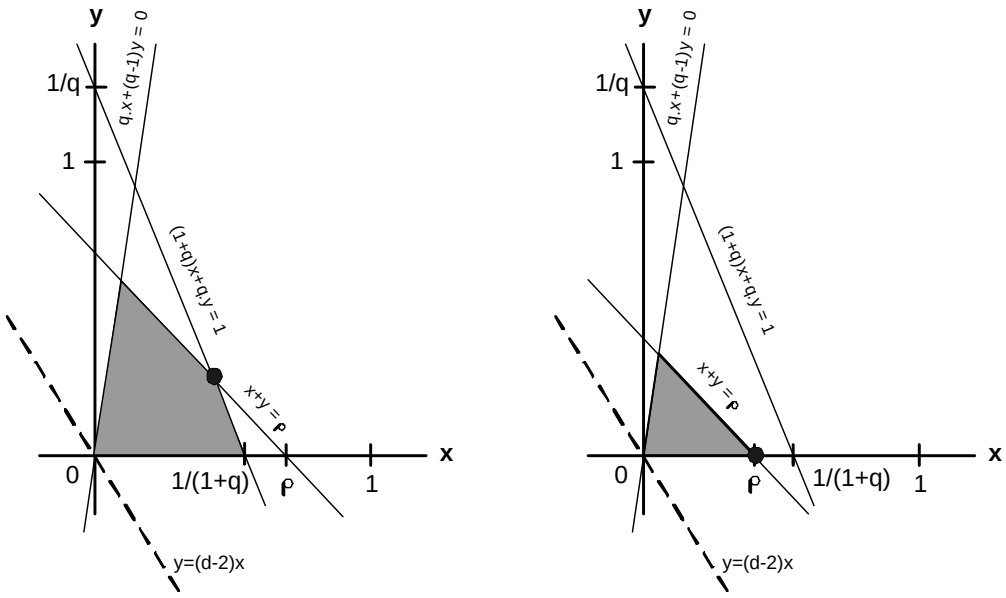
We show on Figure A.1 the graphical resolution of this LP if $\rho \geq (1+q)$, or if $\rho \leq (1+q)$. On both figures, the shaded area represents the solution space of the LP; an optimal solution is the particular vertice from the border of the solution space that maximizes the objective function $(2-d)x + y$. The optimal solution is:

$$x = \rho, \qquad y = 0 \qquad \text{if } \rho \leq 1/(1+q); \tag{A.8}$$

$$x = 1 - q \cdot \rho, \quad y = (1+q)\rho - 1 \quad \text{if } \rho \geq 1/(1+q). \tag{A.9}$$

This yields the optimal values for z_{00}, z_{01}, z_{10} and z_{11}:

$$\begin{cases} z^*_{00} = 1 - (1+q)\rho \\ z^*_{01} = \rho \\ z^*_{10} = q \cdot \rho \\ z^*_{11} = 0 \end{cases} \text{ if } \rho \leq 1/(1+q), \quad \text{and} \quad \begin{cases} z^*_{00} = 0 \\ z^*_{01} = 1 - q \cdot \rho \\ z^*_{10} = 1 - \rho \\ z^*_{11} = (1+q)\rho - 1 \end{cases} \text{ if } \rho \geq 1/(1+q). \tag{A.10}$$



(a) $\rho \geq 1/(1+q)$ (b) $\rho \leq 1/(1+q)$

Figure A.1: Graphical representation of the LP for optimal transmission of 1 layer

The optimal policies for Problem 6.1 are obtained after normalizing according to Step 2 in Section 6.4.1:

$$\sigma_{01}^* = \rho/(1-\rho q), \quad \sigma_{11}^* = 0 \qquad \text{if } \rho \leq 1/(1+q); \tag{A.11}$$

$$\sigma_{01}^* = 1, \qquad \sigma_{11}^* = 1 + (\rho - 1)/(\rho q) \quad \text{otherwise.} \tag{A.12}$$

The maximum rate and minimum distortion are given by:

$$rate_{avg}^* = \sum_{i=0}^{1}\sum_{a=0}^{1} c(i,a) \cdot z_{ia}^* = \rho \tag{A.13}$$

$$dist_{avg}^* = -\sum_{i=0}^{1}\sum_{a=0}^{1} r(i,a) \cdot z_{ia}^* = \begin{cases} 1 - \rho q(2 - d_{10}) & \text{if } \rho \leq 1/(1+q), \\ (1-\rho q)(1 - q(1-d_{10})) & \text{otherwise.} \end{cases} \tag{A.14}$$

Appendix B:
Résumé Long en Français

Chapitre 1 : Introduction

Motivations

Les applications vidéo sur réseau n'en sont encore aujourd'hui qu'à leurs balbutiements. Alors que la technologie de la vidéo numérique progresse et que le réseau Internet devient de plus en plus omniprésent, il est probable que la demande d'applications vidéo sur réseau augmente de manière importante dans les prochaines années. Les opérateurs de télécommunications commencent à déployer de nouveaux services afin de mieux rentabiliser leurs infrastructures (ADSL, UMTS, WiFi), les fabriquants créent constamment de nouveaux produits innovants (téléphones portables, PDAs, décodeurs numériques, ordinateurs portables légers), et les grands acteurs de l'industrie du film et des média rivalisent pour vendre du contenu multimédia numérique en ligne (Pressplay, iTunes, MovieLink). Ces grandes compagnies ne vont peut-être pas changer en un an ou deux la manière dont les gens travaillent, communiquent, et se divertissent — comme certains investisseurs l'on injustement pensé aux débuts de l'Internet — mais ils vont certainement réussir à plus long terme, car les applications vidéo sur réseau peuvent apporter une valeur ajoutée réelle aux entreprises et aux individus.

Pour les entreprises, les applications vidéo sur réseau peuvent apporter des moyens de communication interne plus rapides et plus efficaces. Lors d'événements importants, les dirigeants de certaines grandes sociétés s'adressent dorénavant à leurs employés par le biais du réseau intranet de l'entreprise. Le discours peut être suivi en direct, ou bien stocké et transmis à la demande à n'importe quel moment de la journée (par exemple aux employés basés à l'étranger). La communication vidéo peut rendre la formation plus flexible et moins onéreuse que la formation conventionnelle. Grâce à la visioconférence et la collaboration à distance par la vidéo, les employés peuvent éviter des déplacements professionnels, économisant ainsi du temps et de l'argent. Enfin, les entreprises peuvent utiliser les applications vidéo sur réseau pour communiquer sur leurs produits dans le monde entier, directement à partir de leur page

Web. Par exemple, certaines entreprises font maintenant la publicité du lancement d'un nouveau produit en publiant la vidéo de l'événement sur Internet.

Les applications vidéo sur réseau ont aussi le potentiel d'améliorer la communication et le divertissement des individus. En 2002, les Français ont passé en moyenne 3h20 à regarder la télévision quotidiennement[2]. La vidéo sur réseau peut apporter à l'utilisateur un plus grand choix de programmes que la télévision conventionnelle, ainsi que des programmes interactifs et du contenu orienté vers l'utilisateur, tel que des informations ciblées. En dehors des programmes télévisuels, la consultation de contenu vidéos sur VHS ou DVD (également appelé "home video") est aussi une application de divertissement populaire (et une source importante de revenus pour l'industrie du film[3]). Transmettre des films à la demande sur Internet permettrait aux utilisateurs d'avoir accès à un contenu plus étendu que dans les boutiques de location, et de manière plus rapide. Enfin, la communication visuelle peut améliorer la communication téléphonique ordinaire, par exemple en permettant aux familles éloignées de garder le contact visuel grâce aux visiophones.

La plupart des applications précédemment citées nécessitent un support important de l'infrastructure réseau, ainsi que des logiciels et des matériels spécifiques. L'évolution actuelle du réseau Internet et des terminaux de communication est favorable aux applications de vidéo sur réseau : ces dernières années ont été marquées par le déploiement de connexions Internet haut-débit ; les capacités de calcul des terminaux ont également augmenté, et plusieurs systèmes et standards de codage vidéo ont été conçus spécifiquement pour le streaming vidéo. Pourtant, aujourd'hui, les applications vidéo sur Internet sont souvent de mauvaise qualité. Les vidéos transmises sont souvent saccadées et leur qualité peut se dégrader fortement pendant la consultation. L'objectif de cette thèse est de proposer de nouvelles techniques et de nouveaux algorithmes pour améliorer la qualité des applications de streaming vidéo sur Internet.

Contexte de la Thèse et Contributions Principales

Dans ce mémoire, nous nous concentrons sur les applications de streaming vidéo point–à–point (aussi appelé "unicast" en anglais), c'est-à-dire que les vidéos sont transmises d'un serveur (ou proxy) à un client. Nous formulons des problèmes d'optimisation et obtenons des politiques de contrôle pour des applications vidéo sur l'Internet actuel dit "best–effort" ; nous cherchons particulièrement des procédures d'optimisation de faible complexité, qui sont utilisables par des serveurs qui servent un grand nombre d'utilisateurs simultanément.

[2] D'après la société Médiamétrie, principale société de mesure de l'audience en France (www.mediametrie.fr).

[3] A son apogée, le marché de la "home video" générait la moitié des ventes de l'industrie du film et le trois quart de ses profits (The Economist, 19/09/2002).

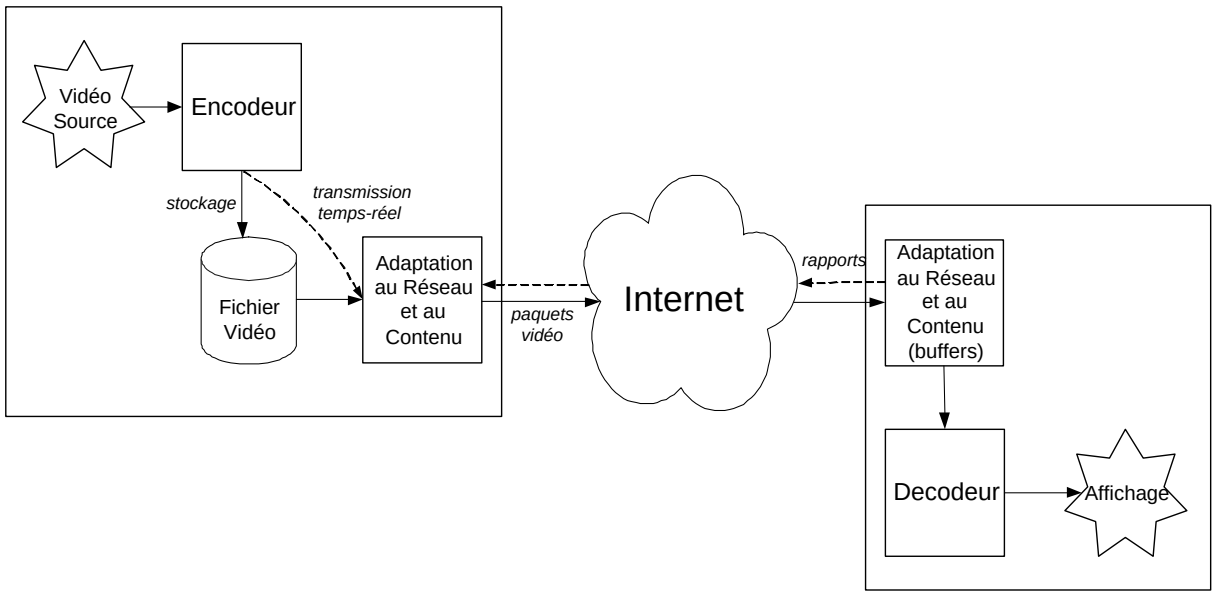


Figure B.1: Système de streaming

La Figure B.1 donne un aperçu d'un système typique de streaming vidéo sur Internet. La vidéo source est d'abord encodée au serveur. Les images vidéos encodées sont stockées dans un fichier pour transmission ultérieure, ou peuvent être envoyées au client directement en temps réel. Les techniques adaptatives au serveur consistent à déterminer la manière dont les paquets vidéo sont envoyés au client, notamment en fonction des conditions actuelles du réseau et des rapports qui peuvent être renvoyés par le client. Du côté du client, les paquets vidéo sont en général temporairement bufferisés avant d'être envoyés au décodeur. Enfin, après décodage, les images vidéo sont affichées à l'utilisateur.

Dans cette thèse, nous étudions des techniques qui adaptent la transmission à la fois aux conditions variables du réseau (adaptation au réseau) et aux caractéristiques de la vidéo transmise (adaptation au contenu). Les techniques adaptatives sont combinées avec une forme particulière de codage vidéo (codage en couches), qui encode la vidéo en au moins deux couches complémentaires ; le serveur peut s'adapter à des changements de condition du réseau en ajoutant/retranchant des couches.

Cette thèse apporte plusieurs contributions majeures :

– Nous comparons des politiques de contrôle qui ajoutent et retranchent des couches avec des politiques de contrôle qui changent la version de la vidéo transmise. Dans le contexte d'une transmission TCP–compatible fiable, nous montrons que le changement de versions atteint en général une meilleure performance que l'ajout/retranchement de couches, à cause du surcoût en bande passante associé au codage en couches.

– Nous présentons un cadre d'étude nouveau pour le streaming adaptatif à faible complexité de vidéos à granularité fine (FGS), sur une connexion TCP–compatible fiable. Nous formulons et résolvons un problème d'optimisation, qui suggère une heuristique en temps réel. Nous montrons que notre système donne des résultats similaires avec des connexions TCP–compatibles peu ou très variables. Nous présentons une implémentation de notre heuristique dans une plateforme de streaming de vidéos MPEG–4 FGS.

– Nous analysons des traces débit–distorsion de vidéos MPEG–4 FGS ; nous trouvons que le contenu sémantique et le codage de la couche de base ont un impact important sur les propriétés de la couche d'amélioration FGS. Nous formulons un problème d'optimisation pour comparer le streaming optimal suivant les caractéristiques de débit–distorsion à différent niveaux d'agrégation ; nous trouvons que le streaming optimal adaptatif scène par scène peut atteindre de bonnes performances, pour une complexité de calcul plus faible que l'adaptation image par image.

– Nous proposons un cadre d'étude point–à–point unifié qui combine l'ordonnancement, la protection d'erreur (FEC) et la dissimulation d'erreur au décodeur ("error concealment"). Nous utilisons des Processus de Décision de Markov (MDPs) sur un horizon infini pour trouver des politiques de transmission optimales avec une faible complexité et pour une grande variété de mesures de performance. En utilisant des vidéos MPEG–4 FGS, nous montrons que la prise en compte de la dissimulation d'erreur au décodeur peut améliorer grandement la qualité de la vidéo reçue, et qu'une stratégie de protection d'erreur statique atteint des performances quasi–optimales.

Chapitre 2 : Streaming Vidéo sur Internet

Dans le chapitre 2, nous donnons un aperçu des principales technologies mises en œuvre dans une application de streaming vidéo sur Internet, c'est–à–dire le réseau Internet et le codage vidéo numérique. En se basant sur ces généralités, nous nous focalisons sur les problèmes qui sont spécifiques au streaming vidéo sur Internet, et nous détaillons les travaux de recherche existants.

Le Réseau Internet

Le réseau Internet est toujours "best–effort", c'est–à–dire qu'il ne garantie pas de qualité de service (QoS) aux applications. Les mesures de QoS qui sont essentielles pour la plupart des applications Internet sont le délai de transmission point–à–point, le taux de perte de paquets et la bande passante disponible pour la connexion. Le réseau Internet actuel est caractérisé par des conditions de transmission à la fois très hétérogènes et variables dans le temps.

Tout d'abord, l'hétérogénéité du réseau provient de la diversité structurelle de l'Internet et de la

diversité du matériel utilisé à travers tout le réseau, tel que les liens physiques [43]. En particulier, l'hétérogénéité en bande passante pour une connexion point–à–point donnée peut être associée à ce que l'on appelle le "problème du dernier kilomètre", c'est–à–dire que le goulot d'étranglement en bande passante d'une connexion est souvent situé sur le lien entre l'utilisateur final et son fournisseur d'accès à Internet. Les liaisons d'accès à Internet les plus utilisées actuellement (câble, ADSL, modems bas–débit) ont des capacités en bande passante différentes, ce qui contribue à l'hétérogénéité du réseau. Outre la diversité des technologies d'accès à Internet, l'éloignement variable des terminaux ainsi que la concurrence des flux à l'intérieur du réseau contribuent également à l'hétérogénéité de l'Internet.

Ensuite, pendant sa durée de vie, une connexion Internet donnée voit ses conditions varier à court et long terme. Ces variations sont principalement dues à la compétition du trafic dans les routeurs et aux changements de route, qui résultent d'une panne ou d'une décision de routage.

Les applications sur Internet peuvent utiliser les protocoles de transmission TCP ou UDP. Alors que le protocole UDP fournit un service minimum de multiplexage/démultiplexage sans connexion, le protocole TCP fournit un service de transmission fiable qui nécessite l'établissement d'une connexion. Outre la retransmission des paquets perdus, TCP implémente deux mécanismes pour contrôler le débit sortant : le *contrôle de flux* et le *contrôle de congestion*. Le contrôle de congestion de TCP permet d'assurer la stabilité du réseau, mais au prix de fortes variations à court terme de la bande passante disponible.

Les caractéristiques "best–effort" du réseau Internet actuel que nous avons soulignées précédemment sont peut-être suffisantes pour des applications de transfert de fichier, mais elles ne sont pas adaptées aux applications temps–réel comme la visioconférence. Il existe des propositions pour apporter de la qualité de service à l'Internet actuel, telles qu'IntServ ou DiffServ. Cependant, la plupart des propositions nécessitent des modifications de l'architecture du réseau actuel, ce qui explique qu'elles n'ont pas encore été déployées. Ainsi, les applications réseau telles que le streaming vidéo doivent s'adapter au manque de qualité de service de l'Internet actuel.

Le Codage Vidéo

A cause des capacités de transmission limitées du réseau Internet, les vidéos doivent être compressées avant d'être transmises sur le réseau. Le codage vidéo consiste à exploiter les redondances spatiales et temporelles inhérentes au signal vidéo, afin de réduire la taille de représentation des vidéos. Les standards de codecs (codeur/décodeur) les plus utilisés pour les applications vidéo sur Internet sont H.261 et H.263 pour la transmission en temps réel, et MPEG–1, 2 et 4 pour le streaming de vidéos stockées. Dans cette thèse nous considérons plutôt les standards MPEG, en particulier MPEG–4. MPEG–4 est un standard récent qui a été conçu pour un large éventail d'applications, notamment le streaming vidéo sur Internet.

Un des objectifs principaux du standard est la manipulation flexible d'objets audio–visuels.

Le *codage hiérarchique* — également appelé *codage scalable* ou *codage en couches* — est une technique de codage qui est particulièrement adaptée aux applications vidéo sur réseau. Le concept principal est d'encoder la vidéo en plusieurs couches complémentaires : la couche de base (BL), et une ou plusieurs couches d'amélioration (ELs). La couche de base est une version de basse qualité de la vidéo. Elle doit être décodée afin d'afficher une qualité minimale acceptable de la vidéo. La qualité de la vidéo affichée à l'utilisateur peut-être progressivement améliorée en décodant les couches d'amélioration successives. Toutes les couches d'amélioration sont codées de manière hiérarchique : afin de décoder la couche d'amélioration d'ordre n, le décodeur doit posséder toutes les couches d'ordre inférieur, c'est-à-dire la couche de base et toutes les couches d'amélioration d'ordre 1 à $n-1$. Il y a plusieurs types de scalabilité : Data Partitioning, scalabilité temporelle, spatiale ou SNR (en anglais Signal–to–Noise Ratio). Ces différents types ont des complexités d'implémentation différentes et des surcoûts en bande passante différents par rapport à une vidéo non–scalable.

La scalabilité à granularité fine (en anglais "Fine Granularity Scalability" ou FGS) est une nouvelle forme de codage en couches, qui a été introduite dans le standard MPEG–4 spécifiquement pour la transmission de vidéos sur Internet [8]. La particularité du codage FGS est que le flux de données de la couche d'amélioration peut être tronqué n'importe où lors de la transmission, la partie restante pouvant toujours être décodée. La Figure B.2 montre un exemple de coupure de la couche d'amélioration FGS avant la transmission sur réseau. Pour chaque image, la partie sombre dans la couche d'amélioration représente la partie de la couche FGS qui est effectivement transmise du serveur au client. Le fait de tronquer la couche d'amélioration FGS pour chaque image ou groupe d'images avant la transmission permet au serveur d'adapter son débit de transmission à la bande passante variable de la connexion. Au client, le décodeur peut utiliser la partie reçue de la couche FGS pour améliorer la qualité de la couche de base.

Dans cette thèse nous considérons le scalabilité à granularité fine dite SNR. Il existe aussi la scalabilité FGS temporelle, et plusieurs améliorations de la technique de codage FGS classique ont été proposées.

Transmission de Vidéos sur Internet

Nous nous concentrons à présent sur l'application même de transmission de vidéos sur réseau. Nous décrivons tout d'abord les choix principaux de conception d'une telle application, choix applicatifs et choix de transport.

Parmi les choix applicatifs à faire pour les vidéos stockées, il faut déterminer si la vidéo doit être téléchargée (comme un fichier) ou bien "streamée". Avec le streaming, l'utilisateur peut commencer à regarder la vidéo rapidement après que la transmission a débuté ; il peut visionner les parties de la

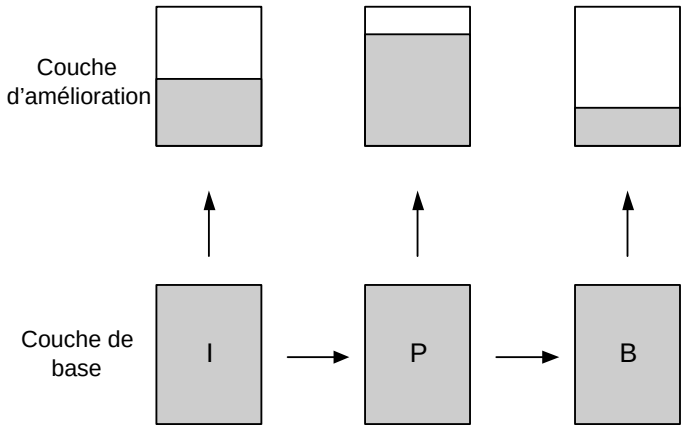


Figure B.2: Exemple de coupure de la couche d'amélioration FGS avant transmission

vidéo qui ont déjà été reçues, tandis que le serveur vidéo est en train de transmettre des parties futures de la vidéo. Notons que la technique de streaming peut concerner à la fois les vidéos stockées ou les vidéos envoyées en temps-réel (type visioconférence). Par rapport au téléchargement, le streaming réduit considérablement le temps d'attente de l'utilisateur pour les vidéos stockées. Cependant, étant donné que le réseau Internet actuel n'a pas de garantie de qualité de service, avec le streaming il n'y a pas de garantie que l'utilisateur puisse regarder la vidéo jusqu'à la fin dans de bonnes conditions. Par conséquent, les applications de streaming vidéo nécessitent des techniques spécifiques d'adaptation au réseau, afin de maximiser la qualité finale de la vidéo affichée.

Il y a quatre familles de techniques de codage vidéo qui peuvent être utilisées pour le streaming adaptatif : le codage à la volée, le changement de versions, l'ajout/retranchement de couches, et l'ajout/retranchement de descriptions. En particulier, le changement de versions consiste à encoder la vidéo à plusieurs débits, correspondant à plusieurs niveaux de qualité. Chaque version est stockée sur le serveur ; le serveur change la version qui est transmise au client afin d'adapter le débit de la vidéo transmise (et ainsi le niveau de qualité) à la bande passante disponible. Une technique alternative au changement de versions consiste à utiliser une vidéo encodée en couches et à changer le nombre de couches à envoyer au client en fonction des conditions du réseau (ajout/retranchement de couches).

Au niveau du transport, le concepteur d'une application de transmission de vidéos sur Internet a le choix entre les protocoles TCP et UDP. TCP est habituellement considéré comme inapproprié pour le streaming vidéo, à cause de (i) sa fiabilité (alors que la vidéo peut en général tolérer un certain taux d'erreurs), (ii) des forts délais induits par les retransmissions, et (iii) des variations de débit engendrées par le mécanisme de contrôle de congestion AIMD. Pour ces raisons, il est en général préférable d'utiliser UDP, en lui adjoignant un mécanisme de contrôle de congestion TCP–compatible à débit peu

variable, et un mécanisme de correction d'erreur partiel : retransmissions sélectives ou codes correcteurs d'erreur FEC (en anglais "Forward Error Correction"). Néanmoins, nous pensons que TCP a beaucoup d'avantages sur UDP pour le streaming de vidéos stockées : sa fiabilité est un atout pour la transmission de vidéos fortement compressées (et donc peu résistantes aux erreurs), et les forts délais ou les variations de bande passante à court terme peuvent être contenus par des buffers placés au client, comme nous le montrons dans la suite du mémoire.

Techniques Adaptatives pour le Streaming de Vidéo sur Internet

Dans cette section nous décrivons les systèmes existants et les travaux de recherche précédents concernant les systèmes de streaming vidéo adaptatifs sur Internet. Nous considérons tout d'abord les techniques générales pour adapter le streaming aux conditions variables du réseau (adaptation au réseau). Du fait des spécificités de la vidéo, on adjoint en général à ces techniques des techniques qui adaptent le streaming aux caractéristiques de la vidéo transmise (adaptation au contenu).

A cause des caractéristiques variables et hétérogènes de l'Internet "best–effort", les systèmes de streaming vidéo doivent implémenter des mécanismes qui adaptent la transmission à l'état actuel du réseau. Un mécanisme d'adaptation au réseau qui est présent dans la plupart des applications actuelles (Real Media, Windows Media ou Quicktime) consiste à introduire un délai de playback de quelques centaines de millisecondes avant que la vidéo puisse être affichée à l'utilisateur. Pendant ce temps, le serveur envoie une portion initiale de la vidéo dans un buffer dédié du client, que nous appelons le *buffer de playback*. Après le début de l'affichage, le serveur continue d'envoyer de nouvelles données vidéo dans le buffer de playback du client, tandis que le décodeur consomme les données qui sont disponibles au début du buffer. Des mécanismes d'adaptation ont été conçus pour maintenir un délai de playback suffisant pendant la durée du streaming, afin de contenir la gigue du réseau. Les applications de streaming de vidéos stockées ont des contraintes de délai moins strictes que les applications de streaming en temps–réel de type visioconférence. Ainsi, elles peuvent supporter un délai de playback plus important, en général jusqu'à plusieurs secondes. Un délai de playback important permet de contenir, au–delà de la gigue, les variations de bande passante à court terme : le serveur peut utiliser les périodes d'excès de bande passante pour envoyer en avance des portions futures de la vidéo, afin d'anticiper les prochaines périodes de diminution drastique de la bande passante disponible.

Les mécanismes de correction d'erreur pour le streaming vidéo, c'est–à–dire les retransmissions sélectives ou l'envoi de codes correcteurs, doivent également s'adapter aux variations de condition du réseau. Plusieurs études ont présenté des techniques d'adaptation au réseau pour l'un ou l'autre de ces mécanismes de correction d'erreur, ou même pour une utilisation combinée des deux mécanismes.

Enfin, comme nous l'avons mentionné précédemment, l'utilisation de vidéos encodées en couches

peut permettre d'adapter le streaming aux conditions actuelles du réseau, et plus précisément aux variations de la bande passante TCP–compatible disponible pour l'application. Saparilla et Ross [113, 114] ont étudié l'allocation optimale de la bande passante aux couches de la vidéo, et ont proposé des politiques d'adaptation au réseau basées sur l'état de remplissage des buffers de playback au client. Rejaie et al. [106] ont étudié un autre mécanisme d'adaptation, pour un protocole de transport TCP–compatible particulier, RAP.

Les techniques précédentes visent à maximiser des mesures de performance orientées réseau, telles que l'usage de la bande passante disponible. Bien que ces mesures peuvent donner de bonnes indications de qualité globale, en général, la qualité de la vidéo affichée après transmission n'est pas directement proportionnelle au nombre de bits reçus ou à la proportion de paquets reçus. Par conséquent, afin de maximiser réellement la qualité perçue au récepteur, les applications de streaming avec adaptation au réseau doivent aussi considérer les spécificités de la vidéo source et de l'encodage utilisé. C'est ce que l'on appelle le streaming avec adaptation au contenu.

L'utilisation de vidéos codées en couches pour l'adaptation aux variations de conditions du réseau peut engendrer de fortes variations de qualité dans la vidéo affichée au récepteur. Cela se produit avec des mécanismes d'adaptation au réseau qui ajoutent et retranchent les couches trop fréquemment, afin de s'adapter au mieux aux variations de la bande passante disponible. De fortes variations de qualité entre les images successives peuvent diminuer la perception globale de la qualité de la vidéo. Des mécanismes ont ainsi été proposé afin de minimiser ces variations de qualité [88, 106, 114]. Dans cette thèse, nous considérons les variations de qualité pour les mécanismes de changement de versions et d'ajout/retranchement de couches dans le Chapitre 3. Dans le Chapitre 4, nous nous intéressons spécifiquement au streaming de vidéos en couches FGS.

Une autre technique d'adaptation au contenu, qui est également spécifique aux vidéos encodées en couche, est la protection inégale des couches. Cela consiste à protéger les couches basses (dont la couche de base) plus fortement que les couches hautes, afin de s'assurer que les couches reçues au client peuvent toutes être utilisées (c'est–à–dire qu'elles respectent la hiérarchie du codage en couches). La protection inégale des couches est en général obtenue en protégeant chaque couche avec une quantité différente de codes correcteurs d'erreur, ou bien en retransmettant les couches les plus importantes avant les couches moins importantes.

Enfin, ce que l'on appelle le streaming optimisé suivant les caractéristiques débit–distorsion est une famille de mécanismes adaptatifs qui consistent à évaluer l'espérance de la distorsion finale de la vidéo après transmission, et de trouver des politiques de transmission optimales qui maximisent cette espérance. Cela nécessite de connaître, au serveur, les caractéristiques débit–distorsion de chaque paquet vidéo, ainsi que l'interdépendance entre les paquets. Chou et Miao [26, 27] ont proposé un cadre général pour évaluer la distorsion finale de la vidéo pour un large éventail d'environnements de streaming. Dans

cette thèse, nous analysons les propriétés débit–distorsion de vidéos MPEG–4 FGS dans le Chapitre 5. Dans le Chapitre 6, nous étudions le streaming optimisé suivant les caractéristiques débit–distorsion, en prenant en compte la dissimulation d'erreur au décodeur.

Chapitre 3 : Streaming de Versions ou de Couches ?

Dans le Chapitre 3, nous comparons, pour des vidéos stockées, les mécanismes adaptatifs de changement de version (à partir d'une vidéo non–scalable encodée à différents débits) et d'ajout/retranchement de couches (à partir d'une vidéo scalable). Dans le cas du changement de versions, il faut concevoir des politiques de contrôle qui permettent de décider quelle est la version qui doit être transmise, en fonction de l'état actuel du système et des conditions du réseau. Dans le cas de l'ajout/retranchement de couches, les politiques de contrôle consistent à décider s'il faut ou non ajouter ou supprimer la transmission d'une couche. Bien que l'ajout/retranchement de couches soit une technique efficace de transmission adaptative, le codage en couches augmente la complexité du codage de la vidéo, ce qui entraîne une efficacité de codage inférieure au codage non scalable.

Nous présentons d'abord notre modèle et les mesures de performance qui sont utilisées pour comparer les deux mécanismes. Nous considérons un codage en 2 couches, et un mécanisme qui alterne entre 2 versions (basse et haute qualité). Dans notre modèle, pour une même qualité, le débit total de la couche de base et d'amélioration est supérieur au débit de la version de haute qualité : le surcoût en débit du codage en couche est de H (en pourcentage). Nous considérons la transmission sur une connexion TCP–compatible fiable. Le serveur envoie les données au taux maximal donné par la bande passante disponible ; le client stocke les données dans un buffer de playback. Nous supposons un délai de playback de quelques secondes, ce qui nous permet de négliger les délais de transmission des paquets. Nous supposons que les vidéos sont encodées à débit constant.

Nous concevons ensuite des politiques de contrôle analogues pour les deux mécanismes, qui permettent de faire une comparaison juste entre les performances du changement de versions, et de l'ajout et retranchement de couches. Pour les deux mécanismes, ces politiques dépendent du niveau de remplissage des buffers de playback au client (estimé au serveur à partir de rapports périodiques renvoyés par le client). Elles visent à maintenir un affichage continu de la vidéo et à minimiser les changements de qualité au client. Pour le mécanisme d'ajout/retranchement de couches, on considère deux implémentations du serveur. Dans la première implémentation, le serveur est contraint d'envoyer les deux couches d'une même image au même moment ; dans la deuxième implémentation, le serveur peut améliorer la qualité d'une partie de la vidéo qui est déjà stockée au client (en transmettant la couche d'amélioration correspondant aux images dont la couche de base est déjà stockée au client).

Nous comparons les performances des deux mécanismes grâce à des simulations, à partir de traces

TCP d'une heure, collectées sur Internet. Nous faisons varier les conditions critiques du système, telles que la bande passante moyenne disponible, et le surcoût en débit associé au codage en couches. Nos simulations montrent que nos politiques de contrôle pour les deux mécanismes adaptent le streaming aux variations du réseau de manière efficace. Néanmoins, dans la première implémentation de l'ajout/retranchement de couches, le surcoût en débit du codage en couches entraîne, dans tous les cas, une baisse des performances de l'ajout/retranchement de couches par rapport au changement de versions.

Lorsque l'on utilise la deuxième implémentation, plus flexible, de l'ajout/retranchement de couches, aucun des deux mécanismes ne semble dominer : pour de faibles valeurs du surcoût du codage en couches H, la meilleure flexibilité du codage en couches peut compenser les pertes de qualité dues au surcoût. Pour de fortes valeurs de H, le changement de versions atteint de meilleurs performances dans toutes les situations de bande passante étudiées. Donc, pour le streaming de vidéos stockées sur une connexion TCP–compatible fiable, il est très important d'utiliser des types de codage en couches efficaces, afin de maintenir un faible surcoût en débit par rapport au codage scalable.

Chapitre 4 : Streaming de Vidéos FGS Stockées

Alors que dans le chapitre 3, nous avons étudié le mécanisme d'ajout/retranchement de couches, à partir d'un codage en couches conventionnel, nous étudions dans le Chapitre 4, le streaming adaptatif de vidéos encodées à granularité fine (FGS). La propriété de granularité fine du codage FGS permet de concevoir des politiques de contrôle qui adaptent le streaming aux variations de bande passante du réseau, de manière très fine (bit par bit). Cela constitue un avantage majeur par rapport aux mécanismes classiques d'ajout/retranchement de couches, ainsi que par rapport au changement de versions pour lequel le débit de chaque version est fixe.

Dans ce chapitre, nous présentons un nouveau cadre d'étude pour le streaming de vidéos FGS. Nous résolvons un problème d'optimisation pour un critère de qualité orienté réseau. D'après l'étude des politiques optimales, nous proposons une heuristique en temps réel, dont nous étudions les performances à partir de traces Internet et d'une implémentation dans un système de streaming de vidéos MPEG–4.

Comme dans le Chapitre 3, nous considérons une connexion TCP–compatible fiable ; la vidéo est encodée à débit constant, et des buffers de playback suffisamment remplis au client nous permettent de négliger les délais de transmission des paquets. Nous supposons aussi que le serveur envoie les données au taux maximal permis par la bande passante disponible ; le serveur peut connaître l'état de remplissage des buffers de playback grâce aux rapports renvoyés par le client. Pour des raisons de simplicité d'implémentation, nous contraignons le serveur à envoyer toutes les données d'une même image (couche de base et couche d'amélioration tronquée) au même moment. Dans notre modèle, le temps au serveur est divisé en intervalles de temps $[t_k, t_{k+1}]$. Au début de chaque intervalle de temps, le

for $k = 0$ *to* $N_{slot} - 1$ **do**

Estimer la valeur de Δ_k à partir des rapports du client

Calculer $X_{avg}(k-1)$

Calculer la valeur de $r_s(k)$ *:*

if $\Delta_k \leq C_{slot}$ **then**

$r_s(k) = r_b$

else if $C_{slot} < \Delta_k \leq 2 \cdot C_{slot}$ **then**

$r_s(k) = \alpha \cdot X_{avg}(k-1) + (1-\alpha) \cdot r_s(k-1)$

else if $\Delta_k \geq 2 \cdot C_{slot}$ **then**

$r_s(k) = \alpha \cdot X_{avg}(k-1) \cdot \frac{\Delta_k}{2 \cdot C_{slot}} + (1-\alpha) \cdot r_s(k-1)$

Vérifier que $r_s(k)$ *reste dans les limites :*

if $r_s(k) < r_b$ **then** $r_s(k) = r_b$

if $r_s(k) > r_b + r_e$ **then** $r_s(k) = r_b + r_e$

end

Algorithm B.1: Heuristique temps–réel

serveur détermine le débit constant sur l'intervalle de temps $[t_k, t_{k+1}]$ de la couche d'amélioration FGS à envoyer au client, noté $r_s(k)$. Par simplicité, chaque intervalle de temps a une durée constante de C_{slot} secondes.

Nous formulons le problème d'optimisation suivant :
trouver une politique de transmission $\boldsymbol{r}_s = (r_s(0), \ldots, r_s(N_{slot}-1))$ qui :

- assure un minimum de qualité, en assurant une transmission de la couche de base sans perte de données (c'est–à–dire sans famine du buffer de playback) ;
- maximise une mesure d'efficacité en bande passante, E, qui donne une bonne indication de la qualité globale de la vidéo ;
- minimise une mesure de variation du débit de la vidéo affichée à l'utilisateur, V, qui donne une indication des variations de distorsion des images affichées.

Nous analysons et résolvons ce problème par programmation dynamique, en supposant que la bande passante disponible $X(t)$ est connue a priori.

Dans une situation réelle de streaming vidéo, la bande passante disponible pour la durée du streaming n'est pas connue a priori. Dans ce cas nous proposons une heuristique, à partir de l'étude des politiques optimales lorsque la bande passante est connue a priori. Cette heuristique adapte à la volée le débit de la couche d'amélioration pour chaque intervalle de temps, $r_s(k)$, en fonction des conditions du réseau. L'Algorithme B.1 présente notre heuristique. Les notations suivantes sont utilisées : r_b et r_e représentent les taux de codage respectifs de la couche de base et de la couche d'amélioration FGS ; Δ_k représente le délai de playback au début de l'intervalle $[t_k, t_{k+1}]$; $X_{avg}(k-1)$ est la moyenne de bande passante disponible sur l'intervalle $[t_{k-1}, t_k]$; α est une constante comprise entre 0 et 1, qui permet de réaliser un compromis entre l'efficacité en bande passante E et la variation du taux de codage V.

Nous faisons des simulations tout d'abord à partir de traces TCP de 5 minutes, collectées sur Internet. En faisant varier les conditions du réseau, nous montrons que notre heuristique atteint de bonnes performances par rapport aux performances données par la politique optimale, c'est–à–dire celle obtenue en connaissant a priori la bande passante disponible. L'efficacité obtenue est proche de l'efficacité de la politique optimale dans la plupart des conditions de réseau ; la variabilité atteinte est en général inférieure à la variabilité d'une politique de transmission qui ajouterait la couche d'amélioration entière seulement une fois durant le streaming.

Nous effectuons d'autres simulations avec le logiciel *ns* (network simulator), pour comparer les performances de notre heuristique et des politiques optimales dans le cas d'une transmission sur TCP ou sur un autre protocole de contrôle de congestion TCP–compatible, TFRC. Pour des conditions de charge du réseau différentes, nous observons que la variabilité minimale atteinte par la politique optimale sur TCP est seulement légèrement supérieure à la variabilité de la politique optimale sur TCP, alors que le débit de TFRC est beaucoup moins variable que celui de TCP. Lorsque l'on compare les performances de l'heuristique sur TCP ou TFRC, nous observons des résultats similaires pour les deux protocoles.

Finalement, nous présentons l'implémentation de notre heuristique, adaptée au cas d'une couche de base à débit variable, dans une plateforme de streaming de vidéos MPEG–4 FGS. Notre implémentation fonctionne sur TCP, mais pourrait également fonctionner sur n'importe quelle connexion RTP/UDP TCP–compatible et partiellement fiable. Nous étudions le streaming d'une vidéo composée de plusieurs segments ayant des caractéristiques visuelles homogènes ; dans notre implémentation, le module de contrôle de débit de la couche d'amélioration FGS fixe le débit à transmettre $r_s(k)$ pour chaque segment. Nous effectuons des simulations sur un réseau LAN dédié, en présence de trafic concurrent. Ces simulations permettent de mettre en valeur les compromis entre différents critères de qualité, en fonction des paramètres de notre heuristique (comme le paramètre α).

Chapitre 5 : Propriétés Débit–Distorsion de Vidéos MPEG–4 FGS pour le Streaming Optimal

Dans notre approche précédente, nous avons considéré le problème de maximiser la qualité de la vidéo affichée au client, en utilisant des mesures de performance orientées réseau, telles que l'efficacité en bande passante. Cependant, la qualité finale de la vidéo affichée n'est pas directement proportionnelle au nombre de bits reçus par le client. S'il est vrai que la réception d'un plus grand nombre de données résulte en général en une meilleure qualité, tous les paquets vidéo n'apportent pas une même contribution en qualité à la vidéo affichée. Par conséquent, l'optimisation de la qualité finale de la vidéo requiert la prise en compte des particularités du type de codage utilisé et des séquences vidéos transmises. Dans le Chapitre 5 nous analysons les propriétés débit–distorsion des vidéos encodées en MPEG–4 FGS, afin d'obtenir des indications utiles pour le streaming optimal.

Nous présentons tout d'abord un cadre d'étude général pour l'analyse des performances de mécanismes adaptatifs de streaming de vidéos FGS. Nous définissons des mesures qui caractérisent le trafic et la qualité des images prises individuellement, et le trafic et la qualité de scènes vidéo (ou plus généralement de n'importe quelle agrégation arbitraire d'images). Nous expliquons comment utiliser les mesures de qualité objectives usuelles que sont le PSNR et le MSE, et nous détaillons les méthodes utilisées pour générer les traces de débit–distorsion.

Nous analysons les traces d'une vidéo de courte durée (*Clip*), ainsi que d'une base de données de vidéos de longue durée et de genres différents. Les résultats de notre analyse sont les suivants. Premièrement, la forme convexe des courbes débit–distorsion pour les plans–bit individuels des images de la couche FGS, suggère de prioriser le découpage de la couche FGS au plus près de la fin d'un plan–bit. (Notons cependant que le découpage de la couche d'amélioration seulement aux frontières des plans–bit résulterait en une adaptation aux conditions variables du réseau avec une granularité plus grossière). Deuxièmement, les différents types d'image de la couche de base (I, P, et B) et, en général, le codage de la couche de base, ont une influence importante sur la qualité totale obtenue. Nous avons observé, pour des débits constants de la couche d'amélioration, que des variations de qualité importantes dans la couche de base correspondent à des variations également importantes dans la qualité totale (couches de base et d'amélioration). Cela suggère de prendre en compte les différents types d'image du codage de la couche de base, pour le streaming de la couche d'amélioration FGS. Finalement, nous avons observé que, pour des débits constants de la couche FGS, la qualité totale de la vidéo a tendance à varier avec le contenu sémantique des scènes vidéos successives. Cela suggère de prendre aussi en compte la structure sémantique, en particulier en scènes, lors du streaming de la couche FGS.

Nous utilisons nos traces pour étudier le streaming optimal suivant les caractéristiques de débit–distorsion à différents niveaux d'agrégation d'image. Nous supposons que la couche de base est transmise sans pertes, et nous nous concentrons sur le streaming de la couche d'amélioration FGS. Nous formulons un problème d'optimisation qui consiste à minimiser la distorsion totale de segments de vidéo (appelés *segments d'allocation*), sous une contrainte de bande passante moyenne disponible. Etant données les fonctions de débit–distorsion de toutes les images, le serveur peut optimiser le streaming du segment considéré en allouant des bits de la bande passante disponible aux images individuelles, afin de maximiser la qualité totale du segment. Alternativement, le serveur peut grouper plusieurs images consécutives d'un segment d'allocation en sous–segments, appelés *séquences de streaming*, et optimiser le streaming sur la base de séquences de streaming (au lieu d'images individuelles). Dans ce cas, le serveur alloue le même nombre de bits à chaque image d'une séquence de streaming.

Nous considérons les cinq cas d'agrégation suivants : image, GoP, scène, constant (le segment d'allocation est divisé, de manière arbitraire, en séquences de streaming contenant un nombre constant d'images), et total (toutes les images d'un segment d'allocation forment une seule séquence de streaming). Nous résolvons notre problème d'optimisation par programmation dynamique, et nous étudions les solutions optimales à partir de nos traces. Nous trouvons que l'ajustement de débit FGS scène par scène réduit grandement la complexité de calcul de l'optimisation comparé à l'ajustement de débit image par image, pour une diminution de qualité mineure. Nous avons aussi observé que l'agrégation arbitraire d'images consécutives (c'est–à–dire sans porter attention à la structure scénique de la vidéo) peut engendrer des détériorations de qualité.

Chapitre 6 : Cadre Unifié pour le Streaming Optimal utilisant les MDPs

Le but du streaming optimal suivant les caractéristiques de débit–distorsion est de maximiser la qualité de la vidéo reçue au client, en évaluant, au serveur, l'espérance de qualité de la vidéo affichée, en fonction des conditions du réseau et des propriétés de débit–distorsion de chaque paquet vidéo. Cependant, les approches d'optimisation actuelles proposées dans la littérature ne prennent pas en compte la possibilité de dissimulation d'erreur au décodeur. Dans le Chapitre 6, nous présentons un cadre d'étude unifié pour le streaming optimal en tenant compte de la dissimulation d'erreur.

Dans une application typique de streaming audio ou vidéo, le serveur ou expéditeur effectue un *ordonnancement* des paquets à transmettre afin de maximiser la qualité de la vidéo affichée. Le serveur peut choisir de ne pas transmettre toutes les couches de certains paquets (ce mécanisme est aussi appelé adaptation de qualité [108]). Sur des canaux de transmission à pertes, l'ordonnancement est suivi par la correction d'erreur, afin de limiter les pertes de paquets et leurs effets sur la vidéo affichée. Pour les applications de streaming de données multimédia, la correction d'erreur consiste à retransmettre les

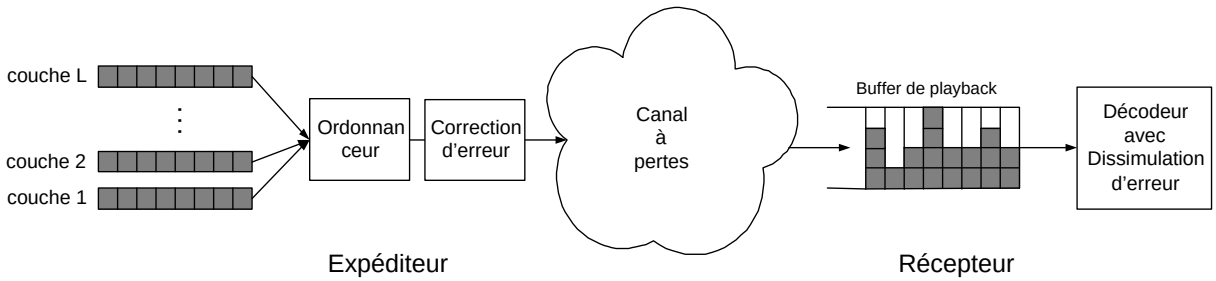


Figure B.3: Système de streaming vidéo avec dissimulation d'erreur au décodeur

paquets perdus avant leurs échéances de décodage, ou bien de transmettre des paquets de redondance (aussi appelé protection contre les erreurs). L'ordonnancement et la correction d'erreur doivent s'adapter conjointement aux variations de conditions du réseau, telles que la bande passante disponible et le taux de pertes de la connexion. Dans notre modèle, la correction d'erreur est effectuée par les codes FEC Reed–Solomon (RS).

Au récepteur, certains paquets vidéo sont disponibles à temps, c'est–à–dire avant leurs échéances de temps ; d'autres ne sont pas disponibles, soit parce qu'ils on été transmis et perdus, ou simplement parce que le serveur a décidé de ne pas les transmettre. Au moment de l'affichage à l'utilisateur, le décodeur applique plusieurs méthodes de *dissimulation d'erreur* afin de dissimuler au mieux les effets des paquets perdus. La dissimulation d'erreur ("Error Concealment" ou EC, en anglais) consiste à exploiter les corrélations spatiales et temporelles de l'audio ou la vidéo, afin d'interpoler les paquets manquants à partir des paquets disponibles. Pour la vidéo, une méthode de dissimulation d'erreur élémentaire est d'afficher, au lieu du macro–bloc manquant de l'image en cours, le macro–bloc à la même place dans l'image précédente.

L'ordonnancement de paquets, la correction d'erreur et la dissimulation d'erreur sont des composants fondamentaux d'un système de streaming vidéo point–à–point. La Figure B.3 illustre leurs fonctions respectives. Traditionnellement, l'ordonnancement et la correction d'erreur sont optimisées sans prendre en compte la présence de dissimulation d'erreur au récepteur [27, 85, 97].

Après avoir donné un exemple simple de l'utilité de prendre en compte la dissimulation d'erreur lors de l'optimisation de la transmission au serveur, nous formulons notre problème d'optimisation. Dans ce chapitre, nous considérons à la fois le streaming de vidéos stockées et le streaming de vidéos en temps–réel. Nous supposons que la couche de base est transmise sans perte et nous nous concentrons sur la transmission des couches d'amélioration. Nous supposons que les couches d'amélioration ne sont pas encodées avec compensation de mouvement. Nous considérons une séquence vidéo homogène ; le canal de transmission est un canal à pertes indépendantes et identiquement distribuées (i.i.d.). Au

décodeur, nous considérons une stratégie de dissimulation d'erreur temporelle simple : une couche manquante d'une image ne peut être remplacée que par la couche correspondante de l'image précédente. Néanmoins, le remplacement s'effectue en général avec une perte de qualité. Nous supposons que le serveur connaît, pour la séquence, les caractéristiques de distorsion de chaque couche après dissimulation d'erreur. Notre problème d'optimisation consiste à trouver la politique de transmission optimale qui minimise la distorsion moyenne avec une contrainte sur le débit moyen de transmission (correspondant à la bande passante moyenne disponible).

Afin d'illustrer nos résultats, nous utilisons des vidéos encodées en MPEG–4 FGS. Nous divisons la couche d'amélioration FGS en L sous–couches ; nous utilisons la vidéo à faibles mouvements *Akiyo*, encodée suivant deux qualités.

Dans un premier temps, nous analysons notre problème d'optimisation en supposant que le serveur peut observer l'état du récepteur au moment de décider de l'action à prendre pour l'image en cours (c'est–à–dire le nombre de paquets source et FEC à envoyer pour chaque couche de l'image). Cela implique d'avoir un canal de transmission retour du récepteur à l'expéditeur fiable, et une connexion avec un délai aller–retour inférieur au temps d'affichage d'image. Cette hypothèse est appropriée pour des applications de streaming vidéo interactives, telles que la visioconférence, qui requièrent des délais de transmission point–à–point très faibles. Cela est aussi approprié pour le streaming de vidéos stockées avec de faibles délais de playback et une forte interactivité.

Nous montrons que notre problème peut être formulé comme un Processus de Décision de Markov (MDP, pour "Markov Decision Process" en anglais), contraint, qui peut–être résolu par programmation linéaire [33, 58]. Le problème est formulé naturellement comme un MDP à horizon fini, l'horizon correspondant au nombre d'images dans le segment vidéo. Cependant, l'effort calculatoire associé à un MDP à horizon fini peut–être très coûteux pour un large horizon [11]. Par conséquent, nous utilisons plutôt des MDPs contraints à horizon infini. Ceux–ci ont des politiques optimales stationnaires et un coût de calcul plus faible. L'approximation d'horizon infini correspond à considérer des segments vidéos de longueur infinie.

Nous considérons le cas d'une vidéo encodée en une seule couche d'amélioration, pour lequel nous donnons une expression formelle des politiques de transmission optimales. Dans le cas général d'une vidéo encodée en plusieurs couches, et sans correction d'erreur, nous comparons les performances des politiques optimales qui tiennent compte de la dissimulation d'erreur, avec celles qui n'en tiennent pas compte lors de l'optimisation. Nous montrons les gains de qualité potentiels importants apportés , au serveur, par la prise en compte de la dissimulation d'erreur du décodeur. Nos simulations indiquent que les procédures complexes d'ordonnancement qui ne considèrent pas la dissimulation d'erreur dans le processus d'optimisation peuvent atteindre des performances qui sont considérablement inférieures aux performances optimales. Nous montrons que les performances alors atteintes sont similaires aux

performances d'une politique optimale simple qui ajoute/retranche une couche suivant un seul paramètre.

Nous comparons les résultats de politiques optimales avec correction d'erreur dynamique, qui autorisent de faire varier le nombre de codes correcteurs d'erreur associées à chaque couche, avec les résultats de politiques optimales avec correction d'erreur statique, qui doivent associer un nombre constant de codes correcteurs pour chaque couche. Nous trouvons que les politiques avec correction d'erreur statique atteignent des performances équivalentes aux politiques générales avec correction d'erreur dynamique.

Nous effectuons des simulations numériques pour étudier la validité de notre modèle à horizon infini pour la transmission optimale de vidéos à nombre fini d'images. Nous montrons que notre modèle donne de bonnes performances en terme de qualité et de débit pour des segments vidéo composés de plusieurs centaines d'images.

Toujours dans l'hypothèse d'une transmission avec connaissance parfaite de l'état du système, nous montrons comment ajouter à notre problème des contraintes de qualité autres que la distorsion moyenne. Nous prenons l'exemple du problème de minimiser la distorsion moyenne avec une contrainte sur le débit maximal et sur la variation maximale de distorsion entre deux images consécutives. Nous montrons que notre formulation en MDP contraint à horizon infini est particulièrement adaptée à l'ajout de contraintes de qualité supplémentaires, car le nouveau problème peut toujours être formulé par un programme linéaire.

Dans un deuxième temps, nous relaxons l'hypothèse selon laquelle le serveur peut observer l'état du récepteur. Dans ce cas notre MDP devient un MDP partiellement observable. Si de tels MDPs sont généralement difficiles à résoudre, le nôtre est plus facile à résoudre en raison de sa structure particulière : nous montrons qu'il revient à résoudre le MDP $\{A_{n-1}, A_n\}$, et nous comparons les performances obtenues avec ou sans connaissance de l'état du client. Les simulations indiquent que la perte de qualité induite par l'absence de connaissance de l'état du client est mineure, notamment grâce à la correction d'erreur qui permet d'avoir une meilleure assurance de l'état du client en fonction des décisions prises précédemment.

Chapitre 7 : Conclusions

Dans cette thèse, nous avons formulé de nouveaux cadres d'étude et résolu de nouveaux problèmes d'optimisation pour le streaming adaptatif au réseau et au contenu de vidéos codées en couches (codage conventionnel ou FGS). Nous avons analysé des politiques de transmission optimales et évalué leurs performances à partir de simulations avec des traces de réseau et des vidéos réelles.

Après avoir donné un aperçu de l'application de streaming vidéo sur Internet dans le Chapitre 2, nous avons conçu, dans le Chapitre 3, des politiques de streaming adaptatives équivalentes pour l'ajout

et retranchement de couches, et le changement de versions, dans le contexte d'une transmission TCP–compatible fiable. Nos simulations ont montré que, pour de faibles valeurs du surcoût en débit du codage en couches, la flexibilité du codage en couches par rapport aux versions permet au mécanisme d'ajout/retranchement de couches d'atteindre des performances similaires au changement de versions. Pour des valeurs moyennes du surcoût du codage en couches (supérieures à 10%), le changement de versions a une meilleure performance dans toutes les situations de bande passante.

L'utilisation de vidéos FGS permet à l'application de streaming de s'adapter aux variations de la bande passante de manière plus fine qu'avec le codage en couches conventionnel. Dans le Chapitre 4, nous avons présenté un nouveau cadre d'étude pour le streaming adaptatif de vidéos FGS stockées. Nous avons obtenu des politiques de transmission optimales, qui maximisent une mesure d'efficacité en bande passante et minimisent une mesure de variabilité du taux de codage. Nous avons ensuite développé une heuristique temps–réel qui est inspirée de la solution optimale. Des simulations à partir de traces Internet ont montré que notre heuristique atteint une performance quasi–optimale. Nous avons comparé le streaming sur TCP avec le streaming sur une connexion TFRC fiable. Des simulations ont indiqué que le streaming de vidéos FGS avec un stockage temporaire au client suffisant, ne nécessite pas des algorithmes TCP–compatibles à débit peu variable pour pouvoir obtenir une bonne qualité de vidéo. Enfin, nous avons présenté une implémentation de notre heuristique dans un système complet de streaming de vidéos MPEG–4 FGS.

Dans le Chapitre 5 nous nous sommes concentrés sur les caractéristiques de contenu de vidéos MPEG–4 FGS. Nous avons analysé les traces débit–distorsion de vidéos de longue durée, en utilisant des mesures de performance qui rendent compte de la qualité des vidéos décodées, tant au niveau des images individuelles qu'au niveau d'agrégation d'images (GoP, scène, etc.). Notre analyse suggère de prioriser le découpage de la couche FGS à la fin des plans–bit, et de prendre en considération les différents types d'images de la couche de base et la structure scénique de la vidéo. Nous avons étudié le streaming optimisé suivant les caractéristiques débit–distorsion à plusieurs degrés d'agrégation d'images. Les simulations à partir de nos traces ont indiqué que l'agrégation permet de réduire la complexité de l'optimisation au prix d'une faible perte de qualité globale. Cependant, l'agrégation arbitraire peut entraîner des détériorations de qualité par rapport à l'agrégation suivant les scènes visuelles. Par conséquent, nous préconisons d'effectuer le streaming vidéo en agrégeant des scènes visuelles.

Enfin, dans le Chapitre 6, nous avons étudié le streaming optimal sur des connexions avec pertes. Nous avons proposé un cadre d'optimisation unifié qui combine l'ordonnancement, la correction d'erreur et la dissimulation d'erreur au décodeur. Nous avons résolu le problème de minimiser la distorsion moyenne suivant un taux de transmission limité, en utilisant la théorie des Processus de Décision de Markov à gain moyen et à horizon infini. Nous avons effectué des simulations avec des vidéos MPEG–4

FGS. Considérant un canal à pertes de paquets avec une information parfaite de l'état du récepteur, nous avons montré que les procédures d'optimisation qui ne prennent pas en compte la dissimulation d'erreur dans le processus d'optimisation, peuvent atteindre des résultats qui sont bien plus faibles que les résultats optimaux. Nous avons montré que notre cadre d'optimisation à horizon infini donne de bonnes performances pour le cas de segments vidéos composés de plusieurs centaines d'images, et qu'il peut être limité à des politiques de transmission avec une redondance statique. Nous avons étendu notre problème d'optimisation en ajoutant une contrainte sur la variation moyenne de distorsion entre les images consécutives, afin d'illustrer que notre système permet de considérer des mesures de qualité autres que la distorsion moyenne. Dans le cas où l'information sur l'état du récepteur n'est pas toujours disponible au serveur, nous avons montré que notre système atteint de bonnes performances ; ainsi, notre approche est aussi appropriée pour des réseaux avec de forts délais de transmission.

Nous pouvons continuer le travail présenté dans cette thèse dans plusieurs directions, à la fois dans les domaines de l'adaptation au réseau et au contenu.

Dans le domaine de l'adaptation au réseau, nos systèmes peuvent être enrichis en considérant, dans la procédure d'optimisation, les retransmissions sélectives de paquets perdus. Alors qu'il existe un grand nombre d'études qui proposent des mécanismes de retransmission sélective pour l'audio ou la vidéo, le streaming optimal avec retransmissions n'a pas reçu suffisament d'attention. La modélisation et la résolution d'un tel problème sont en effet tout particulièrement difficiles. On pourrait également étendre nos procédures d'optimisation avec adaptation au réseau au cas de la transmission sur un réseau Internet avec qualité de service, tel DiffServ, et pour des configurations différentes de la configuration client–serveur classique.

Il y a d'autres orientations futures dans le domaine de l'adaptation au contenu. Nos algorithmes pourraient être testés avec des mesures de qualité objectives plus fiables que le PSNR, et notre travail pourrait bénéficier d'une segmentation temporelle plus fine que la segmentation en plans–séquence (ou "director's cuts"). Enfin, nos mécanismes d'adaptation au contenu pourraient être étendus en utilisant la segmentation spatiale des vidéos en objets, telle qu'elle est définie dans le standard MPEG–4. En combinant la scalabilité en objets, le codage en couches, et la segmentation en scènes, nous pourrions obtenir un nouveau système adaptatif selon lequel le serveur pourrait ajouter/retrancher des couches d'objets individuels, en fonction de l'importance sémantique de chaque objet dans la scène, ainsi que des propriétés débit–distorsion de chaque couche.

Bibliography

[1] DivX Codec. http://www.divx.com.

[2] Quicktime. http://www.apple.com.

[3] Real Media. http://www.realnetworks.com.

[4] Windows Media. http://www.microsoft.com.

[5] ANSI T1.801.03, Digital Transport of One–Way Video Signals — Parameters for Objective Performance Assessment, 1996.

[6] ISO/IEC 14496–1, Coding of Audio–Visual Objects: Systems. ISO/IEC JTC1/SC29/WG11 N2501, October 1998.

[7] ISO/IEC 14496–2, Coding of Audio–Visual Objects: Visual. ISO/IEC JTC1/SC29/WG11 N2502, October 1998.

[8] ISO/IEC 14496–2 / Amd X, Generic Coding of Audio–Visual Objects: Visual. ISO/IEC JTC1/SC29/WG11 N3095, December 1999.

[9] MPEG–4 Applications. ISO/IEC JTC1/SC29/WG11 N2724, March 1999.

[10] Report on MPEG–4 Visual Fine Granularity Scalability Tools Verification Tests. ISO/IEC JTC1/SC29/WG11 N4791, May 2002.

[11] ALTMAN, E. *Constrained Markov Decision Processes*. Chapman and Hall, 1999.

[12] ARAVIND, R., CIVANLAR, M. R., AND REIBMAN, A. R. Packet Loss Resilience of MPEG–2 Scalable Video Coding Algorithms. *IEEE Trans. on Circuits and Systems for Video Technology 6* (October 1996), 426–435.

[13] ASHMAWI, W., GUERIN, R., WOLF, S., AND PINSON, M. On the Impact of Policing and Rate Guarantees in DiffServ Networks: A Video Streaming Application Perspective. In *Proc. of SIGCOMM* (San Diego, CA, August 2001), pp. 83–95.

[14] ATKINSON, R., AND FLOYD, S. Internet Architecture Board Concerns & Recommendations Regarding Internet Research & Evolution. Internet Draft, February 2003.

[15] AVARO, O., ELEFTHERIADIS, A., HERPEL, C., RAJAN, G., AND WARD, L. MPEG–4 Systems: Overview. *Signal Processing: Image Communication 15* (2000), 281–298.

[16] BANSAL, D., AND BALAKRISHNAN, H. Binomial Congestion Control Algorithms. In *Proc. of INFOCOM* (Anchorage, AL, May 2001), pp. 631–640.

[17] BASSO, A., VARAKLIOTIS, S., AND CASTAGNO, R. Transport of MPEG–4 over IP/RTP. In *Proc. of Packet Video Workshop (PV)* (Cagliari, Italy, May 2000).

[18] BLAKE, S., BLACK, D., CARLSON, M., DAVIES, E., WAN G, Z., AND WEISS, W. An Architecture for Differentiated Services. IETF RFC 2475, December 1998.

[19] BOLOT, J.-C. End–to–End Packet Delay and Loss Behavior in the Internet. In *Proc. of SIGCOMM* (Ithaca, NY, September 1993), pp. 289–298.

[20] BOLOT, J.-C., FOSSE-PARISIS, S., AND TOWSLEY, D. Adaptive FEC–Based Error Cotrol for Internet Telephony. In *Proc. of INFOCOM* (New York City, March 1999), pp. 1453–1460.

[21] BOLOT, J.-C., AND TURLETTI, T. Experience with Control Mechanisms for Packet Video in the Internet. *ACM Computer Communications Review 28*, 1 (January 1998), 4–15.

[22] BRADEN, R., CLARK, D., AND SHENKER, S. Integrated Services in the Internet Architecture: an Overview. IETF RFC 1633, June 1994.

[23] CAI, H., SHEN, G., WU, F., LI, S., AND ZENG, B. Error Concealment for Fine Granularity Scalable Video Transmission. In *Proc. of the International Conference on Multimedia and Expo (ICME)* (Lausanne, Switzerland, September 2002).

[24] CHEN, R., AND VAN DER SCHAAR, M. Resource–Driven MPEG–4 FGS for Universal Multimedia Access. In *Proc. of the International Conference on Multimedia and Expo (ICME)* (Lausanne, Switzerland, August 2002), pp. 421–424.

[25] CHOU, P., MOHR, A., WANG, A., AND MEHROTRA, S. Error Control for Receiver–Driven Layered Multicast of Audio and Video. *IEEE Trans. on Multimedia 3*, 1 (March 2001), 108–122.

[26] CHOU, P. A., AND MIAO, Z. Rate–Distortion Optimized Sender–Driven Streaming over Best–Effort Networks. In *Proc. of the Workshop on Multimedia Signal Processing (MMSP)* (Cannes, France, October 2001), pp. 587–592.

[27] CHOU, P. A., AND MIAO, Z. Rate–Distortion Optimized Streaming of Packetized Media. *submitted to IEEE Trans. on Multimedia* (February 2001).

[28] CLARK, D., AND FANG, W. Explicit Allocation of Best–Effort Packet Delivery Service. *IEEE/ACM Trans. on Networking 6*, 4 (August 1998), 362–373.

[29] CÔTÉ, G., EROL, B., GALLANT, M., AND KOSSENTINI, F. H.263+: Video Coding at Low Bit Rates. *IEEE Trans. on Circuits and Systems for Video Technology 8*, 7 (November 1998), 849–866.

[30] DAWOOD, A. M., AND GHANBARI, M. Scene Content Classification From MPEG Coded Bit Streams. In *Proc. of the Workshop on Multimedia Signal Processing (MMSP)* (Copenhagen, Denmark, September 1999), pp. 253–258.

[31] DEMPSEY, B. J., LIEBEHERR, J., AND WEAVER, A. C. On Retransmission–Based Error Control for Continuous Media Traffic in Packet–Switching Networks. *Computer Networks and ISDN Systems 28*, 5 (March 1996), 719–736.

[32] DENARDO, E. V. *Dynamic Programming: Models and Applications*. Prentice–Hall, 1982.

[33] DERMAN, C. *Finite State Markovian Decision Processes*. Academic Press, New York, 1970.

[34] ENGELBRECHT, S. E., AND KATSIKOPOULOS, K. V. Planning with Delayed State Information. Tech. Rep. UM-CS-1999-030, University of Massachussets, Amherst, May 1999.

[35] FEAMSTER, N., BANSAL, D., AND BALAKRISHNAN, H. On the Interactions Between Layered Quality Adaptation and Congestion Control for Streaming Video. In *Proc. of the International Packet Video Workshop (PV)* (Kyongju, Korea, May 2001).

[36] FENG, W., KANDLUR, D., SAHA, D., AND SHIN, K. Adaptive Packet Marking for Maintaining End–to–End Throughput in a Differentiated–Services Internet. *IEEE/ACM Trans. on Networking 7*, 5 (October 1999), 685–697.

[37] FENG, W.-C., KRISHNASWAMI, B., AND PRABHUDEV, A. Proactive Buffer Management for the Streamed Delivery of Stored Video. In *Proc. of the ACM Multimedia Conference* (Bristol, U.K., September 1998).

[38] FITZEK, F., AND REISSLEIN, M. MPEG–4 and H.263 Video Traces for Network Performance Evaluation. *IEEE Network 15*, 16 (November 2001), 40–54.

[39] FLOYD, S. A Report on Recent Developments in TCP Congestion Control. *IEEE Communications Magazine* (April 2001).

[40] FLOYD, S., AND FALL, K. Promoting the Use of End–to–End Congestion Control in the Internet. *IEEE/ACM Trans. on Networking 7*, 4 (August 1999), 458–472.

[41] FLOYD, S., HANDLEY, M., PADHYE, J., AND WIDMER, J. Equation–Based Congestion Control for Unicast Applications. In *Proc. of SIGCOMM* (Stockholm, Sweden, August 2000), pp. 43–56.

[42] FLOYD, S., AND JACOBSON, V. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Trans. on Networking 1*, 4 (August 1993), 397–413.

[43] FLOYD, S., AND PAXSON, V. Difficulties in Simulating the Internet. *IEEE/ACM Trans. on Networking 9*, 4 (August 2001), 392–403.

[44] FRANCESCHINI, G. The Delivery Layer in MPEG–4. *Signal Processing: Image Communication 15* (2000), 347–363.

[45] FROSSARD, P. FEC Performance in Multimedia Streaming. *IEEE Communication Letters 5*, 3 (March 2001), 122–124.

[46] FROSSARD, P., AND VERSCHEURE, O. Joint Source/FEC Rate Selection for Quality–Optimal MPEG–2 Video Delivery. *IEEE Trans. on Image Processing 10*, 12 (December 2001), 1815–1825.

[47] GHANBARI, M. *Video Coding: an Introduction to Standard Codecs*. IEE Telecommunications Series, 1999.

[48] GUILLEMOT, C., CHRIST, P., WESNER, S., AND KLEMETS, A. RTP Payload Format for MPEG–4 with Flexible Error Resiliency. Internet Draft, March 2000.

[49] HARTANTO, F., KANGASHARJU, J., REISSLEIN, M., AND ROSS, K. W. Caching Video Objects: Layers vs. Versions ? In *Proc. of the International Conference on Multimedia and Expo (ICME)* (Lausanne, Switzerland, August 2002).

[50] HEINANEN, J., BAKER, F., WEISS, W., AND WROCLAWSKI, J. Assured Forwarding PHB group. IETF RFC 2597, June 1999.

[51] HERPEL, C., AND ELEFTHERIADIS, A. MPEG–4 Systems: Elementary Stream Management. *Signal Processing: Image Communication 15* (2000), 299–320.

[52] HORN, U., STUHLMÜLLER, K., LINK, M., AND GIROD, B. Robust Internet Video Transmission Based on Scalable Coding and Unequal Error Protection. *Signal Processing: Image Communication 15* (1999), 77–94.

[53] HSIAO, P.-H., KUNG, H. T., AND K-S., T. Video over TCP with Receiver–based Delay Control. In *Proc. of NOSSDAV* (Port Jefferson, New York, June 2001).

[54] HUANG, C.-L., AND LIAO, B.-Y. A Robust Scene–Change Detection Method for Video Segmentation. *IEEE Trans. on Circuits and Systems for Video Technology 11*, 12 (December 2001), 1281–1288.

[55] HUANG, H.-C., WANG, C.-N., AND CHIANG, T. A Robust Fine Granularity Scalability using Trellis–based Predictive Leak. *IEEE Trans. on Circuits and Systems for Video Technology 12*, 6 (June 2002), 372–385.

[56] JACOBSON, V. Congestion Avoidance and Control. In *Proc. of SIGCOMM* (Stanford, CA, August 1988), pp. 314–329.

[57] JACOBSON, V., NICHOLS, K., AND PODURI, K. An Expedited Forwarding PHB. IETF RFC 2598, June 1999.

[58] KALLENBERG, L. C. M. *Linear Programming and Finite Markovian Control Problems*. Mathematisch Centrum, Amsterdam, 1983.

[59] KALVA, H., HUARD, J.-F., TSELIKIS, G., ZAMORA, J., CHEOK, L.-T., AND ELEFTHERIADIS, A. Implementing Multiplexing, Streaming, and Server Interaction for MPEG–4. *IEEE Trans. on Circuits and Systems for Video Technology 9*, 8 (November 1999), 1299–1312.

[60] KARMARKAR, N. A New Polynomial Time Algorithm for Linear Programming. *Combinatorica*, 4 (1984), 373–395.

[61] KIM, T., AND AMMAR, M. A Comparison of Layering and Stream Replication Video Multicast Schemes. In *Proc. of NOSSDAV* (Port Jefferson, New York, June 2001), pp. 63–72.

[62] KIM, T., AND AMMAR, M. H. Optimal quality adaptation for MPEG–4 fine–grained scalable video. In *Proc. of INFOCOM* (San Francisco, CA, April 2003).

[63] KIMURA, J.-I., TOBAGI, F. A., PULIDO, J.-M., AND EMSTAD, P. J. Perceived Quality and Bandwidth Characterization of Layered MPEG–2 Video Encoding. In *Proc. of the SPIE International Symposium on Voice, Video and Data Communications* (Boston, MA, September 1999).

[64] KOENEN, B. MPEG–4 Multimedia for Our Time. *IEEE Spectrum* (February 1999), 26–33.

[65] KOPRINSKA, I., AND CARRATO, S. Temporal Video Segmentation: A Survey. *Signal Processing: Image Communication*, 16 (2001), 477–500.

[66] KRASIC, C., LI, K., AND WALPOLE, J. The Case for Streaming Multimedia with TCP. In *Proc. of the Workshop on Interactive Distributed Multimedia Sytems (IDMS)* (Lancaster, U.K., September 2001).

[67] KUROSE, J. F., AND ROSS, K. W. *Computer Networking: A Top–Down Approach Featuring the Internet*. Addison Wesley, 2001.

[68] LAOUTARIS, N., AND STAVRAKAKIS, I. Adaptive Playout Strategies for Packet Video Receivers with Finite Buffer Capacity. In *Proc. of the International Conference on Communications (ICC)* (Helsinki, Finland, June 2001), pp. 969–973.

[69] LE LÉANNEC, F., TOUTAIN, F., AND GUILLEMOT, C. Packet Loss Resilient MPEG–4 Vompliant Video Coding for the Internet. *Signal Processing: Image Communication 15* (1999), 35–56.

[70] LEE, Y.-C., KIM, J., ALTUNBASAK, Y., AND MERSEREAU, R. M. Layered Coded vs. Multiple Description Coded Video over Error–Prone Networks. *Signal Processing: Image Communication 18* (2003), 337–356.

[71] LI, W. Overview of Fine Granularity Scalability in MPEG-4 Video Standard. *IEEE Trans. on Circuits and Systems for Video Technology 11*, 3 (March 2001), 301–317.

[72] LI, W., LING, F., AND CHEN, X. Fine Granularity Scalability in MPEG–4 for Streaming Video. In *Proc. of the International Symposium on Circuits and Systems (ISCAS)* (Geneva, Switzerland, May 2000), pp. 299–302.

[73] LI, X., AMMAR, M., AND PAUL, S. Video Multicast over the Internet. *IEEE Network 13* (April 1999), 46–60.

[74] LIN, E., PODILCHUK, C., JACQUIN, A., AND DELP, E. A Hybrid Embedded Video Codec using Base Layer Information for Enhancement Layer Coding. In *Proc. of the International Conference on Image Processing (ICIP)* (Thessaloniki, Greece, October 2001), pp. 1005–1008.

[75] LIU, C.-C., AND CHEN, S.-C. S. Providing Unequal Reliability for Transmitting Layered Video Streams over Wireless Networks by Multi–ARQ Schemes. In *Proc. of the International Conference on Image Processing (ICIP)* (Kobe, Japan, October 1999), pp. 100–104.

[76] LOGUINOV, D., AND RADHA, H. On Retransmissions Schemes for Real–time Streaming in the Internet. In *Proc. of INFOCOM* (Anchorage, AL, May 2001), pp. 1310–1319.

[77] LOGUINOV, D., AND RADHA, H. End–to–End Internet Video Traffic Dynamics: Statistical Study and Analysis. In *Proc. of INFOCOM* (New York City, May 2002).

[78] LUI, J., LI, B., AND ZHANG, Y.-Q. Adaptive Video Multicast over the Internet. *IEEE Multimedia* (January–March 2003), 22–33.

[79] LUPATINI, G., SARACENO, C., AND LEONARDI, R. Scene Break Detection: a Comparison. In *Proc. of RIDE* (Orlando, Florida, February 1998), pp. 34–41.

[80] MATHY, L., EDWARDS, C., AND HUTCHISON, L. The Internet : A Global Telecommunications Solution. *IEEE Network* (July 2000).

[81] MCCANNE, S., JACOBSON, V., AND VETTERLI, M. Receiver–driven Layered Multicast. In *Proc. of SIGCOMM* (Stanford, CA, August 1996), pp. 117–130.

[82] MCCANNE, S. R. *Scalable Compression and Transmission of Internet Multicast Video*. PhD thesis, Univ. of California, Berkeley, 1996.

[83] MEDIAWARE SOLUTIONS. MyFlix 3.0. `http://www.mediaware.com.au`.

[84] MIAO, Z., AND ORTEGA, A. Optimal Scheduling for Streaming of Scalable Media. In *Proc. of Asilomar Conference on Signals, Systems, and Computers* (Pacific Grove, CA, November 2001), pp. 1357–1362.

[85] MIAO, Z., AND ORTEGA, A. Expected Run–time Distortion Based Scheduling for Delivery of Scalable Media. In *Proc. of International Conference of Packet Video (PV)* (Pittsburg, PA, April 2002).

[86] MICROSOFT. ISO/IEC 14496 Video Reference Software. Microsoft–FDAM1–2.3–001213.

[87] MOON, S. B., KUROSE, J., AND TOWSLEY, D. Packet Audio Playout Delay Adjustment: Performance Bounds and Algorithms. *Multimedia Systems 6* (1998), 17–28.

[88] NELAKUDITI, S., HARINATH, R. R., KUSMIEREK, E., AND Z.-L., Z. Providing Smoother Quality Layered Video Stream. In *Proc. of NOSSDAV* (Chapel Hill, North Carolina, June 2000).

[89] NONNENMACHER, J., BIERSACK, E., AND TOWSLEY, D. Parity–Based Loss Recovery for Reliable Multicast Transmission. In *Proc. of SIGOMM* (Cannes, France, September 1997), pp. 289–300.

[90] OLSSON, S., STROPPIANA, M., AND BAINA, J. Objective Methods for Assessment of Video Quality: State of the Art. *IEEE Trans. on Broadcasting 43*, 4 (December 1997), 487–495.

[91] ORTEGA, A. Variable Bit Rate Video Coding. In *Compressed Video over Networks*, M.-T. Sun and A. R. Reibman, Eds. Marcel Dekker, 2001, ch. 9, pp. 343–382.

[92] PADHYE, J., FIROIU, V., TOWSLEY, D., AND KUROSE, J. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proc. of SIGCOMM* (Vancouver, Canada, September 1998), pp. 303–314.

[93] PADHYE, J., KUROSE, J., TOWSLEY, D., AND KOODLI, R. A Model Based TCP–Friendly Rate Control Protocol. In *Proc. of NOSSDAV* (Basking Ridge, NJ, June 1999).

[94] PAPADOPOULOS, C., AND PARULKAR, G. M. Retransmission–Based Error Control for Continuous Media Applications. In *Proc. of NOSSDAV* (Zushi, Japan, April 1996).

[95] PAXSON, V. End–to–End Internet Packet Dynamics. *IEEE/ACM Trans. on Networking 7*, 3 (June 1999), 277–292.

[96] PERKINS, C., HODSON, O., AND HARDMAN, V. A Survey of Packet–Loss Recovery Techniques for Streaming Audio. *IEEE Network* (Sept–Oct 1998), 40–47.

[97] PODOLSKY, M., VETTERLI, M., AND MCCANNE, S. Limited Retransmission of Real–Time Layered Multimedia. In *Proc. of the Workshop on Multimedia Signal Processing (MMSP)* (Los Angeles, CA, December 1998), pp. 591–596.

[98] QIU, L., ZHANG, Y., AND KESHAV, S. Understanding the Performance of Many TCP Flows. *Computer Networks 37*, 3–4 (2001), 277–306.

[99] RADHA, H., CHEN, Y., PARTHASARATHY, K., AND COHEN, R. Scalable Internet Video Using MPEG–4. *Signal Processing: Image Communication 15* (September 1999), 95–126.

[100] RADHA, H., VAN DER SCHAAR, M., AND CHEN, Y. The MPEG–4 Fine–Grained Scalable Video Coding Method for Multimedia Streaming over IP. *IEEE Trans. on Multimedia 3*, 1 (March 2001), 53–68.

[101] RAJENDRAN, R. K., VAN DER SCHAAR, M., AND CHANG, S.-F. FGS+: Optimizing the Joint SNR–Temporal Video Quality in MPEG–4 Fine Grained Scalable Coding. In *Proc. of the International Symposium on Circuits and Systems (ISCAS)* (Scottsdale, Arizona, May 2002), pp. 445–448.

[102] REIBMAN, A. R., BOTTOU, U., AND BASSO, A. DCT–Based Scalable Video Coding with Drift. In *Proc. of the International Conference on Image Processing (ICIP)* (Thessaloniki, Greece, October 2001), pp. 989–992.

[103] REIBMAN, A. R., JAFARKHANI, H., WANG, Y., ORCHARD, M. T., AND PURI, R. Multiple–Description Video Coding Using Motion–Compensated Temporal Prediction. *IEEE Trans. on Circuits and Systems for Video Technology 12*, 3 (March 2002), 193–204.

[104] REIBMAN, A. R., WANG, Y., QIU, X., JIANG, Z., AND CHAWLA, K. Transmission of Multiple Description and Layered Video over an EGPRS Wireless Network. In *Proc. of the International Conference on Image Processing (ICIP)* (Vancouver, Canada, October 2000), pp. 136–139.

[105] REISSLEIN, M., LASSETTER, J., RATNAM, S., LOTFALLAH, O., FITZEK, F., AND PANCHANATHAN, S. Traffic and Quality Characterization of Scalable Encoded Video: A Large–Scale Trace-Based Study. Tech. rep., Dept. of Electrical Eng. Arizona State University, December 2002.

[106] REJAIE, R., ESTRIN, D., AND HANDLEY, M. Quality Adaptation for Congestion Controlled Video Playback over the Internet. In *Proc. of SIGCOMM* (Cambridge, September 1999), pp. 189–200.

[107] REJAIE, R., HANDLEY, M., AND ESTRIN, D. RAP: An End-to-End Rate-Based Congestion Control Mechanism for Realtime Streams in the Internet. In *Proc. of INFOCOM* (New York, March 1999), pp. 1337–1345.

[108] REJAIE, R., AND REIBMAN, A. Design Issues for Layered Quality–Adaptive Internet Video Playback. In *Proc. of the Workshop on Digital Communications* (Taormina, Italy, September 2001), pp. 433–451.

[109] RHEE, I. Error Control Techniques for Interactive Low–bit Rate Video Transmission over the Internet. In *Proc. of SIGCOMM* (September 1998), pp. 290–301.

[110] ROSENBERG, J., QIU, L., AND SCHULZRINNE, H. Integrating Packet FEC into Adaptive Voice Playout Buffer Algorithms on the Internet. In *Proc. of INFOCOM* (Tel Aviv, Israel, March 2000), pp. 1705–1714.

[111] ROSS, K. W. Randomized and Past–Dependent Policies for Markov Decision Processes With Multiple Constraints. *Operations Research 37*, 3 (May–June 1989), 474–477.

[112] SAHU, S., NAIN, P., TOWSLEY, D., DIOT, C., AND FIROIU, V. On Achievable Service Differentiation with Token Bucket Marking for TCP. In *Proc. of SIGMETRICS* (Santa Clara, CA, June 2001).

[113] SAPARILLA, D., AND ROSS, K. W. Optimal Streaming of Layered Video. In *Proc. of INFOCOM* (Tel Aviv, Israel, March 2000), pp. 737–746.

[114] SAPARILLA, D., AND ROSS, K. W. Streaming Stored Continuous Media over Fair–Share Bandwidth. In *Proc. of NOSSDAV* (Chapel Hill, North Carolina, June 2000).

[115] SAW, Y.-S. *Rate Quality Optimized Video Coding*. Kluwer Academic Publishers, 1999.

[116] SEN, S., REXFORD, J. L., DEY, J. K., KUROSE, J. F., AND TOWSLEY, D. F. Online Smoothing of Variable–Bit–Rate Streaming Video. *IEEE Trans. on Multimedia 2*, 1 (March 2000), 37–48.

[117] SERVETTO, S. D. *Compression and Reliable Transmission of Digital Image and Video Signals*. PhD thesis, University of Illinois, May 1999.

[118] SISALEM, D., AND SCHULZRINNE, H. The Loss–Delay Based Adjustment Algorithm: A TCP–Friendly Adaptation Scheme. In *Proc. of NOSSDAV* (Cambridge, U.K., July 1998).

[119] SREENAN, C. J., CHEN, J.-C., AGRAWAL, P., AND NARENDRAN, B. Delay Reduction Techniques for Playout Buffering. *IEEE Trans. on Multimedia 2*, 2 (June 2000), 88–100.

[120] SZE, H. P., LIEW, S. C., AND LEE, Y. B. A packet–loss–recovery scheme for continuous–media streaming over the internet. *IEEE Communications Letters 5*, 3 (March 2001), 116–118.

[121] TAN, W.-T., AND ZAKHOR, A. Real–Time Internet Video Using Error Resilient Scalable Compression and TCP–Friendly Transport Protocol. *IEEE Trans. on Multimedia 1*, 2 (June 1999), 172–186.

[122] TAN, W.-T., AND ZAKHOR, A. Video Multicast Using Layered FEC and Scalable Compression. *IEEE Trans. on Circuits and Systems for Video Technology 11*, 3 (March 2001), 373–386.

[123] TANG, X., AND ZAKHOR, A. Matching Pursuits Multiple Description Coding for Wireless Video. *IEEE Trans. on Circuits and Systems for Video Technology 12*, 6 (June 2002), 566–575.

[124] TUNG, Y.-S., WU, J.-L., HSIAO, P.-K., AND HUANG, K.-L. An Efficient Streaming and Decoding Architecture for Stored FGS Video. *IEEE Trans. on Circuits and Systems for Video Technology 12*, 8 (August 2002), 730–735.

[125] TURLETTI, T., FOSSE-PARISIS, S., AND BOLOT, J.-C. Experiments With a Layered Transmission Scheme over the Internet. Tech. Rep. RR–3296, INRIA, France, November 1997.

[126] VAN DER SCHAAR, M., AND LIN, Y.-T. Content–based Selective Enhancement for Streaming Video. In *Proc. of the International Conference on Image Processing (ICIP)* (Thessaloniki, Greece, October 2001), pp. 977–980.

[127] VAN DER SCHAAR, M., AND RADHA, H. A Hybrid Temporal–SNR Fine–Granular Scalability for Internet Video. *IEEE Trans. on Circuits and Systems for Video Technology 11*, 3 (March 2001), 318–331.

[128] VAN DER SCHAAR, M., AND RADHA, H. Unequal Packet Loss Resilience for Fine–Granular–Scalability Video. *IEEE Trans. on Multimedia 3*, 4 (December 2001), 381–393.

[129] VAN DER SCHAAR, M., AND RADHA, H. Adaptive Motion–Compensation Fine–Granular–Scalability (AMC–FGS) for Wireless Video. *IEEE Trans. on Circuits and Systems for Video Technology 12*, 6 (June 2002), 360–371.

[130] VICISANO, L., RIZZO, L., AND CROWCROFT, J. TCP–like Congestion Control for Layered Multicast Data Transfer. In *Proc. of INFOCOM* (San Francisco, CA, March–April 1998), pp. 996–1003.

[131] VICKERS, B., ALBUQUERQUE, C., AND SUDA, T. Source Adaptive Multi–Layered Multicast Algorithms for Real–Time Video Distribution. *IEEE/ACM Trans. on Networking 8*, 6 (2000), 720–733.

[132] VIERON, J., TURLETTI, T., HENOCQ, X., GUILLEMOT, C., AND SALAMATIAN, K. TCP–Compatible Rate Control for FGS Layered Multicast Video Transmission Based on a Clustering Algorithm. In *Proc. of the International Symposium on Circuits and Systems (ISCAS)* (Scottsdale, Arizona, May 2002), pp. 453–456.

[133] VQEG. Video Quality Experts Group: Current Results and Future Directions. In *Proc. of SPIE Visual Communications and Image Processing* (Perth, Australia, June 2000), pp. 772–753.

[134] WANG, J., AND CHUA, T.-S. A Framework for Video Scene Boundary Detection. In *Proc. of ACM Multimedia Conference* (Juan-les-Pins, France, December 2002).

[135] WANG, Q., XIONG, Z., WU, F., AND LI, S. Optimal rate allocation for progressive fine granularity scalable video coding. *IEEE Signal Processing Letters 9*, 2 (February 2002), 33–39.

[136] WANG, Y., CLAYPOOL, M., AND ZUO, Z. An Emirical Study of RealVideo Performance Across the Internet. In *Proc. of ACM Sigcomm Internet Measurement Workshop* (San Francisco, CA, November 2001).

[137] WANG, Y., AND ZHU, Q.-F. Error Control and Concealment for Video Communications: A Review. *Proceedings of the IEEE 86*, 5 (May 1998), 974–997.

[138] WIDMER, J., DENDA, R., AND MAUVE, M. A Survey on TCP–Friendly Congestion Control. *IEEE Network 15*, 3 (May–June 2001), 28–37.

[139] WILLINGER, W., AND PAXSON, V. Where Mathematics Meets the Internet. *Notices of the American Mathematical Society 45*, 8 (August 1998), 961–970.

[140] WINKLER, S. *Vision Models and Quality Metrics for Image Processing Applications*. PhD thesis, Swiss Federal Institute of Technology, 2000.

[141] WU, D., HOU, Y. T., AND ZHANG, Y.-Q. Transporting Real–Time Video over the Internet: Challenges and Approaches. *Proceedings of the IEEE 88*, 12 (December 2000), 1855–1875.

[142] WU, D., HOU, Y. T., ZHU, W., ZHANG, Y.-Q., AND PEHA, J. M. Streaming Video over the Internet: Approaches and Directions. *IEEE Trans. on Circuits and Systems for Video Technology 11*, 3 (March 2001), 1–20.

[143] WU, F., LI, S., AND ZHANG, Y.-Q. A Framework for Efficient Progressive Fine Granularity Scalable Video Coding. *IEEE Trans. on Circuits and Systems for Video Technology 11*, 3 (March 2001), 332–344.

[144] YAJNIK, M., MOON, S., KUROSE, J., AND TOWSLEY, D. Measurement and Modelling of the Temporal Dependence in Packet Loss. In *Proc. of INFOCOM* (New York City, March 1999), pp. 345–352.

[145] YANG, Y., KIM, M., AND LAM, S. Transient Behaviors of TCP–Friendly Congestion Control Protocols. In *Proc. of INFOCOM* (Anchorage, AL, May 2001), pp. 1716–1725.

[146] ZHANG, Q., ZHU, W., AND ZHANG, Y.-Q. Resource Allocation for Multimedia Streaming over the Internet. *IEEE Trans. on Multimedia 3,* 3 (September 2001), 339–335.

[147] ZHANG, R., REGUNATHAN, S. L., AND ROSE, K. End–to–end Distortion Estimation for RD–based Robust Delivery of Pre–compressed Video. In *Proc. of Asilomar Conference on Signals, Systems and Computers* (Pacific Grove, CA, October 2001).

[148] ZHANG, Y., DUFFIELD, N., PAXSON, V., AND SHENKER, S. On the Constancy of Internet Path Properties. In *Proc. of ACM SIGCOMM Internet Measurement Workshop* (San Francisco, CA, November 2001).

[149] ZHANG, Y., PAXSON, V., AND SHENKER, S. The Stationarity of Internet Path Properties: Routing, Loss, and Throughput. Tech. rep., ACIRI, May 2000.

[150] ZHAO, L., KIM, J.-W., AND KUO, C.-C. J. Constant Quality Rate Control for Streaming MPEG–4 FGS Video. In *Proc. of the International Symposium on Circuits and Systems (ISCAS)* (Scottsdale, Arizona, May 2002), pp. 544–547.