Resilient Video Coding for Wireless and Peer-to-peer Networks

Resilient Video Coding for Wireless and Peer-to-peer Networks

Proefschrift

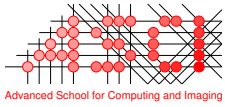
ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. dr. ir. J. T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op 2 oktober 2007 om 10.00 uur
door Jacco Reynoud TAAL
elektrotechnisch ingenieur
geboren te Rotterdam.

Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. ir. R. L. Lagendijk

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. ir. R. L. Lagendijk,	Technische Universiteit Delft, promotor
Prof. dr. ir. P. F. A. Van Mieghem	Technische Universiteit Delft
Prof. dr. ir. H. J. Sips	Technische Universiteit Delft
Prof. dr. ir. C. H. Slump	Universiteit Twente
Prof. dr. ir. P. H. N. de With	Technische Universiteit Eindhoven
Univ.-Prof. Dr.-Ing. Eckehard Steinbach	Technische Universität München
Dr. A. Reibman	AT&T Research

This document is typeset with the LaTeX `memoir` typesetting system.
The photo on the cover is taken in April 2006 in the pedestrians tunnel connecting the old and new part of Shanghai.



This work was carried out in the ASCI graduate school.
ASCI dissertation series number 155

ISBN: 978-90-9022238-7

Voor Jenneke

Preface

The work that resulted in this thesis was carried out at Delft University of Technology in three different projects. In each of these project video coding was part of the research, but every time the emphasis was put on a different viewpoint towards video coding was taken.

The first project, Ubiquitous Communications (UBICOM), was totally funded by the Delft University of Technology. The UBICOM scenario was a low-power wearable augmented reality system where a virtual world could be projected onto the real world, using semi transparent virtual reality helmet.

In the Context Aware User and Terminal (CACTUS) project a modern user, equipped with wearable devices was central. Devices, worn by the user or integrated in rooms, buildings, cars, etc, will be able to communicate with each other in an *ad hoc* manner. To support his tasks, his personal device communicates and negotiates on his behalf user with other devices such as screens, input devices and for instance vending machines. CACTUS, as sub project of the FREEBAND project, was funded by the Ministry of Economic Affairs in The Netherlands.

In the I-SHARE project, we focused on sharing resources and data among devices, people and groups of people. In this scenario, Internet wireless networks and peer-to-peer (P2P) networks play an important role to interconnect devices and to facilitate sharing. I-SHARE was also funded by the Ministry of Economic Affairs, as a sub project of the FREEBAND project.

This thesis concentrates on the aspect of video compression in these projects. Each project had a different viewpoint on video compression. Not only the scenario in which compression was used differed, but also the chosen solutions to implement video compression. Although we concentrate on video compression, we approach the task of video compression from the environment in which it has to operate, namely a networked device, from which different constraints are imposed on the video coder.

J. R. Taal, Delft, March 2007.

Acknowledgements

Although doing a Ph.D. track was a decision of my own, I know many people have supported me in pursuing it. First I would like to thank my promotor Inald Lagendijk for his support and patience with me. I always enjoyed the deep discussions we had. Thanks for giving me this opportunity, working with you is a privilege.

I thank my roommates, I have had many in the projects: (in order of appearance) Maarten Ditzel and Hylke van Dijk, both for taking care of me as a fresh student on the 16th floor. Hylke for the many interesting and abstract discussions we had about our work. Maarten, thanks for the really nice time, the volleyball games and our friendship. Jan, thanks for the fun we had on the 9th floor; too bad you could not finish your thesis. Ronald, we share a sense of humour. It was really nice to have you in the same room on the 10th floor for these years. Your have an interesting research topic and wish you luck in writing your thesis, which you probably do with ease. Jenneke, the first woman in the same office, it was a privilege. Your enthusiasm has always brought cheer in our room.

The people that I perhaps owe most, were not my roommates. Ivaylo, thank you. For the work we done together and the many papers we wrote together. I will never forget our first trip to China. Johan, your enthusiasm worked on me. We had and still have great ideas, some of them are now being materialized.

The Information and Communication Theory Group has always been inspiring and competitive, but also very lively and fun. I thank all my colleagues in the group. This is the best group of all.

I like to thank Henk Sips and Dick Epema for your support and cooperation during the projects.

During my journey up and down the building, I have met and worked with many people. I would like to thank all that I have missed here.

I like to thank Prof. He of Tsinghua University, Beijing, China, for hosting me and Ivaylo in 2003. It has been one of the most interesting experiences I had. I like to thank Chen Zhibo for the really nice cooperation on our paper. I wish you all good luck in your career.

My best friends, Maurice and Michiel. The holidays and weekends we spent

together are unforgettable and a big part of my life. They helped me through the process. I hope our lives will stay entangled forever.

I thank Rob Becqué, who sadly could not finish his thesis, for his inspiration.

Finally, I thank my family for their love and support.

Roy and Cindy, I am very proud of the life you have build up together with your children Jiska and Minke.

Mom, I suspect you have always known that I could do this. You always supported me in whatever step I took. You are the best mom in the world.

Contents

List of Tables

List of Figures

List of Abbreviations

802.11	IEEE 802.11 WiFi Standard
ADSL	Asynchronous Digital Subscriber Loop
ALM	application level multicast
AMDC	asymmetric multiple description coding
APP	application
ARC	Adaptive Resource Contracts
ARQ	automatic resent query
AVC	Advanced Video Codec (a.k.a. MPEG4/AVC)
BER	bit-error rate
B-frame	bidirectionally predicted frame
CACTUS	Context Aware User and Terminal
CBR	constant bit rate encoding
CDN	content delivery network
CPU	central processing unit
CRC	cyclic redundancy check
CSI	channel state information
CSP	channel state predictor
DC	direct current
DCT	Discrete Cosine Transform
DPCM	Differential Pulse Code Modulation
DVB	Digital Video Broadcasting
DVD	Digital Versatile Disc
EM	electro-magnetic
EZW	Embedded Zerotree Wavelet
FEC	forward error correction
FER	frame-error rate
FGS	Fine Granular Scalability
GOP	group of pictures
H.263	ITU-T H.263 Video Coding Standard
H.264	ITU-T H.264 Video Coding Standard
HD	high definition
HDTV	high definition television
I-frame	intra coded frame
IP	Internet Protocol
JPEG2000	JPEG2000 Image Compression Standard
JPEG	Joint Picture Experts Group
LC	layered coding
LC-LEC	layered coding with lateral error correction
LC-UEP	layered coding and unequal error protection
LEC	lateral error correction
LSB	least significant bit
MAC	media access control
MC	motion compensation
MCTF	Motion Compensated Temporal Filtering
MDC	multiple description coding
SBMDC	source based multiple description coding
MDCT	multiple description correlating transform
MD-FEC	multiple description coding using forward error correction

MD	multiple description
MDSQ	multiple description scalar quantization
MDVQ	multiple description vector quantization
ME	motion estimation
MPEG	Motion Picture Experts Group
MPEG-2	Motion Picture Experts Group Video Coding Standard II
MPEG-4	Motion Picture Experts Group Video Coding Standard IV
MSB	most significant bit
MSE	mean square error
MSP	medium sharing predictor
NAL	network adaptation layer
NIC	network interface card
NW	network
OSI	Reference Model for Open Systems Interconnection
P2P	peer-to-peer
P2P-TV	Peer-to-Peer Television
PCM	Pulse Code Modulation
pdf	probability density function
P-frame	inter coded frame
pmf	probability mass function
PSNR	peak signal-to-noise-ratio
QoS	quality of service
RAVC	real-time adaptive video coding
RD	rate-distortion
RTCP	Realtime Transport Control Protocol
RTP	Realtime Transmission Protocol
SDC	single description coding
SDTV	Standard Definition Television
SM	shared medium
SNR	signal-to-noise-ratio
SPIHT	Set Partitioning in Hierarchical Trees
SSI	signal strength indication
SSIA	signal strength indication of acknowledged frames
SRI	successive refinement of information
SVC	Scalable Video Codec
TCP/IP	Internet Protocol (See IP, TCP)
TCP	Transport Control Protocol
TM.5	Test Model 5 rate control algorithm
UBICOM	Ubiquitous Communications
UDP	User Datagram Protocol
UEP	unequal error protection
UMTS	Universal Mobile Telecommunications System
VBR	variable bit rate encoding
VC	video coder or Video Codec
VLC	variable length coding
VRCA	video rate control algorithm
VSM	Video Streaming Model
WiFi	Wireless Fidelity (IEEE 802.11)
YUV	a representation of pictures by one Luminance dimension and two color dimensions

One

Introduction

Video compression is a technology that is implemented in a lot of everyday tools, toys and appliances. It is found in products ranging from high definition television to Digital Versatile Disc (DVD) player and digital hard disk video recorder, and from computer to personal digital assistant and mobile telephone. In the not too distant future, even traditionally passive appliances such as refrigerators will have a screen capable of showing video. The environment in which video coding takes places varies greatly. Still the user expects the best possible quality in all circumstances. The TV watcher should for instance be able to keep watching his favourite program when he walks from the living room to the kitchen and then to the bedroom. Each display has a different resolution and may have different connections to the in-home network. In most of these cases the owner will not actually realize that video compression and adaptation is taking place. The functionality is hidden in software and hardware.

Because video compression plays such an important role in everyday communications of everybody, this enabling technology should not be visible for its users. Users should be shielded from complex configurations and choices regarding video and compression format, even when the circumstances in which the video coder has to perform are changing and not known *a priori*. Traditional video compression applications such as Digital Video Broadcasting over Satellites and Digital Video Cameras, were designed with specific bit rates in mind and for specific network conditions. In current day applications, however, the conditions in which they are used are often heterogeneous and very dynamic. The video coder should be designed in such a way that it can deal with uncertainties in the environment (e.g. the network, the device platform). This means that the video coder should be adaptive and the produced video stream should be resilient to changes in the network.

Recent developments in video compression standards, yield better compres-

sion ratios at wider bit rate ranges. Furthermore new paradigms such as scalable (layered) compression and multiple description coding have received much attention and are now resulting in implementations and standardization.[1]

Besides distribution via DVD and digital-TV, recently, distribution via peer-to-peer (P2P) networks is now also becoming popular, although currently mostly by simple downloading. More than fifty percent of the Internet backbone traffic is already P2P traffic. P2P also offers the possibility to stream video, due to the very flexible way P2P communicates between peers and the efficient use of bandwidth in the network. Especially for offering less-popular content (the Long Tail [4]), P2P networks may become a cheap and viable alternative to Television broadcasting over cable or Internet.

During our research several video streaming scenarios were investigated. On one hand, these scenarios shared a common target: maximization of the average video quality as received by the clients. We require information about the network behavior in order to design or adapt the video coder. The application sets constraints on the video streaming system. For instance, a delay constraint has impact on all parts of the video streaming system (application, video coder, transmission protocols and network). On the other hand, in each solution we used a different type of video compression and a different type of cooperation among video coder network adaptation layer (NAL) and network layers.

In this thesis we first present a framework such that each streaming scenario is a specific case of the framework. We use this framework, called the Video Streaming Model, to explain the concept of cooperation of layers and to describe the parameters that play an important role in the scenarios. In the second part of the thesis, we present five papers with different approaches to implement error-resilient or adaptive video compression algorithms in different scenarios.

In this introduction we first describe recent advances in the area of video compression. These advances have greatly increased the number of way to implement a video streaming system. After that we formulate the problem description of the problems to which we try to contribute in this thesis. We give an overview of the structure of this thesis in more detail. Finally we summarize the contributions that this work brings to the field of video compression.

1.1 Advances in Video Compression

The advances made in recent years in the field of video compression have enabled many streaming applications. Furthermore, the number of ways to im-

[1] With "scalable" is meant here that the produced bitstream is constructed and designed in such a way that it is applicable for multiple or a range of bitrates, resolutions, framerates, or, in general, devices

plement a video streaming system has increased. This has made it possible to find a tailored solution for any specific streaming scenario. We classify these advances as follows:

1. Higher compression ratio,
2. New encoding paradigms,
3. Control of the encoding parameters.

The new encoding standards H.264 Advanced Video Codec (AVC) offers a bit rate saving of around 40% with respect to Motion Picture Experts Group Video Coding Standard IV (MPEG-4) and of around 60% respect to Motion Picture Experts Group Video Coding Standard II (MPEG-2) [95, 94]. Although increased ratio means that lower bit rates are required to obtain the same quality, this comes with an increased complexity of both encoder and decoder. These new standards are able to maintain acceptable picture qualities at low bit rates, so that the bit-rate range at which the encoders are useful has increased.

Although MPEG-2 and MPEG-4/Fine Granular Scalability (FGS) [42] support some form of scalability, it was seldomly used because the enhancement layers suffered from high compression loss, since temporal correlation was not exploited in the enhancement layers. The H.264/Scalable Video Codec (SVC) [64], soon to be standardized, is the first standard to offer real spatial, temporal and fine-granular SNR scalability at acceptable costs in terms of loss of compression ratio.

Another paradigm that received much academic attention is multiple description coding (MDC) [54, 34]. MDC is an encoding technique in which more than one description is used to describe the same data. The descriptions have to be sufficiently different to increase the reconstruction quality whenever more of these mutually enhancing descriptions are received by the decoder. Especially, the error-resilience and scalability features make MDC applicable in error-prone and heterogeneous environments. MDC may become the primary encoding paradigm when video streaming over P2P networks gains a strong foothold. However, this paradigm barely left the academic arena and did not lead yet to a service ready for the consumer market.

The way the encoder setting are controlled also received academic attention. First of all, variable bit rate (VBR) and constant bit rate (CBR) encoding are used to generate a bit stream with a constant quality or constant rate. VBR and CBR are used for making DVDs and for Digital Video Broadcasting (DVB) and are targeted to transmission and storage media with well-known constant size and capacity[72]. In scenarios where network capacity is dynamic, the encoding rate should adapt to these network rate changes. This happens either by adapting the encoding rate during real-time encoding, or by intelligently selecting after encoding which parts to transmit and which parts to skip in order to make real time play out possible. In both cases, the performance of the network should be known in order to make the right choices.

1.2 Problem Description

A video coder cannot be viewed independently from the rest of system in which a video is encoded, transmitted, streamed or stored, decoded and displayed. In this thesis we look at different streaming applications. We define (video) streaming as a continuous transmission of (video) data, such that after some given delay a continuous real-time play out is possible. For different scenarios, which are described by the number of clients, the type of network and other system requirement, this will inevitably lead to different choices regarding the encoding algorithm.

For a given scenario a specific video streaming system can be designed that takes the scenario constraints and uncertainties into account. With the recent advances in video coding in mind, we now have several options to implement the video coder and how to arrange cooperation with OSI[2] transmission layers. We have split up the design choices in the following way:

1. Video coding paradigm,
2. Encoding algorithm,
3. Type of cooperation with Reference Model for Open Systems Interconnection (OSI)-stack layers,
4. Encoder settings.

The first three choices are made while designing the system. The encoder settings can also be chosen at design time, but then no adaptation to a dynamic environment can be done.

With respect to choosing the video coding paradigm, we investigate the following options in Chapter 3:

1. Real-time adaptive video compression (Section 3.2),
2. Scalable Coding (Section 3.3),
3. Multiple Description Coding (Section 3.4).

The choice of encoding paradigm is not independent of the actual video compression method chosen, since not all standards have a scalability extension and MDC video compression is not standardized at all. Regarding the video compression standard, we have used H.263, MPEG-4, H.264 and Dirac in our experiments.

With regard to cooperation between layers we explore the following options

[2] OSI: Reference Model for Open Systems Interconnection. [78]

1. quality of service (QoS) cooperation: performing QoS negotiations in order to fully adapt all layers to changing conditions (Chapters 5 and 4).

2. Bottom-up cooperation: informing higher layers about the state of the lower layers (network and NAL) (Chapters 6).

3. No real-time cooperation: resulting in each layer operating independently in a best-effort fashion. The expected average network conditions are only taken into account at forehand. (Chapters 7 and 8).

Finally, The encoder settings are settings such as quantization-step size, prediction scheme and motion-estimation parameters, depending on the chosen encoding implementation or standard. Given the heterogeneous and changing environments of most applications, we wish to adapt the encoder settings to up-to-date information about the network characteristics.

The problem that is addressed in this thesis is how to adapt and control the video coder such that it performs optimally in the context of the video streaming system scenarios.

1.3 Organization and Scope

This thesis consists of two parts. The first part describes a framework of the scenarios presented in the second part. In Part I the Video Streaming Model and three solutions to implement a context-aware video coder are presented. The second part consists of five published articles that are all centered around a video compression method which operates in a heterogeneous and changing environment.

Part I. Network Aware Video Coding

Chapter 2
Video Streaming Model
We present the Video Streaming Model (VSM) that generalizes the scenarios presented in Part II. The VSM is an abstract model of a video streaming system. In the VSM, the application, video coder and network adaptation layer (NAL) form separate functional blocks that cooperate. Between the blocks interface parameters are defined that express the context[3] in which the functional blocks have to operate. Whether these interface parameters are actually exchanged or negotiated depends on the chosen cooperation model.

Chapter 3
Network Aware Video Coding Strategies

[3] We define the *context*, as the state of the other layers that the video coder has to cooperate with, for instance the network and the application, and the *state* of the video source. The context may vary over time as for instance the network behavior changes.

Three different network aware video coding strategies are discussed, namely, real-time adaptive video coding, scalable coding and multiple description coding. Each strategy is 'Network Aware', in the sense that network characteristics and behavior are taken into account to adapt the encoder settings.

Part II. Streaming Scenarios

Scenario 1. Low-latency real-time streaming of video to low-power mobile devices. *In a changing and error-prone environment, the encoder and decoder have to deal with many constraints such as bandwidth and end-to-end delay while being resilient to channel errors. Since we are dealing with low-power devices, power consumption is included in the design and taken into account as a scarce resource. The video compression algorithm should be able to adapt to all variations in the context while keeping an optimal quality level and obeying the constraints.*

Chapter 4
End-to-end optimization of mobile video streaming using QoS
In this chapter we investigate the real-time adaptive video coding paradigm with QoS cooperation. The Adaptive Resource Contracts – QoS system is used to negotiate all resources and performances with a network access layer and an application. In the experiment, we study the end-to-end behavior of the ARC video streaming system. A network is simulated with changing properties. The NAL and the video coder adapt their internal parameters, in order to give an optimal quality while obeying the constraints defined by network characteristics and application.

Chapter 5
Error Resilient Video Compression using Behavior Models
In this chapter, a low-delay texture encoder is constructed from a JPEG2000 (single frame) encoder, augmented with inter-prediction without motion compensation. To be able to adapt to the network changes we devise behavior models of this encoder. An end-to-end distortion metric is presented that takes the the network characteristics and video characteristics into account.

Scenario 2. Low-latency real-time streaming of video over Wireless Fidelity (IEEE 802.11) (WiFi) networks. *For streaming video over WiFi connections, the IEEE 802.11 WiFi Standard (802.11) media access control (MAC) layer is modified to support streaming applications and to give status updates to the video coder. This scenario demands a tight cooperation between video coder and 802.11 MAC-layer in order to achieve a low latency while operating in a dynamic and error-prone wireless environment.*

Chapter 6
Optimized Video-Streaming over 802.11 by Cross-Layer Signaling
By adapting the 802.11 modulation scheme, the modified MAC algorithm increases the reliability of the link at the cost of a lower transmission speed. Furthermore, it informs the real-time video coder of the current and expected network statistics, such that the video coder can adapt its target rate setting. With this scheme it is possible to do very low latency streaming. We present the results of experiments where a real time encoded video signal is streamed over a wireless link under different and varying conditions.

> ***Scenario 3. Streaming of video to many clients using P2P networks.*** *In this scenario, the network offers high bandwidths and a flexible P2P communication structure between server and clients. The varying capacities and congestion may result in long delays, which are not desired in a streaming application. A solution with MDC and P2P communication offers the needed error resilience and scalability to deal with congestion and rate diversity of the clients.*

Chapter 7
Fair Rate Allocation of Scalable MDC for Many Clients
In P2P networks with many connected clients, we have to deal with heterogeneous client bandwidths. Streaming over P2P networks, with delay constraints, demands an error-resilient video coding. Also, P2P networks offers the possibility to have different overlay networks. There may be different paths from source to destinations. By using MDC, different descriptions are streamed over different overlay networks. By doing so, packet losses and peer failure will only have a limited effect. We use multiple description coding using forward error correction (MD-FEC) to generate descriptions. This chapter is about how to find a good rate allocation, while taking into account the packet loss rates on channels and different capacities of each client. We introduce fairness to find a fair trade off between increasing quality for a number of clients while decreasing the quality for others.

Chapter 8
Asymmetric MDC using Layered Coding and Lateral Error Correction
When different (P2P) overlay networks offer different bandwidths and different reliabilities (packet loss rates), we have an asymmetric network. Where (symmetric) MDC systems are designed to deal with symmetric channels, we design an asymmetric multiple description coding (AMDC) method that combines the flexibility of Scalable Coding and Erasure Codes (like MD-FEC), but now asymmetric descriptions are generated. In this chapter we discuss the idea of AMDC and how to find a mapping of erasure codes to descriptions, for a given certain unbalanced network.

—

Chapter 9
Discussion
The thesis is concluded with a discussion where we reflect on the results and the choices made in the presented scenarios. We evaluate and compare the different presented solutions. Furthermore we give recommendations and an outlook to the near future with respect to video compression.

1.4 Contributions of the Thesis

The work presented in this thesis contributed to the field of video compression on several aspects. Modeling the behavior of an encoder is essential when tight control of the encoder in a dynamic environment is required. In Section 3.2, we derive an integral model of a video coder in a bandwidth and power constrained environment. The model can then be used in a quality-of-service system, where the video coder has to cooperate with a network adaptation layer and an application. We performed tests where we employed such a behavior model and quality of service (QoS) system, in a simulated wireless network. This is described in Chapter 4. The same model can also be employed for streaming over wireless WiFi networks. These real-world experiments are described in Chapter 6. Chapter 5 describes the derivation of a behavior model for a JPEG2000 Image Compression Standard (JPEG2000)-based differential texture encoder for a error-prone wireless channel.

Multiple description coding fits a network scenario where multiple paths between sender and receiver can be used. MDC has an extra parameter, namely the redundancy which can be tuned to match the network reliability. In Chapter 7 we focus on controlling an MDC coder, and to optimize its settings in a scenario with multiple receiving clients. Asymmetric MDC encoders can be tuned to match a network with multiple asymmetric channels. We have derived an algorithm that optimizes average quality for a LC-LEC encoder, which is discussed in Chapter 8.

Part I

Network Aware Video Coding

Two

Video Streaming Model

2.1 Introduction

In this thesis, we present different video compression systems. In each case the compression system has to operate in a different scenario. The different scenarios require different designs and algorithmic choices for the video coder. The presented video streaming systems, however, share a general design. In this chapter we generalize the video streaming system. We call this framework the 'Video Streaming Model' (VSM).

Before introducing the VSM, we first consider data flow in the video streaming system (Figure 2.1). The raw video data X is too large to be transmitted without compression.[1] The task of the video coder (VC) is to take care of compressing X in order to fit on the network. The network has a limited capacity and is an error-prone environment where data may get corrupted delayed or lost. For instance, when router queues congest the packets will be delayed or dropped. In the case of wireless networks, electro-magnetic interference will

[1] With raw video we mean the uncompressed sequence of pictures as obtained from camera.

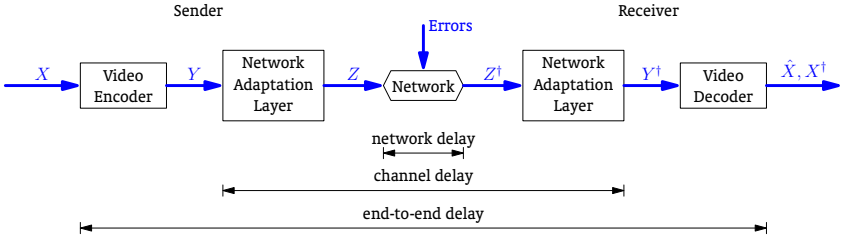


Figure 2.1 — *Video Transmission Channel*

corrupt packets, necessitating retransmissions. Then, the network adaptation layer's task is to protect this data from corruption in the network through forward error correction and retransmissions.[2] Finally, when data has found its way through the network and appears at the receiver, the NAL—handling retransmission and error correction—gives the corrected data to the video decoder. Since the NAL is not always able to repair or recover all data or data may simply arrive too late, we have to assume the the video decoder does not always have all data required for perfect decoding. Under these circumstances the video decoder has to recombine received data and decode it.

Figure 2.1 also shows our definitions of delay. In a real-time encoding scenario, end-to-end delay is defined as the entire delay between starting encoding a frame and displaying the received frame at the decoding side. In a non-real-time encoding scenario, end-to-end delay is defined as the entire delay between requesting video play out and displaying the first frame at the receiving end. Network delay is defined as the (average) time required for transmitting data from sender to receiver. Channel delay is then defined as the end-to-end-delay minus video encoding and decoding delay, or reversely as the network delay plus the channel encoding and channel decoding time. When the end-to-end delay constraint becomes smaller, the time to perform video coding and channel coding also becomes smaller. This will inevitably result in less coding efficiency for the video coder and more uncorrectable channel errors for the NAL. Ever-present errors and congestion on the network make that a continuous transmission at the compressed data rate is sometimes not possible. We introduce buffers at the receiving end to mitigate jitter and to provide time for doing retransmissions, thereby inevitably increasing end-to-end delay. The video codec and NAL therefore have to take the delay constraint into account in when maximizing the picture quality (video coder) and throughput (NAL).

We consider an end-to-end quality metric which depends on two factors. First, the (lossy) encoder compresses the original signal X so that even upon correct reception, decoding yields a distorted version $\widehat{X}$. The second factor is due to network errors. When some data got lost, decoding results in a corrupted version of X, denoted as $X^{\dagger}$. Eventually, the user at the receiving end will notice these corruptions and is able to rate these impairments as being annoying or maybe only barely noticeable. Although far from perfect, a mean-square-error metric or peak-signal-to-noise ratio is often used to express the quality of a video stream.

Shannon's separation principle [65], states that source coding and channel coding *can* be independent, but only under the assumption of infinite length sequences. This means that, in practice, independently operating source and channel coders will not be optimal, since we always want to limit delay by

[2] Channel coding or channel protection will from now on refer to both forward error correction and retransmission

using finite sequences. This is especially true in streaming applications, where short delays are required.

Besides the video coder and the NAL, the VSM also contains the application layer. The function of the application layer is to offer an interface between the user and the the video streaming system.

Many attempts have been made to model the entire video streaming system with a QoS framework [2, 50, 48]. Zhang *et al.* give an overview of QoS approaches for video delivery over wireless Internet [96]. Van der Schaar discusses the need for cross-layer optimization in Ref. [88].

In the next section we treat the VSM, the application, the video coding layer and the NAL as separate layers with their own functionality. In the VSM the layers are cooperating to perform the joint task of video streaming in a given scenario. In different scenarios we can have different types of cooperation between layers. Section 2.3 describes the generalized scenario and the different possible types of layer cooperation. The chapter is concluded with a discussion.

2.2 The Video Streaming Model

2.2.1 Overview

The VSM can be seen as a compact version of the OSI-model[78].[3] Figure 2.2 shows the VSM schematically. In addition to the data flow connection between layers, we define interface parameters on the interface between two layers. These interface parameters reflect the behavior and describe the properties of the data flowing between the layers. During streaming some of these parameters may change, for instance when channel bandwidth changes. Instead of running as stand-alone functional blocks, each layer operates in a dynamic and changing environment, reflected by changing interface parameters.

In addition to the three layers we introduce in our model the network (NW) as a black box between the sender and receiver. The network is outside our control but shows behavior that we should take into account in the rest of our system. This behavior could either be described by on-line measurements or by an adaptive behavior model.

Since we decoupled the system functionalities into separate layers, layer cooperation between them becomes necessary. Each layer has a specific task which can be controlled with one or more internal parameters. When the system is observed as a whole, many different combinations of settings for the internal parameters exist. However, only a subset of these combinations yields a solution that gives a global optimum while fitting the global constraints.

[3] OSI defines 7 layers: Application, Presentation, Session, Transport, Network, Link, Physical. Our model roughly collapses Session and all lower layers in the NAL and fits the video coder in the Presentation Layer.

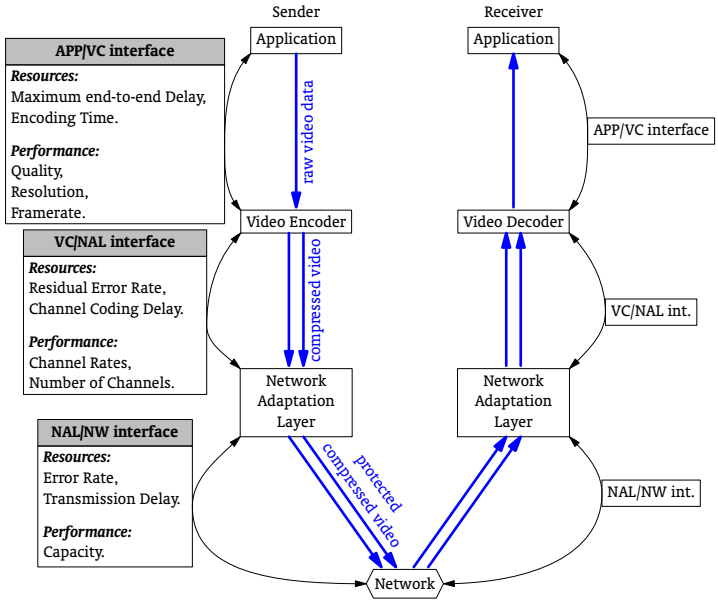


Figure 2.2 — *Video Streaming Model. The left side shows the transmission stack for the sender, the right side for the receiver. The application (APP) defines the high-level constraints of the system, such as quality, resolution etc. The video coder (VC) represents the actual coder. The network adaptation layer (NAL), protects and transmits the encoded content over the network (NW). On each layer interface, a set of parameters is exchanged that defines the properties of the data the data that is exchanged between each layer.*

Only for the video streaming system as a whole it is possible to make a trade off between resources such as rate and delay, and performances such as video quality[4]. The application is therefore able to select one single optimal setting for all lower layers, such that all constraints are taken into account. To find a global (constrained) optimum is a matter of performing constrained optimization, which can be performed off-line when all constraints and behaviors are exactly known at forehand, or real-time in an adaptive fashion using a QoS mechanism [68, 52, 90].

At the interfaces between layers, we distinguish *resource interface parameters* and *performance interface parameters*. Although the parameters that are defined on the interfaces should give sufficient information to be able to make good trade offs between resources and performance, in practice the number of

[4]Video Quality is often expressed in the average peak signal-to-noise-ratio (PSNR) (in dB) of the video frames

interface parameters should not increase too much for complexity reasons.

A *resource* for one layer can be *performance metric* for another layer and *vice versa*. For instance, channel capacity is a resource to be used by the video coder. However, for the NAL, channel capacity is a performance metric since the more capacity (while keeping other parameters the same), the better the performance.

2.2.2 Application

The application layer acts as an interface between the user and the video coder. In our model, the application imposes quantifiable scenario constraints to the video coder and indirectly to the rest of the system. We might think of constraints such as minimal resolution, frame rate, and maximum end-to-end delay. The objective of the application is to achieve the highest possible video quality while adhering to the constraints. The role of the user is to control the application. The user may for instance change the maximum end-to-end delay parameter or allow a lower frame rate.

End-to-end delay plays an important role in streaming applications. It may not always be possible to meet end-to-end delay constraints without hampering the quality. A video application is a *Streaming* application, when due to the mere presence of a delay constraint, concessions to the picture quality have to be made. Without a delay constraint, there is no reason to make any concession towards the quality. In that case the application becomes a download application which can take as much time as required to obtain *all* video data losslessly or at a required quality level.

The statistical characteristics of the video have a great impact on the actual compression ratio or on the amount of effort put into compression. For instance, static video is easier to compress and can be encoded at a lower rate than high-motion video, while maintaining the same quality. In the VSM the scenario defines which video (or which type of video) is streamed, therefore video characteristics are regarded as application constraints to the video coder.

2.2.3 Video Coder

The video coder has many internal control parameters, for instance quantization-step size, format, motion estimator complexity etc. These parameters have to be set correctly to produce an encoded stream that gives maximum quality while and obeying the constraints from the application such as resolution, frame rate and end-to-end delay. The NAL on the other side also imposes constraints to the video coder, such as capacity, average packet-loss rate, etc.

The video coder produces either a single video stream, or multiple streams for layered coding (LC) (Section 3.3) or multiple description coding (MDC) (Section 3.4). These streams are streamed separately to the receiver in different

channels. For each separate channel, the network has individual characteristics. For instance for layered coding (LC), the base layer is streamed over the least error-prone channel, whereas the enhancement layer may be transmitted over a more error-prone channel. The receiver recombines all these streams to produce a single video picture stream.

In Figure 2.2, the video coder is connected to two layers via interfaces. From these two interfaces we can construct a set of resource and performance parameters for the video coder as shown in Table 2.1

Interface	*Performance*	*Resource*
Application/VC	Quality Resolution Frame rate	End-to-end delay
VC/NAL	Channel coding delays Channel residual error rates	Number of channels Channel capacities

Table 2.1 — *Set of Performance and Resource parameters defined on the interfaces of the Video coder with the application and NAL*

It may be surprising that channel delay is a performance parameter as seen from the video coder. There are two explanations. The first is because of symmetry of resources and performance. Since channel delay is a *resource* for the NAL, it is by definition a *performance* metric for the video coder. The second—more intuitive—explanation is that the less delay is imposed by the video coder, the more delay is available for channel coding, given a fixed end-to-end delay constraint.

The residual error rate is also a performance metric, because the video coder takes this error rate into account when predicting the picture quality. In other words, the video decoder is able to handle such a residual error rate while producing the given quality.

If data is corrupted, the decoded picture quality is affected. It depends on the way packets are formed, whether whole frames are affected or only parts of the image. Since a typical video codec uses frame prediction and motion compensation, a damaged picture will cause propagation of errors, called drift. The impairment is visible in all future frames, until an intra coded frame (I-frame) is received. Video decoders often are designed to deal with corrupted or missing data. These decoders are able to reconstruct missing parts of a frame by using surrounding frames and areas in a perceptually acceptable way. This often means re-displaying the previous frame, but also more advanced techniques to interpolate the missing frame exist [3, 17]. Still the corruption will be visible to some extent. Although, by their nature, individual impairments are unpredictable, models exist that predict an average impairment under a certain given loss rate.

In Chapter 3 we will introduce real-time adaptive video coding, scalable coding and multiple description coding as solutions to fill in the video coder. Figure 3.2 shows a block scheme of a typical Motion-Compensated-Transform-based video coder. Most video encoders are based on or are derivatives of this scheme.

2.2.4 Network Adaptation Layer

The network adaptation layer (NAL) provides the video coder controlled access to the underlying network. A virtual channel is created through the underlying network to the destination such that the video coder is shielded from routing issues, retransmissions and packet error correction. Since the model is not limited to one client or to single streams, the NAL in general offers multiple parallel virtual channels to multiple clients. Each of these channels may have different properties such as bandwidth and packet-loss rate or bit-error rate.

The task of the NAL is to transmit data without errors as fast as possible to the receiver(s), given delay and capacity constraints from the surrounding layers. Technically this means that retransmissions and error correction are required to fulfill this task, since the underlying network is generally prone to errors.

An implementation of NAL is the Internet Protocol (IP) stack. These protocols takes care of packetization, rate control and retransmissions (Transport Control Protocol (TCP)). The real-time variants RTP/RTCP are often used for streaming, since a delay constraint can be taken into account. User Datagram Protocol (UDP) is often used for streaming video, when no rate control and retransmissions are required.

The underlying network frequently suffers from congestion and packet losses. The TCP protocol retransmits packets when they are not received at the receiver. The video codec may therefore assume that TCP always delivers the data. A drawback of retransmissions, on the other hand, is that it may take a long time until the packet is finally delivered. Especially when the delay constraint is tight, a retransmission may come too late which results in a missing frame in the picture stream. For this reason, most streaming applications use UDP. Since UDP does not do retransmissions, data packets may get lost, therefore the video decoder should be resilient to lost packets. Another option is to limit the number of retransmissions and to stop retransmitting when the data has expired, i.e. when the moment the frame had to be displayed at the decoder side has passed.

In the VSM, we assume that the NAL may also implement forward error correction (FEC) to protect data from corruption. Especially on wireless transmissions, interference causes packets to arrive in a corrupted form, leaving some bits erroneous. With FEC, channel codes are appended to each packet, such that at the receiving end, the corrupted data can be corrected. In the

wireless 802.11 protocol, FEC is implemented in the Physical layer of the OSI-model. There are no guarantees that all errors will be corrected. When when the number of errors is too high to be able to correct the packet completely, this will result in a corrupted packet which will either be dropped or passed on to the video decoder marked as a packet with residual errors.

These types of protection on one hand increase the reliability of the transmission but on the other hand cost bandwidth and increase delay.

Table 2.2 shows the relevant interface parameters for the NAL.

Interface	*Performance*	*Resource*
VC/NAL	Channel rates Number of channels	Channel residual error rate Channel coding delays
NAL/NW	Transmission delay Network error rate	Network capacity

Table 2.2 — *Interface parameters for the Network Adaptation Layer.*

2.3 Scenario

A 'Scenario' defines the properties and constraints to the VSM and the environment in which it has to operate. A scenario defines the number of clients, the display type of the users device, the kind of network used for streaming and the distribution of the bandwidths of connected clients. Furthermore, the scenario prescribes whether the video is being streamed live or is pre-recorded and whether all clients are streamed simultaneously or individually. Video compression may happen in real time, for live broadcasting, or can happen off-line for video-on-demand services. An example of a scenario is mobile video streaming, where the display size is very limited, the network is dynamic and very lossy at certain times and the battery and processing power are limited. The scenario also defines the criterion to optimize, often just quality or a trade off between quality and rate.

Finally, the scenario defines the way the layers are cooperating in order to achieve a common goal (i.e. optimal quality video streaming). Cooperation comes in different level of complexity.

Fixed Cooperation All interface and internal parameters are established and fixed at design time, and no real-time adaptation occurs.

Bottom-up Every layer informs the layer above about its current operation, such that other layers can adapt and try to accommodate the changed situation.

QoS Negotiation All layers exchange information about their current operation and possible other points of operation. The layers then negotiate until all application and network constraints are met.

The following is a list of the most important properties and constraints defined by a scenario:

- Type of streaming application,
- The targeted number of clients,
- The targeted display device's capabilities,
- Heterogeneity of the networks,
- Heterogeneity of the client down link bandwidth,
- Whether the source is prerecorded or streamed live,
- The optimization criterion,
- Type of cooperation between layers.

The video streaming scenarios we investigated and presented in this thesis, are:

- Low-latency real-time streaming of video to low-power mobile devices, in Chapters 4 and 5.
- Low-latency real-time streaming of video over WiFi networks, in Chapter 6.
- Streaming of video to many clients using P2P networks, in Chapters, 7 and 8.

The first two are examples of a real-time streaming scenario and the third is a broadcasting scenario.

2.4 Discussion

The VSM is aimed to be a generalization of the video streaming systems presented in Part II. Other scenarios that are not discussed here, but may fit in the same model or require small modifications to the model are

- Live surveillance, (live unicast transmission),
- Prerecorded unicast streaming (video-on-demand),
- Television broadcasting,
- Streaming with transcoding.

By defining the model as a stack of layers with interfaces and sets of interface parameters, the model is easily mapped to a QoS approach where each

layer finds a jointly optimal setting, by exchanging information and negotiation. In Chapter 4 we use a QoS system to adaptively find optimal video coder and channel coder settings in a real-time video coding and streaming experiment. The Adaptive Resource Contracts (ARC) QoS system is used to let all layers cooperate such that the system promptly adapts to changes in network characteristics. ARC was generally designed for complex modular systems that should operate in dynamic and resource-scarce environments. ARC is explained in more detail in Chapter 4 and 5 and in Refs. [73, 87, 86, 90].

In the VSM, video compression is decoupled from channel coding and transmission. The benefits are that the video coder is shielded from decisions and implementation issues in the other layers. Furthermore, the actual encoding algorithm and implementation may be replaced by another. Although from a design point of view this is very attractive, in reality, the video coder and NAL cannot be easily separated since they are inter-dependent: the limited capacity of a network directly limits both the amount of channel protection and the rate produced by the video coder. This necessitates joint optimization. Especially when the network characteristics change over time, each subsystem should adaptively change its parameters according to the current network conditions. In these dynamic scenarios, our solution is that parameters can be communicated between the video coder and NAL, such that a joint optimum can be found when these layers optimize their settings.

QoS cooperation is nevertheless neither required nor the only use of this model. The model tells us which parameters and variables are relevant in the design of our video streaming systems and how they relate to each other. Another approach is that layers simply exchange status information with other layers in a bottom-up approach, without QoS negotiations. For instance with layered coding (Section 3.3) the video-rate control can simply consist of selecting the number of layers that can be transmitted over the channels without losses. Streaming systems can also be designed while assuming a certain network behavior but without any cross-layer signaling. For instance an MDC encoder that is designed for a dynamic peer-to-peer network but where no up-to-date network information is available. The redundancy introduced by MDC has to be adjusted to match the error-resilience or scalability required by the network.

Three

Network Aware Video Coding Strategies

3.1 Introduction

In streaming scenarios, the video coder is working in cooperation with an application and a network communication layer. The video coder should therefore be aware of the underlying network while at the same time obeying the conditions set by the application. In order to cope with fluctuations in bandwidth, packet losses and congestion, the video coder should either be informed of changes in the network or generate a video stream that can cope with changes in the network characteristics.

In Part II of this thesis different scenarios are presented. In each scenario a different way of cooperation and a different type of video coding is used. Each scenario dictates a different video coding strategy to deal with network dynamics, network-losses and heterogeneity of the clients. In this chapter we discuss three different video coding methods, which will be used in subsequent chapters.

The first coding approach is real-time adaptive video coding (RAVC). Using a QoS interface, up-to-date network characteristics are used to continuously update encoder settings. Knowledge about network and application is propagated and negotiated between layers to obtain a global constrained optimum as defined by the application. To perform the QoS optimizations, the behavior of the encoder has to be modeled which is discussed in Section 3.2. The work presented in Chapters 4 and 6 use RAVC and QoS cooperation between layers.

Section 3.3 discusses an approach where cooperation is based on Layered Coding instead of RAVC. The actual encoding may even be done off-line. Based on the network characteristics, each client only receives the layers that can be transmitted without congesting the network. Although this still requires

up-to-date knowledge of the network behavior, QoS *negotiations* are no longer necessary. In Chapter 5 we follow a different approach, we construct a behavior model for a progressive JPEG2000 coder that can be used in a QoS type of cooperation between layers.

Section 3.4 describes the MDC approach in which multiple independently decodable descriptions are generated by the encoder. MDC descriptions are inherently scalable to network capacity and resilient to errors, which make MDC suitable for lossy packet networks such as P2P networks, where no hard guarantees can be given about delivery of packets. In this case the video coder can be designed for a particular network or network protocol (such as peer-to-peer, UDP). However, no real time network information is required to adapt to network changes. Chapters 7 and 8 discuss the use of MDC in P2P networks where tight QoS cooperation is not possible or available.

3.2 Real-time adaptive video coding

3.2.1 Introduction

Real-time adaptive video coding (RAVC) is targeted at streaming applications where the video is encoded in real time. Encoding in real time gives the advantage that if network behavior is dynamic, we can instantaneously react to changes in the network. A change in bandwidth, for instance, could immediately result in changing the quantization-step size for the next frame. A quintessential requirement is the availability of up-to-date information about the current network state. We rely on a QoS mechanism that exchanges this information between all layers of the system. The interface-parameter sets in the VSM contain the resource and performance parameters. In the UBICOM project we devised the Adaptive Resource Contracts (ARC) method for implementing QoS [73, 87, 86, 90]. ARC offers the possibility to request, negotiate and to contract these parameters for a certain amount of time.

To provide performance and resource information to other layers, the Video Coder should monitor its own resource usage and performance. A problem is that the performance is only known after encoding the frames under consideration by the encoder. For ARC we rather need to predict the performance and resource usage, based on given encoder settings and context information, but before the actual encoding has been performed. To be able to make these predictions we need a behavior model. For instance based on the target bit rate setting, the model predicts the resulting quality (with some limited precision). The behavior model presented in this section, predicts the performance based on the interface parameters from the VSM and the internal video coder parameters. The behavior model is based on the ITU-T H.263 Video Coding Standard (H.263) [62] video encoder, which also stood model for the typical video encoder presented in Section 2.2.3.

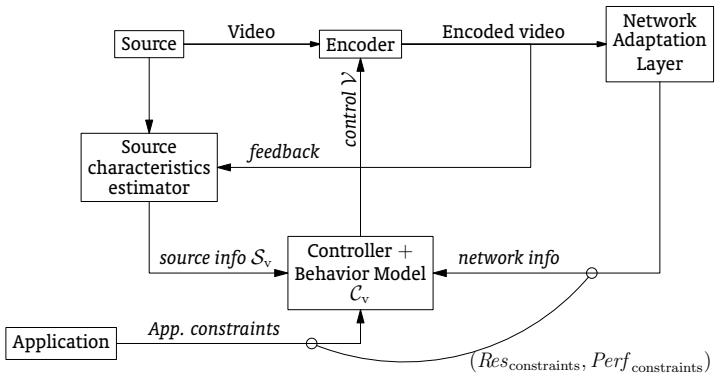


Figure 3.1 — *Control Loop of the real-time adaptive video coding (RAVC)*

Figure 3.1 shows the block scheme of the real time adaptive video coder within the context of application and NAL. The encoder produces the encoded bit stream and hands this over to the NAL as discussed in the previous chapter. The encoder controller uses up-to-date information of the NAL and of the source to compute the new encoder settings. The source characteristics estimator analyzes the video source and the produced video stream and updates the source characteristics. The behavior model that is discussed here extensively, resides inside the encoder controller. The inner workings of the source characteristics estimator is not discussed here explicitly, but can be inferred from how we construct the behavior models later on.

3.2.2 Video Coder Behavior Model

The video encoder behavior is described by a number of parameters. We group all parameters according to Table 3.1. The grouping of parameters is as follows:

$\mathcal{V}$	Internal parameters of the Video Coder.
$Res(\mathcal{V})$	Set of parameters describing resource usage of the video coder.
$Perf(\mathcal{V})$	Set of parameters describing the performance of the video coder.
$\mathcal{S}_v$	Set of parameters describing the statistical characteristics of the video source.
$\mathcal{C}_v$	Set of parameters containing platform-dependent parameters and timing constants independent of video source or network.

The resource and performance parameter sets reflect the interface parameters of the VSM listed in Table 2.1. We added central processing unit (CPU) usage to this list, since CPU-time was taken into account as a resource in the UBICOM scenario. We also incorporate the channel bit-error rate (as opposed to a packet-loss rate), since the communication in UBICOM offered wireless transmission

in which residual bit errors could occur when the channel protection could not correct all errors.

In addition to the sets of interface parameters, we have set of internal video coder parameters $\mathcal{V}$, discussed in the next section, and a set of source characteristics $\mathcal{S}_{\mathrm{v}}$. These parameters describe statistical characteristics of the video source, for instance the amount of variance and the amount of motion. These characteristics of course may change during the sequence. We assume that these parameters are estimated while encoding in a feedback loop and are hence effectively *a priori* known to the encoder. Finally, we have a set of platform and encoder dependent constants $\mathcal{C}_{\mathrm{v}}$. The values of these parameters are constant for a given platform and encoder implementation, but are not dependent on the video source or network conditions.

The behavior model presented in the remainder of this section predicts performance and resource usage as function of encoder settings $\mathcal{V}$. We can therefore summarize the model by functions $Perf(\mathcal{V})$ and $Res(\mathcal{V})$. In Chapter 4 and 6, the behavior model is used to find a global constrained optimum for the settings of the video coder, application and NAL, using the ARC system. There the constrained optimization criterion is defined as:

$$\max_{\mathcal{V}} \; Q(\mathcal{V}) \tag{3.1}$$

such that

$$Res(\mathcal{V}) \; \leq Res_{\mathrm{constraints}} \tag{3.2}$$

and

$$Perf(\mathcal{V}) \; \geq Perf_{\mathrm{constraints}} \tag{3.3}$$

where $Res_{\mathrm{constraints}}$ and $Perf_{\mathrm{constraints}}$ are the QoS constraints on the interfaces with the application and NAL. The criterion maximizes the end-to-end quality Q over all possible combinations of the encoder settings $\mathcal{V}$ while adhering to the application and NAL constraints. The used end-to-end quality metric is based on the PSNR of individual video frames and is further introduced in Section 3.2.9. Note that there is not necessarily always a solution to this optimization problem, in that case the user of the system should relax his requirements or cancel the transmission.

The video coder behavior model describes and models the relationships between internal encoder settings and the interface parameters of Table 3.1. The goal of this model is to predict the performance (e.g. Quality) and resource usage (e.g. delay, transmission rate) of a given video coder, given internal coder settings. This model is used in Chapter 4 in a QoS system to find a global constrained optimum.

The video coder on which our model is based is the H.263 coder. Figure 3.2 shows the block scheme of the encoder. Five main encoding steps are distinguished:

Table 3.1 — *Overview of Video Coding Behavior Model parameters. Each of these parameters may vary over time. However, for notational clarity we have dropped the frame index i as argument of these parameters.*

Parameter set	*Symbol*	*Parameter*
$\mathcal{V} =$	R	Encoding rate
	N_{fs}	Number of frame skips
	β	Intra-refresh rate
	l_{me}	Motion vector search range
$Res(\mathcal{V}) =$	T_v	End-to-end delay
	C_v	Total CPU usage
	R_c	NAL transmission rate (Bandwidth)
$Perf(\mathcal{V}) =$	Q	End-to-end video quality metric
	T_c	NAL transmission delay
	p_c	NAL bit error rate
	C_c	NAL CPU usage
$\mathcal{S}_{\mathrm{v}} =$	σ_X^2	Mean variance of video-frame macroblocks
	G_0	Prediction gain between two consecutive frames
	G_{me}	Motion compensated prediction gain between two consecutive frames
	N_{mc}	Motion coherence
	N_{mcc}	Motion compensated coherence
	σ_l	Standard deviation of the motion vector length
	f_{fps}	Frame rate
$\mathcal{C}_{\mathrm{v}} =$	t_{1mv}	vector evaluation time
	T_{fixed}	minimal encoding time
	γ	filter strength
	θ_I, $R_{\mathrm{I}\emptyset}$, $D_{\mathrm{I}\emptyset}$	RD-curve parameters I-macroblocks
	θ_P, $R_{\mathrm{P}\emptyset}$, $D_{\mathrm{P}\emptyset}$	RD-curve parameters P-macroblocks
	$\sigma^2_{\mathrm{artifact}}$	artifact distortion
	L	residual bit-error L-parameter

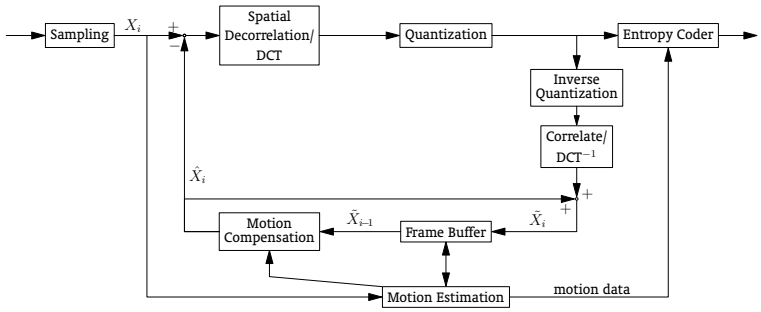


Figure 3.2 — *Typical Motion-Compensated-Transform video encoder*

1. Frame prediction and frame skipping,
2. Motion estimation and compensation,
3. Intra macroblock refreshing for introducing error-resilience,
4. Rate control and quantization,
5. variable length coding (VLC) / Entropy Coding.

Each of these steps has its own parameters and imposes its own behavior on the video coder as a whole. The choice of which internal parameters and source characteristics to incorporate into this model is on one hand a result of the wish to limit the model complexity. On the other hand, since it should be possible to make trade-offs between performance and resources, a sufficiently complex model is required.

In the following sections we will, step by step, introduce the behavior model and thereby relate the parameters from Table 3.1 to each other. We follow a bottom-up approach where we first find a model to predict the amount of variance that has to be encoded, given motion search range and frame skip parameters. Then we find a model for predicting the quantization distortion given the variance and the encoder rate setting. The next step is to find a model to predict the decoded video quality, taking quantization distortion, frame skips, and residual channel error distortion into account, assuming a network with bit-errors. Finally, we establish models to predict resource usage (delay and cpu time) given the encoder settings and network information.

3.2.3 Prediction Gain

The H.263 frame prediction scheme of uses *intra*-coded I-frames, *inter*-coded P-frames and *bidirectionally predicted* B-frames. In the UBICOM project we have chosen not to use B-frames, for their added delay and for simplicity reasons. We therefore only take I-frames and P-frames into account in this behavior model.

Inter prediction exploits predictability between frames and thereby decreases the amount of variance of the signal that is encoded. In general inter prediction increases coding efficiency because less rate is needed to attain the same quality. In order to predict the rate necessary to encode a particular frame, we first predict the variance of these frames. Because these difference pictures are not available before the actual encoding has taken place, we need to model the behavior of the frame predictor, based on data that *is* available at forehand.

The ratio between the variance of a frame X_i and the variance of the difference between the frame and the prediction X_i is called the prediction gain:

$$G_i \;=\; \frac{\mathrm{var}\left[X_i\right]}{\mathrm{var}\left[X_i - \widetilde{X}_{i-1}\right]}. \tag{3.4}$$

Here X_i is the current frame, which is predicted by the previous reconstructed frame (reference frame) $\widetilde{X}_{i-1}$. Prediction gain is a measure for how good a frame prediction is. If we can estimate G_i at forehand and know the variance of a frame $\mathrm{var}\left[X_i\right] \;=\; \sigma_X^2$, we can also estimate the variance of the frame difference.

In our model prediction gain is influenced by two parameters, namely frame skip N_{fs} and motion vector search range l_{me}. We will first discuss the effect of these parameters separately and then their aggregated effect.

3.2.4 Frame prediction and Frame skipping

In a real-time streaming scenario, frames may have to be skipped for several reasons. First of all it may happen that encoding every frame consumes too much time to achieve real time performance. Secondly, it may turn out to yield better compression ratio to skip frames and to concentrate on finer quantization, for instance in low-motion scenes.

The frame skip parameter $N_{\mathrm{fs}} \;\geq\; 1$ is defined as the integer frame distance between two encoded frames. In this case we define the prediction gain between frame i and frame $i - N_{\mathrm{fs}}$, depending on the frame skip (N_{fs}) and motion vector search range (l_{me}) parameters: follows:

$$G_i(N_{\mathrm{fs}}, l_{\mathrm{me}}) \;=\; \frac{\mathrm{var}\left[X_i\right]}{\mathrm{var}\left[X_i - \widetilde{X}_{i-N_{\mathrm{fs}}}\right]} \tag{3.5}$$

Since in general the difference between frames increases when their distance increases, we expect prediction gain to decrease when N_{fs} increases. Figure 3.3 illustrates for frames in the `foreman` [1] sequence how prediction gain $G(N_{\mathrm{fs}}, 0)$ decays to one when the number of skipped frames is increased when

[1] `foreman` and `carphone` are well-known video sequences, often used for test and evaluation purposes

no motion estimation takes place.[2] The circles show the prediction gain when no motion compensation is used.

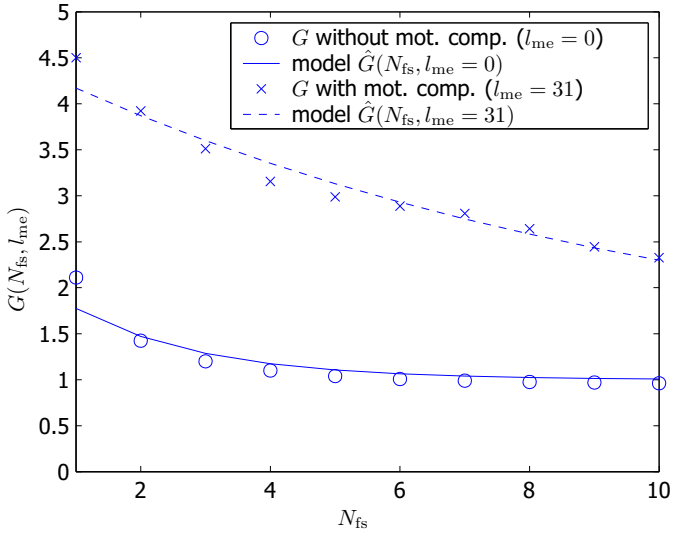


Figure 3.3 — *This figure shows the relationship between prediction gain and increased frame distance for frames in the* `foreman` *sequence.*

The reason that $G(N_{\mathrm{fs}}, 0)$ decays to 1 is that the worst prediction that can happen is when there is *no* predictability between the two frames, meaning that the entire frame has to be intra coded.

Our aim is to estimate the prediction gain given the frame skip parameter. Based on the behavior displayed in Figure 3.3, we assume an exponential decaying model:

$$\hat{G}(N_{\mathrm{fs}}, l_{\mathrm{me}} = 0) \;=\; 1 + (G_0 - 1)\, \exp\left[\frac{1 - N_{\mathrm{fs}}}{N_{\mathrm{mc}}}\right], N_{\mathrm{fs}} \geq 1. \qquad (3.6)$$

Here N_{mc} is the Motion Coherence length, being the average number of skipped frames at which the prediction gain minus one has decayed by a factor e. G_0 is the expected prediction gain when no frames are skipped ($N_{\mathrm{fs}} = 1$). Since the parameters N_{mc} and G_0 depend on the video source characteristics, they are considered 'Source Parameters' and are contained in the $\mathcal{S}_{\mathrm{v}}$ vector. The solid line in Figure 3.3 is plotted with model Eq. (3.6). The standard deviation of the shown points with the model Eq. (3.6) is 0.12, which is sufficiently small, since a difference in G of 0.12 corresponds to a difference of 0.1dB in quality.

[2] For notational clarity, we drop the frame index i from here onwards for all parameters.

3.2.5 Motion Compensation

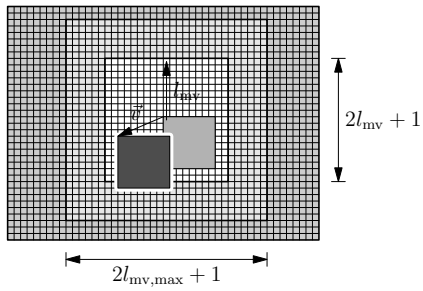


Figure 3.4 — *Explanation of motion vector search range. Each small square represents one pixel. The range in which the search is performed is l_{me} which should be smaller than or equal to the maximum range $l_{\mathrm{me,max}}$. The $\vec{v}$ vector is a vector within the range.*

Motion compensation (MC) is used to further improve the prediction $\widetilde{X}_i$ of the encoded frame, resulting in P-frames with lower variance. Camera motion and object motion are compensated, resulting in a prediction that is closer to the frame being predicted. We assume the motion estimator complexity to be parametrized. Since motion estimation is a time and power-consuming task, we do not always want to perform full motion estimation (ME), but only search among a limited set of candidate vectors. Prediction gain will decrease when not all possible motion vectors are evaluated, resulting in a suboptimal motion compensation vector. We experimentally found a trade off between complexity and prediction gain. Correct modeling of this behavior also depends on the implementation of ME, for instance full search, diamond search, N-step search or hierarchical ME [19, 18]. Although advanced ME techniques give good results at lower complexities, we have chosen the full search method since it is easy to model. More advanced ME techniques can still be incorporated, but will require a different model since they have a different complexity – prediction gain trade off.

We parametrized full-search ME by making the search range adaptive. Search range $l_{\mathrm{me}} \in \mathbb{N}$ is defined as a square around the current (macro) block position with distance l_{me} to the center, in which all possible motion vectors are evaluated. The maximum motion vector length in H.263 is 15 pixels. Since we use half-pixel (half-pel) motion estimation, which means that a motion vector could point half-way between two pixels, the pixel offsets are multiplied by two to obtain integer values. The maximum motion vector length is 15 pixels resulting in $l_{\mathrm{me}} \leq 31$. The total number of vectors that are evaluated during full search ME is $n_{mv} = (2l_{\mathrm{me}} + 1)^2$ (see Figure 3.4).

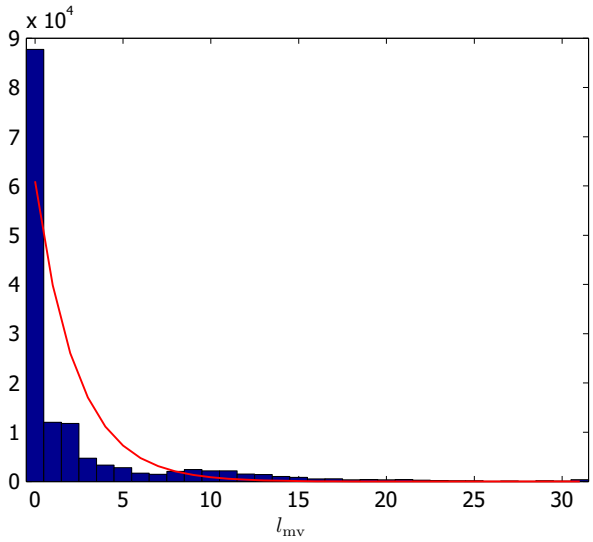


Figure 3.5 — *Histogram of motion vector lengths for the* `foreman` *sequence. The Solid line represents the model in Eq. (3.8) fitted to this data.*

Figure 3.5 shows a histogram of the found motion vector lengths $l_{\mathrm{mv}} \in \mathbb{N}$ for the `foreman` sequence. The histogram suggests a exponential probability density function (pdf) for the physical (continuous) motion vector length:

$$f_{l_{\mathrm{mv}}}(x) \;=\; \frac{1}{\mathrm{std}\left[l_{\mathrm{mv}}\right]} \exp \frac{-x}{\mathrm{std}\left[l_{\mathrm{mv}}\right]}. \tag{3.7}$$

When $N_{\mathrm{fs}} = 1$, $\mathrm{std}\left[l_{\mathrm{mv}}\right]$ can be replaced by σ_l, the standard deviation of motion vector length:

$$f_{l_{\mathrm{mv}}}(x) \;=\; \frac{1}{\sigma_l} \exp \frac{-x}{\sigma_l}. \tag{3.8}$$

σ_l is a measure for the amount of motion between two consecutive frames and is considered another Source Parameter (in $\mathcal{S}_{\mathrm{v}}$).

Figure 3.6 illustrates that prediction gain increases when the motion search range l_{me} is increased. Based on this behavior we assume that a motion vector 'match' (when the right motion vector is found) increases the prediction gain with a constant amount while finding the wrong motion vector does not improve prediction gain. The probability of a match increases when the search range is expanded and is given by

$$P_{\mathrm{match}}(N_{\mathrm{fs}} = 1, l_{\mathrm{me}}) \;=\; \mathrm{Prob}[l_{\mathrm{mv}} \leq l_{\mathrm{me}}] \;=\; 1 - \exp \frac{-l_{\mathrm{me}}}{\sigma_l} \tag{3.9}$$

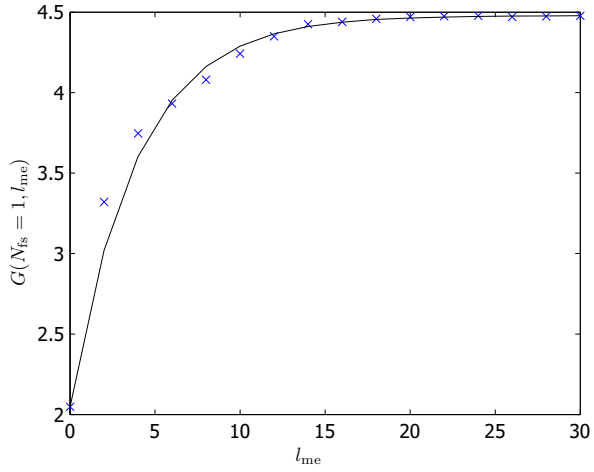


Figure 3.6 — *Prediction Gain versus motion vector search range for the* `foreman` *sequence. The crosses mark the experimental values for the* `foreman` *sequence and line shows the fitted model using Eq.(3.10)*

such that $P_{\mathrm{match}} = 0$ when there is no motion compensation ($l_{\mathrm{me}} = 0$) $P_{\mathrm{match}} = 1$ with unconstrained motion compensation ($l_{\mathrm{me}} \to \infty$).

We are now able to estimate the prediction gain with the following model.

$$\begin{aligned}\hat{G}(N_{\mathrm{fs}}=1,l_{\mathrm{me}}) &= (1-P_{\mathrm{match}}(N_{\mathrm{fs}}=1,l_{\mathrm{me}}))\,(G_0-1)+ \\ &\quad P_{\mathrm{match}}(N_{\mathrm{fs}}=1,l_{\mathrm{me}})\,(G_{\mathrm{me}}-1)+1\end{aligned} \tag{3.10}$$

where Source Parameter G_{me} denotes the prediction gain when full motion estimation is performed. The continuous line in Figure 3.6 shows the behavior of the model Eq. (3.10) for fitted values of G_0 and G_{me}. Although the fit shown in Figure 3.5 is not very precise, we justify this model by pointing to the sufficient accuracy of the resulting model Eq. 3.10 (Figure 3.6).

3.2.6 Joint effect of frame skip and motion compensation on prediction gain

The joint effect of frame skip and motion compensation can be observed in Figure 3.3. At maximal motion compensation ($l_{\mathrm{me}} = 31$), the prediction gain is higher, but also decays at a slower rate when the frame skip increases. This means that when maximal motion compensation is used, the predictability between frames stays high over longer frame distances. This slower decay at

maximal motion compensation is parametrized with a different motion coherence parameter N_{mcc} (motion compensated coherence).

When frames are skipped, motion vectors are computed between frames that are farther apart in time. Since there is often more motion in this longer period, the motion vectors will on average become larger. Figure 3.7 shows how the standard deviation of the motion vector length increases for `foreman` when frame skip increases (crosses). If we assume the motion in subsequent frames to be independent, we can sum the variances of the motion vector lengths of the individual frames. This results in the standard deviation of the motion vector length over N_{fs} frames:

$$\mathrm{std}\left[l_{\mathrm{mv}}\right] \;=\; \sqrt{N_{\mathrm{fs}}}\,\sigma_l \tag{3.11}$$

and in the probability of a match:

$$P_{\mathrm{match}}(N_{\mathrm{fs}}, l_{\mathrm{me}}) \;=\; \mathrm{Prob}[l_{\mathrm{mv}} \leq l_{\mathrm{me}}] = 1 - \exp\frac{-l_{\mathrm{me}}}{\sqrt{N_{\mathrm{fs}}}\,\sigma_l} \tag{3.12}$$

by replacing σ_l in (3.9) by $\sqrt{N_{\mathrm{fs}}}\sigma_l$. The solid line in Figure 3.7 is plotted using model (3.11) and suggests that our i.i.d. assumption of the motion vectors is reasonable.

We can now construct a model for estimating the prediction gain given the frame skip and the motion vector search range, by combining Eqs. (3.6) and (3.10):

$$\begin{aligned}\hat{G}(N_{\mathrm{fs}}, l_{\mathrm{me}}) \;= (1 - P_{\mathrm{match}}(N_{\mathrm{fs}}, l_{\mathrm{me}}))\left(\exp\frac{1-N_{\mathrm{fs}}}{N_{\mathrm{mc}}}\right)(G_0 - 1) + \\ P_{\mathrm{match}}(N_{\mathrm{fs}}, l_{\mathrm{me}})\left(\exp\frac{1-N_{\mathrm{fs}}}{N_{\mathrm{mcc}}}\right)(G_{\mathrm{me}} - 1) + 1\end{aligned} \tag{3.13}$$

For frames in the `foreman` sequence, we have generated a 3-D plot of the experimentally measured prediction gain for different frame skips and motion vector search ranges in Figure 3.8(a). In Figure 3.8(b) the model in Eq. (3.13) is shown for the same conditions. A reasonably good match is obtained since the standard deviation of the error between the model and the measured values of the prediction gain was in this case 0.21. When assuming typical prediction gains of 3 and higher, this corresponds to differences in the video quality of maximally 0.3 dB, which is acceptably small.

3.2.7 Intra Block Refresh

Intra block refresh is a technique used to reduce the effect of channel errors. Residual channel errors may impair a frame and these impairments are propagated through consecutive inter-coded frames. One way to stop error propagation is to send intra coded frames once in a while. Another option is to send

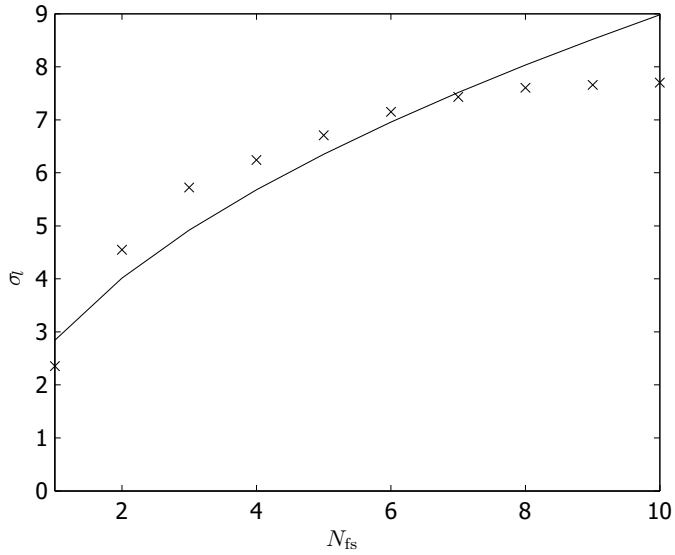


Figure 3.7 — *Standard deviation of the motion vector length for increasing frame skips of the* `foreman` *sequence*

intra-coded macro blocks in inter coded frames. A fraction β of all macro blocks is then randomly selected for intra coding.

The average variance of individual intra macro blocks is σ_X^2. For inter-coded macro blocks we can estimate the variance by $\sigma_X^2 \hat{G}^{-1}(N_{\mathrm{fs}}, l_{\mathrm{me}})$. Since in general $G > 1$, a high β results in less coding efficiency. Clearly there is a trade off between increasing error-resilience (by increasing β) and increasing coding efficiency (by decreasing β). The effect of β on (decoding) quality will become clear after taking into account quantization and residual channel errors in the following paragraphs.

3.2.8 Quantization and Variable Length Coding

In order to predict the rate of the produced bit stream after quantization and variable length coding (VLC), we use rate-distortion (RD) theory [36]. For Gaussian memoryless sources, RD theory gives us a lower bound of the needed bit rate when a certain distortion is allowed:

$$R(D_q) \geq \frac{1}{2} \log_2 \frac{\sigma_X^2}{D_q}, \text{ for } 0 \leq D_q \leq \sigma_X^2, \tag{3.14}$$

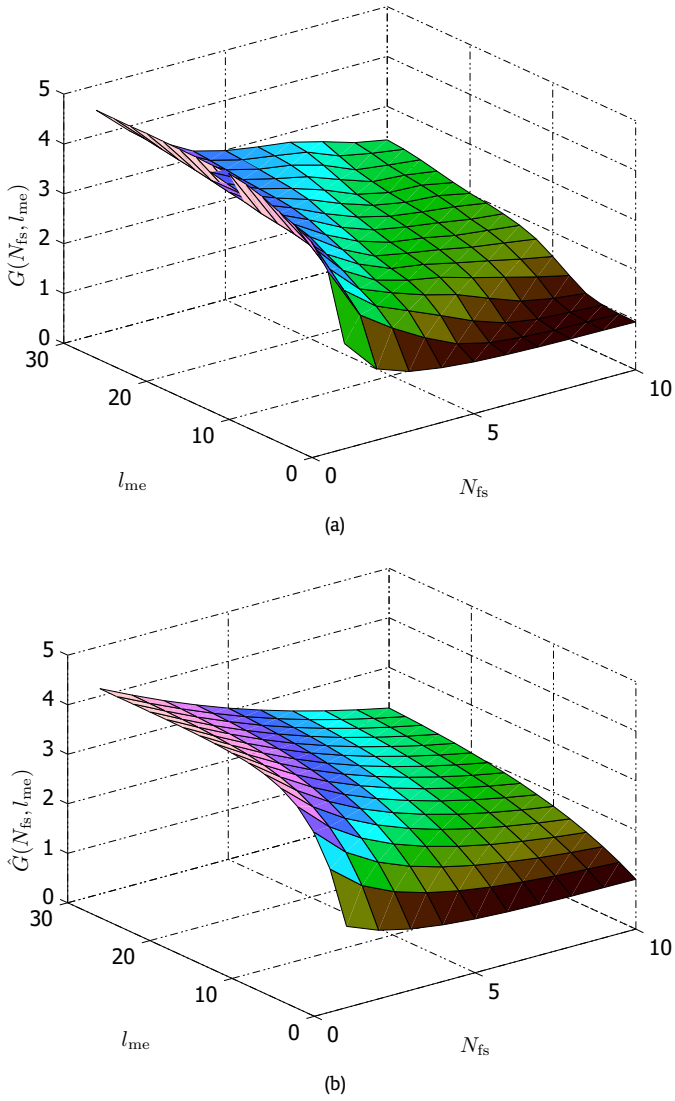


Figure 3.8 — *(a) Experimental values of Prediction Gain for different frame skips N_{fs} and search ranges l_{me}. (b) The estimated prediction gain using the model in Eq. (3.13).*

where R is the bit rate (bits/sample) and D_q is the mean square error (MSE) quantization distortion $D_q = \mathrm{var}\left[X - \widehat{X}\right]$.

For non-Gaussian sources and sources with memory, this model is not very accurate. For a practical video source most video coders are able to perform better than this. We based our model on the model introduced by Stuhlmüller et al. [70]:

$$\hat{D}_q \;=\; \frac{\theta}{R - R_\emptyset} + D_\emptyset. \tag{3.15}$$

Here $R_\emptyset$ is the rate-offset, $D_\emptyset$ is the distortion-offset and θ is the RD factor.

To incorporate the use of intra coded macro blocks and intra frames ($\beta = 1$), we took a different approach than Stuhlmüller et al. We use different RD-curves for intra and inter coded macro blocks. The reason is that we can now incorporate our prediction gain estimation for inter macro blocks. Furthermore, we have normalized the RD-model by extracting the variance of the input signal. The distortion is calculated as the average distortion over all macro blocks as follows:

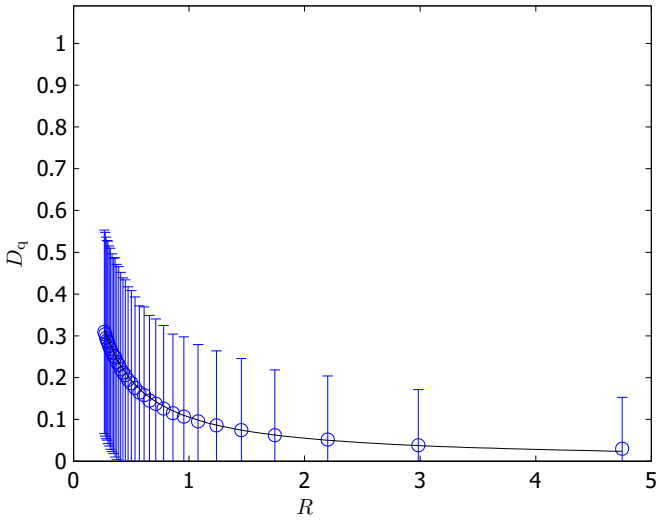
$$\begin{aligned}\hat{D}_q(\mathcal{V}) = \hat{D}_q(R, N_{\mathrm{fs}}, l_{\mathrm{me}}, \beta) =& (1-\beta)\frac{\sigma_X^2}{\hat{G}(N_{\mathrm{fs}}, l_{\mathrm{me}})}\left(\frac{\theta_P}{R - R_{\mathrm{P}\emptyset}} + D_{\mathrm{P}\emptyset}\right) + \\ & \beta\,\sigma_X^2\left(\frac{\theta_I}{R - R_{\mathrm{I}\emptyset}} + D_{\mathrm{I}\emptyset}\right).\end{aligned} \tag{3.16}$$

Because we use different models for intra and inter macro blocks, we now have the following source parameters θ_I, θ_P, $R_{\mathrm{I}\emptyset}$, $R_{\mathrm{P}\emptyset}$, $D_{\mathrm{I}\emptyset}$ and $D_{\mathrm{P}\emptyset}$ which are considered to be intrinsic to the implemented video coder and are assumed not to change. Figure 3.9 shows for intra and inter coded macro blocks the RD-curves for the `foreman` sequence. The circles are the average values of the experimentally obtained macro block distortions. The continuous line represents the fitted model, and gives an accurate fit. The error bars indicate the standard deviation of the distortion of individual macro blocks. Although the RD behavior of individual macro blocks has a large spread, by using the law of large numbers the average distortion over a large number of macro blocks can be well estimated by (3.16). In the context where this model is used we are more interested in the average RD-behavior over a large number of macro blocks and over multiple frames, than the behavior for a particular macro block.

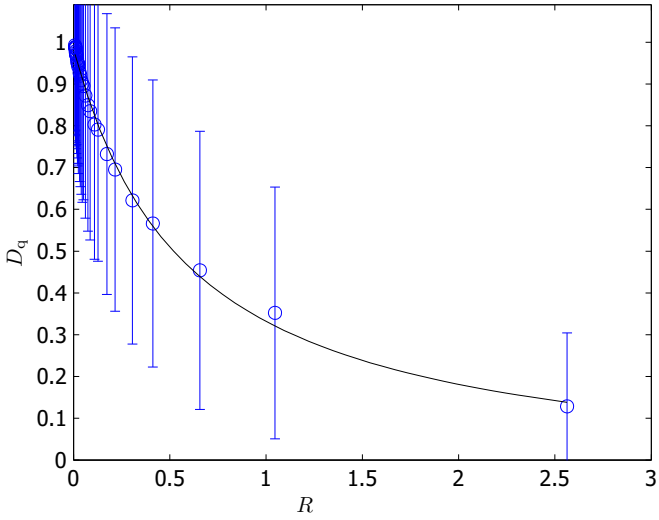
3.2.9 End-to-end quality metric

In addition to the quantization noise, the perceived picture quality at the receiver side is also influenced by:

1. Residual error impairments of transmitted (non-skipped) frames



(a) intra



(b) inter

Figure 3.9 — *Rate Distortion curves for (a) intra and (b) inter coded blocks*

2. Frames skipped by the encoder,

Residual channel errors (errors not corrected by error correction or retransmission) impose their own local impairments on the received pictures. Frame skips are effectively perceived as the previous frame being displayed twice as long. Especially in scenes with a lot of motion, frame skips degrade the perceived quality. We assume residual errors to be independent of the errors introduced by quantization and by skipping frames. We discuss non-skipped and skipped frames first separately and then combine the two effects in a single metric for end-to-end quality Q.

3.2.9.1 Effect of Residual Channel Errors and quantization on encoded and transmitted frames

Each transmitted frame suffers from quantization distortion and—when residual channels errors occur—residual error impairments. We define the total distortion on an encoded and transmitted (non-skipped) frame i as the summation of quantization distortion $D_q(i)$ (Eq. (3.16)) and residual channel error distortion $D_c(i)$ (we refer to Figure 2.1 for a clarification on the used symbols):

$$\begin{aligned} \hat{D}_{\mathrm{non-skipped}}(\mathcal{V}, i) &= \mathrm{var}\left[X_i - X_i^{\dagger}\right] \\ &= \underbrace{\mathrm{var}\left[X_i - \widehat{X}_i\right]}_{D_q(\mathcal{V},i)} + \underbrace{\mathrm{var}\left[\widehat{X}_i - X_i^{\dagger}\right]}_{D_c(i).} \end{aligned} \tag{3.17}$$

Predicting the variance of the residual-error impairment signal is very difficult since these impairments are unpredictable in size, amount and nature. Only based on a large number of experiments, we are able to compute an average residual error variance, if we know encoder settings and network loss rates. Furthermore, as discussed in Section 2.2.3 the type of errors depends on whether residual bit-errors occur or only dropped packets.

Motion compensation worsens the effects of residual errors. Since a corrupted part of the frame may be used to predict a block at a different position in the next frame, image impairments will be smeared out. Since most video coders use half-pixel ME, which causes a filtering effect, an error will eventually fade out as in an infinite impulse response system. The following model, adopted from [71], predicts the variance of the residual error in each frame i introduced by an initial distortion in frame j:[3]

$$\sigma_c^2(i,j) = \begin{cases} \sigma_u^2(j)\,\frac{1-\beta\,(i-j)}{1+\gamma\,(i-j)}, & 0 \le (i-j) \le \lfloor \beta^{-1} \rfloor \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

[3]The frame indexes are introduced again from here onwards, since multiple frames are involved in calculating the end-to-end quality.

where γ is the leakage, it describes the half-pixel ME filtering strength, and is assumed to be a constant factor. The intra-block-refresh rate β was already introduced in Section 3.2.7. Variance $\sigma_u^2(j)$ is the amount of errors injected in frame j. We assume that after $\lfloor \beta^{-1} \rfloor$ all blocks are refreshed and no residual errors from the initial impairment are present.

$\sigma_u^2(j)$ depends on how many channel errors occur during transmission and how (on average) a channel error affects a frame. In Figure 3.10(a) the relation between bit-error rate p_c and average $\sigma_u^2(j)$ is shown. The crosses show the average distortion of all experiments. In each experiment (10000 in total), a different error pattern was induced on the data to simulate independent channel bit errors. The error bars indicate the standard deviation over the separate experiments.

To model the behavior shown in Figure 3.10(a), we assume a linear relationship between $\sigma_u^2(j)$ and the amount of distortion of an independent artifact $\sigma^2_{\text{artifact}}$:

$$\sigma_u^2(j) = n \cdot \sigma^2_{\text{artifact}} \tag{3.19}$$

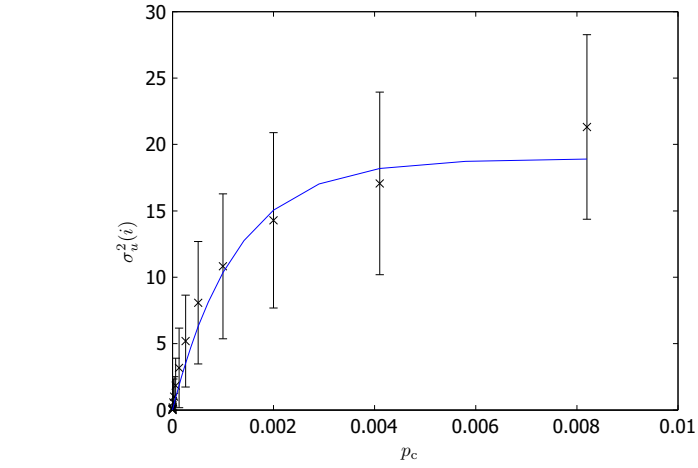
$$= \left[1 - (1 - p_c(j))^L\right] \sigma^2_{\text{artifact}}, \tag{3.20}$$

The factor n denotes the estimated number of independent artifacts, which is estimated by $1 - (1 - p_c(j))^L$, where L is a constant for a given encoder and has to be found by curve-fitting. L is used to model the fact that at higher bit-error rates the introduced bit errors are not independent anymore. $p_c(j)$ is the expected bit-error rate at time of transmitting of frame j. For a thorough discussion we refer to our paper [75], included in this thesis as Chapter 5. $\sigma^2_{\text{artifact}}$ and L are encoder dependent parameters in $\mathcal{C}_{\text{v}}$ and assumed to be constant.

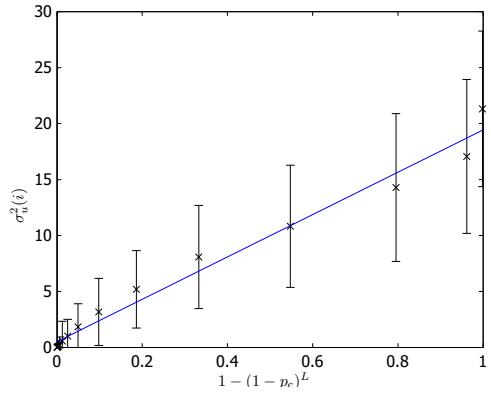
Experiments indicate that this model gives an accurate prediction of the average normalized energy of injected errors, as shown in Figure 3.10(a). In Figure 3.10(b) we see an approximately linear relationship between $(1 - (1 - p_c)^L$ and $\sigma_u^2(i)$ as in Eq.(3.20). In Figure 3.10(a) the line shows that Eq. (3.20) gives a good average approximation of the experimental data (crosses), but with a quite large spread of actual measured distortion. Since we are interested in the average end-to-end quality over multiple frames, the spread in individual frame distortions is not so relevant.

The total residual error distortion that is present at frame i is an accumulation of the induced residual errors from all previous frames.

$$\hat{D}_c(i) = \sum_{j=-\infty}^{i} \sigma_c^2(i,j) = \sum_{j=i-\lfloor \beta^{-1} \rfloor}^{i} \sigma_c^2(i,j). \tag{3.21}$$



(a)



(b)

(c)

Figure 3.10 — *(a) Plot of and residual channel error distortion as function of the number of artifacts $1-(1-p_c)^L$ for frame 2 in the* `foreman` *sequence. For this particular frame $L = 1.61 \cdot 10^3$ and $\sigma^2_{\text{artifact}} = 17.5$ were found by curve-fitting (b) Experimental values of the distortion due to residual channel errors p_c. (c) Example of artifacts due to residual channel errors.*

3.2.9.2 Effect of skipping frames on end-to-end distortion

When the decoder detects that a frame has been skipped at the encoder, it fills in the missing frames by reusing the last received frame until a new frame is received. The visual perception of frame skips is not easy to model, although several attempts were made [92, 85, 22]. We used a model that simply adds a penalty to the distortion for each skipped frame. Since in place of a skipped frame, a previous frame is shown, the MSE distortion is in fact the difference between the current frame and the previous frame. This amount of distortion can be estimated using the prediction gain. We estimate the distortion of frame i, when the last received frame was frame j:

$$\begin{aligned} D_{\text{skipped}}(\mathcal{V}, i, j) &= \text{var}\left[X_j^\dagger - X_i\right] \\ &= \text{var}\left[(X_j^\dagger - X_j)\right] + \text{var}\left[(X_j - X_i)\right] \\ \hat{D}_{\text{skipped}}(\mathcal{V}, i, j) &= D_q(\mathcal{V}, j) + D_c(j) + \frac{\sigma_{X_i}^2}{(G_0 - 1)\exp\frac{1-(i-j)}{N_{\text{mc}}} + 1} \end{aligned} \tag{3.22}$$

assuming independence of quantization errors and frame differences.

We can now estimate the average end-to-end quality Q which is a performance parameter of the video coder ($Perf(\mathcal{V})$). The end-to-end quality is calculated, using the complete chain of models and all encoder settings described above. Since we take skipped frames and propagation of errors into account, we should estimate an average Quality Q over multiple frames. The average perceived quality over frames $j, \ldots, j + (N_{\text{fs}} - 1)$ of which the first one is encoded and transmitted and the following $N_{\text{fs}} - 1$ frames are skipped at the encoder, is predicted by the average PSNR over these N_{fs} frames

$$\begin{aligned} \hat{Q}(\mathcal{V}, j) = \frac{1}{N_{\text{fs}}}\Bigg(&20\log_{10}\frac{255}{D_{\text{non-skipped}}(\mathcal{V}, j)} + \\ &\sum_{i=1}^{N_{\text{fs}}-1} 20\log_{10}\frac{255}{D_{\text{skipped}}(\mathcal{V}, i, j)}\Bigg) \end{aligned} \tag{3.23}$$

Evaluation of the aggregate behavior of this model is performed in the experiments in Chapters4 and 6 in the context of the ARC QoS system.

3.2.10 Resource usage

In order to be able to make fair trade offs between the performance and resource usage, we need to predict the resource usage given the encoder settings. The first resource, NAL bandwidth R_c follows directly from the encoding rate R. For the other resources, delay and CPU-usage, we propose a behavior model.

Motion estimation is the most complex operation in an encoder. We found that the other operations (DCT, quantization etc.) have a relatively constant contribution to the total computation time. However, motion estimation is parametrized by the search range and greatly influences the total amount of computations. We model the time needed to compress a frame as a constant amount $T_\emptyset$ plus an amount depending on the search range l_{me}. Since we assumed a parametrized full-search ME, the amount of motion vectors calculated is proportional to the area of the search range.

$$\hat{T}_{\mathrm{frame}} \;=\; T_\emptyset + t_{1\mathrm{mv}}\,(2\,l_{\mathrm{me}}+1)^2 \tag{3.24}$$

where $t_{1\mathrm{mv}}$ is a calibration parameter being the time needed for evaluating one motion vector.

3.2.10.1 End-to-end Delay

T_v is defined as the maximum allowed end-to-end transmission time between recording and displaying (see Figure 2.1) which yields

$$\hat{T}_v \;=\; T_{\mathrm{record}} + T_{\mathrm{encode}} + T_c + T_{\mathrm{decode}} + T_{\mathrm{display}}. \tag{3.25}$$

We assume T_{record}, T_{decode} and T_{display} to be constants that can be estimated at forehand. T_c is the NAL transmission delay interface parameter that has to taken into account. This leaves T_{encode} to be estimated. Since, in general, end-to-end delay is bounded, T_{encode} has to be bounded as well.

We have to take into account that frame skips also increase encoder delay, since when N_{fs} frames are skipped, it takes at least $\frac{N_{\mathrm{fs}}}{f_{\mathrm{fps}}}$ seconds, before the next frame arrives. Furthermore, the actual encoding time also increases delay. Besides the bound on transmission delay, real-time coding requires the encoding time to be smaller than

$$T_{\mathrm{frame}} \leq \frac{1}{f_{\mathrm{fps}}}, \tag{3.26}$$

since otherwise encoding speed is lower than real time play out speed, making real time compression impossible.

This results in the following equation to estimate encoder delay:

$$\hat{T}_{\mathrm{encode}} \;=\; \begin{cases} \frac{N_{\mathrm{fs}}}{f_{\mathrm{fps}}} + T_{\mathrm{frame}} & \text{when } T_{\mathrm{frame}} < \frac{1}{f_{\mathrm{fps}}} \\ \infty & \text{otherwise} \end{cases} \tag{3.27}$$

3.2.10.2 CPU usage

The CPU-usage fraction C_v is the percentage of time available for video coding including (NAL) channel coding. $C_v = 100\%$ means that all CPU-time is used

for video coding and channel coding and no CPU-time is left for other applications. Per second, we encode $\frac{f_{\text{fps}}}{N_{\text{fs}}}$ frames, resulting in a total CPU fraction

$$\hat{C}_v = \frac{f_{\text{fps}}}{N_{\text{fs}}} T_{\text{frame}} + C_c, \tag{3.28}$$

where C_c is the CPU fraction that the NAL requires.

3.2.11 Parameter Estimation

We realize that this model, when considering all intermediate models for estimating the prediction gain, RD behavior and effect of residual channel errors, is fairly complex and has a limited precision. Evaluation of the accuracy of this model can be done by evaluating its composing sub models, which is done in the previous sections and by evaluating its aggregate behavior. Evaluation of the aggregate behavior is performed in the experiments in Chapters4 and 6 in the context of the ARC QoS system.

Even if we would have validated this aggregated model with many different video sources and many different channel conditions, we would question the value and reliability of that validation. In a real usage scenario we have to assume that we cannot find the optimal model parameters, for complexity reasons and because of the fact that we can only measure behavior after encoding and cannot perform multi-pass encoding in a real-time scenario. Furthermore since video statistics are non-stationary and show large spreads in distortions of frames and blocks, while using the same settings, real-time estimating the model parameters is cumbersome. Therefore, a fair amount of uncertainty will always be present in the model parameters and in the estimations of the resource usage and performance. These imprecisions could be combat by increasing the model complexity: using higher order models or more parameters. But this will also lead to a model for which it is more difficult to find the optimal model parameters in a real time usage scenario.

The real value of this model lies in the fact that we are interested in the estimations of the average behavior over longer periods of time than just one frame. This suggests that an averaging filter could be applied to combat the uncertainty, but which—as a side effect—makes that the predicted behavior reacts slower to changes in the channel conditions or source characteristics.

3.3 Scalable, Layered and Progressive Coding

3.3.1 Introduction

In the previous section we presented real time adaptive video coding as a solution where a tight coupling between video coder and NAL was exploited to be able to adapt to a changing and error-prone network. Scalable coding gives

a solution for video streaming to postpone the decision of the rate at which the video is transmitted until the time of transmission instead of at the time of encoding.

In scalable coding the encoded video streams are organized in such a way, that by selecting only parts of the stream, or sub streams resolution, frame rate, bandwidth or quality of the video can be scaled.

Layered Coding is a form of scalable coding where the stream is actually split up in separate discrete layers or sub streams, which can then be transmitted separately. Upon reception of the first layer, called the base layer, the decoder reconstructs the source, generally at a lower resolution, frame rate or quality. The base layer on itself is perfectly decodable and results in a basic representation of the original data. If the decoder also receives higher layers, it is able to reconstruct at higher quality levels (or resolution or frame rate). An enhancement layer is useless without the corresponding base layer. Layers are ordered or prioritized in the sense that, to decode layer i, the preceding layers $0, \ldots, i-1$ also have to be available at the decoder. Only the base layer (0) depends on no other layer. Depending on the network characteristics and requirements of the application, a subset of the layers is streamed to the client. For instance if channel capacity is limited to R_c, only l layers can be streamed:

$$l \leftarrow \max_{l} R_l \tag{3.29}$$

$$\text{such that } R_l = \sum_{i=0}^{l-1} r_i \leq R_c, \tag{3.30}$$

where r_i are the rates of the individual layers.

With layered coding it is possible to transmit video streams at different rates to clients with different bandwidths. In a network where different channels exist with different levels of QoS, we can transmit each layer over a different channel. The base information over the channel with the highest QoS, and the refinement information over the channel with the lowest QoS.

Progressive coding is another form of scalable coding that obtains rate or quality scalability but without explicitly producing separate layers. The bit stream and the encoding process are organized in such a way that the most important information, and often the coarsest structures in the frame, are encoded first and appear at the beginning of the bit stream. As the encoding continues, the encoded information progressively becomes less and less important and more fine-grain. The decoder which processes this progressive bit stream is now able to reconstruct a coarse representation of the source, and to progressively refine this representation, until all bits are processed or until another criteria is met. Such a stream is truncate-able at any point and still leaves a decodable sub stream. Another advantage of progressive coding is that errors occurring while transmitting a progressive bit stream have a lim-

ited effect. Only the less important information, positioned *after* the bit-error, will have to be discarded while the first part can still be decoded.

Scalable, layered, and progressive coding are often used interchangeably and often more than one of these applies to a particular coder. For instance, the scalable picture coding standard JPEG2000, is able to produce progressive bit streams, but also implements resolution, and quality scalability.

Scalable coding is also called 'Successive Refinement of Information' (SRI). W. Equitz studied this topic [29] which is based on MDC [1] theory. Thus, layered coding can be seen as a special case of MDC. The author calls a source successively refinable when at each refinement iteration the rate distortion bound of that source can be achieved. Only a limited set of signals is successively refinable, such as Gaussian signals with MSE distortion. Examples of signals are known which are not successively refinable [29]. In practice however, not achieving the rate distortion bound is not necessarily a problem. A small penalty in bit rate is often acceptable when a scalable encoder is required, as is the case with for instance H.264/SVC.

3.3.2 Differential Layered Coding

Perhaps the most basic layering technique is differential encoding. In differential layered coding layers are encoded step-by-step and reconstruction is done by successively adding refinements to the base reconstruction [80]. It can be used to implement layering using standard encoder building blocks.

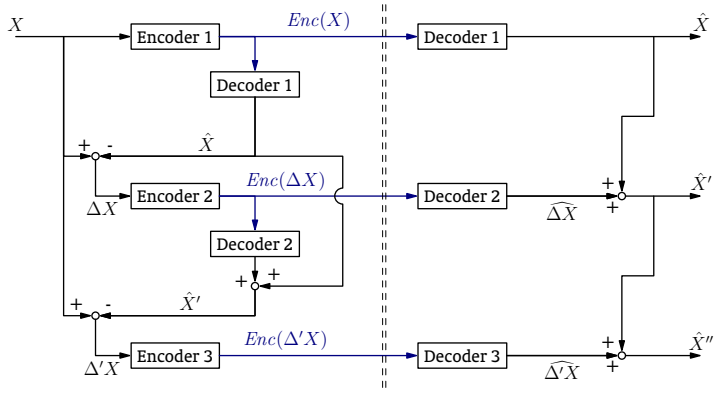


Figure 3.11 — *Differential Layering with three layers. Left the encoder side using generic encoders and decoders. Right the receiving decoder side.*

We give an example of a differential layered encoder in Figure 3.11. Encoding X results in encoded signal $Enc(X)$. Decoding base layer $Enc(X)$ again results in a course representation $Dec(Enc(X)) = \widehat{X}$. We define the residual

or differential signal $\Delta X = X - \widehat{X}$ and mean-square-error distortion

$$D = \mathrm{E}\left[(X - \widehat{X})^2\right] = \mathrm{E}\left[(\Delta X)^2\right].$$

If we also decode $Enc(X)$ at the encoder. we can construct the residual signal ΔX. We now can encode this residual signal with another, finer encoder, which results in a decoded signal $\widehat{\Delta X}$ with decoding error $\Delta' X = \Delta X - \widehat{\Delta X}$.

To reconstruct the image with the enhancement data, the decoder simply has to add the decoded base signal $\widehat{X}$ with decoded enhancement data $\widehat{\Delta X}$:

$$\begin{aligned} \widehat{X}' &= \widehat{X} + \widehat{\Delta X} \\ &= \widehat{X} + \Delta X + \Delta' X \\ &= \widehat{X} + (X - \widehat{X}) + \Delta' X \\ &= X + \Delta' X \end{aligned} \tag{3.31}$$

As a result, the reconstruction of X using base and enhancement layer, $\widehat{X}'$ is distorted with a residual signal $\Delta' X$ and yields a distortion $D' = \mathrm{E}\left[(\Delta' X)^2\right]$. Note that distortion ΔX itself is eliminated in the last step of Eq.(3.31) and is no longer present in the reconstruction of the second layer, This structure easily scales to generate any number of enhancement layer. Thus every layer decreases the effect of the reconstruction distortion of the previous layer.

With differential layered encoding, we can in principle make a layered coder from any non-scalable encoder. In Chapter 7 we describe the use of a scalable encoder based on the non-scalable Dirac encoder, developed by the British Broadcasting Corporation [26]. There we used the differential encoding scheme to generate multiple layers for each single frame. Only for the base layer motion compensation is used. This on one hand leads to less efficiently encoded enhancement layers, but on the other hands makes it possible to decode any enhancement layer without depending on previous frames. Figure 3.12 shows the PSNR–rate curves for the layered Dirac encoder. Each branch point corresponds to a different base-layer rate and base quality and each branch shows the quality-rate curve for the enhancement layer. For comparison also the H.264 curve is shown. We clearly see that there is a penalty in quality if a low base rate is chosen, due to the lack of motion compensation for the enhancement layers.

3.3.3 Progressive Coding: Bit-plane Coding

Bit-plane coding is another method based on successively refining layers [49]. In bit-plane coding, we code the signal in a progressive manner. Often bit-plane coding is preceded by Discrete Cosine Transform (DCT) decorrelation and uniform quantization. Instead of coding every quantized coefficient separately, we

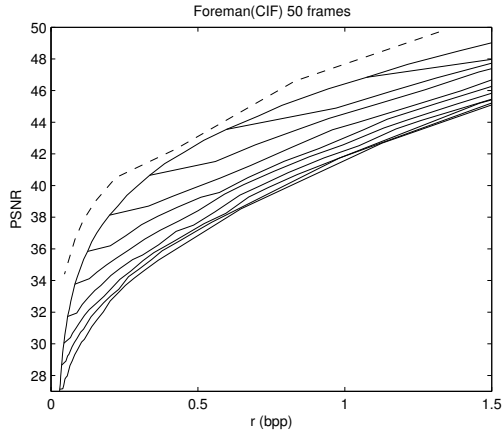


Figure 3.12 — *Quality–Rate curves for a two-layer Dirac encoder. Each branch has a different rate for the base layer. The dashed line shows the RD curve for the H.264 codec.*

take all i^{th} bit of all pixels in a block together, forming a bit plane and encode them efficiently using variable length coding. By starting with the most significant bit (MSB) and working towards the least significant bit (LSB), a progressive bit stream is produced which can be cut off between any bit plane.

Examples of still-picture compression standards that are progressive are Embedded Zerotree Wavelet (EZW) [66], Set Partitioning in Hierarchical Trees (SPIHT) [63] and JPEG2000 [69]. JPEG2000 is used in Chapter 5.

3.4 Multiple Description Coding

Multiple Description Coding (MDC) is described by Goyal and Kovaçevic [32] as: '[...] source coding in which several descriptions of the source are produced such that various reconstruction qualities are obtained from different subsets of the descriptions.'

With MDC several descriptions are generated, each of which can independently be decoded. When more descriptions are received, the decoder can improve the reconstruction quality, much in the same way as Layered Coding. However, in layered coding all *lower* layers have to be present to improve quality, whereas for (symmetric) MDC any description will improve quality. To achieve this layered-less property of MDC, the price we have to pay is that we have to introduce redundancy in the bit stream. On one hand, in both cases it depends on the networks characteristics how many descriptions/layers can be transmitted without congesting the network. On the other hand, MDC is

inherently error resilient, missing a description has no great impact, whereas not receiving a base layer of layered coded sequence has a great impact on the quality.

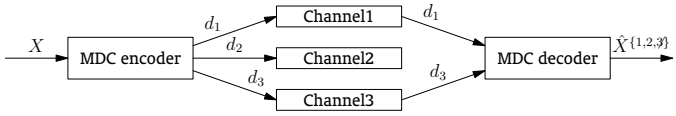


Figure 3.13 — *Multiple Description Coding scenario*

Figure 3.13 show a usage scenario of MDC where the NAL offers different channels for transmitting video to a client. When each of these channels is prone to errors, the decoder will receive only a subset of the parts that were transmitted. In principle all combinations are possible of what parts will arrive at the decoder and what parts not. Ideally, the packet-loss events on these paths would be independent of eachother. Under these circumstances, the decoder has to reconstruct the video stream as good as possible. In our VSM the NAL offers different channels with known rate and known average packets loss rates. The MDC encoder can then be designed to match these channels such that an optimal average quality is received by the user. In Chapter 7 we use MDC in a multi-client scenario using a P2P network.

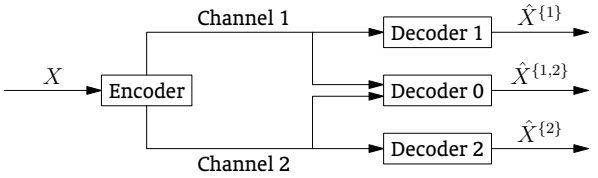


Figure 3.14 — *Two Description Source coding, with one source, two channels and three decoders.*

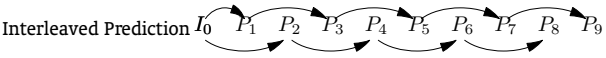
In Figure 3.14 a two-description coding case is illustrated. We have one input signal X, but the encoder generates two descriptions which are independently transmitted over two channels. Depending on how many descriptions are received, a different decoder is used. The central decoder 0 reconstructs $\widehat{X}^{\{1,2\}}$ using descriptions 1 and 2, resulting in central distortion D_0. The side decoder 1 uses description 1 to reconstruct signal $\widehat{X}^{\{1\}}$, resulting in side distortion D_1. The same holds for decoder 2 and $\widehat{X}^{\{2\}}$.

To design an MDC system, one has to know network characteristics such as the number of channels, the capacity per channel and the error rate per channel. In addition, one has to take scenario properties, such as the number of clients into account. Depending on these properties, one can optimize the

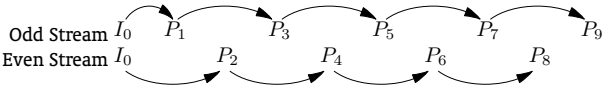
number of descriptions and the amount of redundancy, given a certain MDC method. In this section we will first show a simple MDC method. Then we discuss the trade off between error-resilience the amount of redundancy and the rate distortion bounds for two-description encoding of Gaussian sources. After that different practical MDC techniques are discussed.

3.4.1 Odd-Frame / Even-Frame Streaming

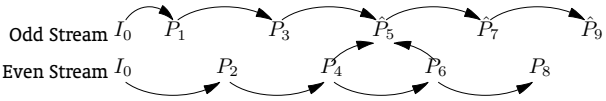
Odd / Even Frame Streaming is a simple implementation of MDC [5, 15]. Consider a video sequence at 30 frames per second. Normally, a video coder would apply a predictive coding scheme such as $IBBPBBP\ldots$ or $IPPPP\ldots$. The latter will be used in this example for simplicity reasons. We know that this prediction scheme is vulnerable to losses. Any lost P-frame results in a cascade of errors in subsequent P-frames, until a new I-frame arrives. To make our encoding less vulnerable to these errors, we could insert more I-frames, but these consume much bandwidth. We would like to have a mechanism to repair a missing frame, such that we can prevent the cascade of errors. The problem lies in the fact that all P-frames depend on all previous P-frames until the most recent I-frame. Suppose that in our temporal prediction scheme, for each prediction we skip *one* frame as in:



When we pull apart the two thus created sub streams and encode them separately, we have made two descriptions that both have half the frame rate (15 fps) and are independently decodable.



When the receiver combines the two streams, a complete reconstruction at the full frame rate is possible. Now suppose one frame is lost. First of all the other sub stream is completely unharmed. We could from that point on, only display the unharmed sub stream, effectively reducing the frame-rate. However, we can improve this solution by temporally interpolating the missing frame using the surrounding frames, thereby introducing a relatively large—but acceptable— prediction error. The following dependent frames in the harmed sub stream will propagate this prediction error but will not be affected as much as when no interpolation was performed [5].



Using this solution, we have increased the error resilience. However, in the single description case ($IPPP\ldots$) we obtain a better prediction gain since we predict upon frames that are closer in time. In the MDC case, the prediction gain will be smaller, which in turn results in a higher total bit rate to code the two descriptions. The extra bit rate needed for this is the redundancy introduced by MDC.

3.4.2 Theory and special cases of MDC

To explain Multiple Description Coding confine ourselves to a Gaussian source. Let $X_1, X_2, \ldots$ be a sequence of i.i.d. Gaussian random variables with $\sigma_X = 1$. According to Shannons rate-distortion (RD) theory at least these inequalities have to hold:

$$D_1 \geq 2^{-2R_1} \tag{3.32}$$

$$D_2 \geq 2^{-2R_2} \tag{3.33}$$

$$D_0 \geq 2^{-2(R_1+R_2)}. \tag{3.34}$$

In 1979, Gersho, Ozarow, Witsenhausen, Wolf, Wyner and Ziv, posed the question of what the achievable rates and qualities are for a two-description encoded source [28]. In 1980, Ozarow presented the tight bounds for memoryless Gaussian sources with a mean square error distortion [54]:

$$D_1 \geq 2^{-2R_1} \tag{3.35}$$

$$D_2 \geq 2^{-2R_2} \tag{3.36}$$

$$D_0 \geq 2^{-2(R_1+R_2)} \cdot \gamma(D_1, D_2, R_1, R_2) \tag{3.37}$$

where

$$\gamma(D_1, D_2, R_1, R_2) = \begin{cases} \frac{1}{1-\left(\sqrt{(1-D_1)(1-D_2)}-\sqrt{D_1 D_2 - 2^{-2(R_1+R_2)}}\right)^2} & \text{for } (D_1+D_2) < 1+2^{-2(R_1+R_2)} \\ 0 & \text{otherwise.} \end{cases} \tag{3.38}$$

From these tighter bounds we see that besides the Shannon bounds on the two side distortions a stronger bound is put on the central distortion, which is dependent on the magnitude of the side distortions and the bit rates of the descriptions. The implications of these inequalities are not immediately clear so Goyal [35] analyzed these inequalities in three cases. Goyal defines the *base rate* r as the rate that is needed to obtain distortion $D_0 = 2^{-2r}$ with a single description coder. He defines the excess rate or *redundancy* as $\rho = R_1 + R_2 - r$. For more in depth derivations we refer to Ref. [35].

Individually good If we assume that both descriptions are individually as good as possible, i.e. when Eqs. (3.35) and (3.36) become equalities: $D_i = 2^{-2R_i}$, then (3.37) reduces to

$$D_0 \geq \frac{1}{2}\min(D_1, D_2).$$

We interpret this as follows: if both descriptions are individually encoded as good as possible, the central distortion can only be marginally (1 bit) better than the best side description.

The redundancy in this case is always

$$\rho > \min(R_1, R_2) - 1/2.$$

For high rates the redundancy is then almost as high as the rate for a single description.

Making two exactly the same descriptions (duplicating) is an example of the 'individually good' case, where $D_0 = D_1 = D_2$ and $\rho = R_1 = R_2$.

Jointly good descriptions If the joint reconstruction is as good as possible, then $D_0 = 2^{-2(R_1+R_2)}$. In this case $\gamma = 1$ which leads to

$$D_1 + D_2 \geq 1 + 2^{-2R_1+R_2}.$$

This result can be interpreted as at least one side distortion being very large. The redundancy in this case is minimal since $r = R_1 + R_2$ yielding $\rho = 0$.

Layered coding is a form of jointly-good MDC, or in other words: layered coding is a special case of MDC. When $D_0 = 2^{-2(R_1+R_2)}$ and $D_1 = 2^{-2R_1}$, description 1 forms a base layer and description 2 forms an enhancement layer. In this case D_2 tends to one for sufficiently large R_1.

Symmetric descriptions Suppose we fix $R_1 = R_2$ and $D_1 = D_2$. Ozarows inequalities can then be rewritten as

$$D_1 \geq \begin{cases} \frac{1}{2}[1 + 2^{-2r} - (1 - 2^{-2r})\sqrt{1 - 2^{-2\rho}}] & \text{for } \rho \leq \rho_T \\ 2^{-(r+\rho)} & \text{for } \rho > \rho_T \end{cases} \tag{3.39}$$

$$\rho_T = r - 1 + \log_2 1 + 2^{-2r} \tag{3.40}$$

$$D_0 = 2^{-2r} \tag{3.41}$$

Figure 3.15 show the achievable rates in the symmetric case.

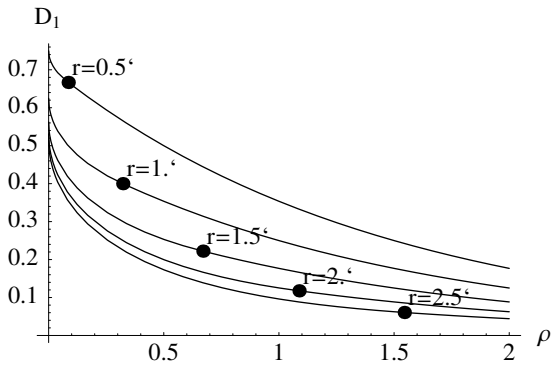


Figure 3.15 — *Achievable rates in a two-description coding system for different values of the Base rate r*

It is argued in [35], that the Symmetric MDC case is an intermediate case between the first two cases. This is not always true since the individually-good and the jointly-good cases may be asymmetric. We like to present a fourth case: Asymmetric MDC

Asymmetric descriptions In all cases where $D_1 \leq D_2$, but D_2 not necessarily very large, we speak of asymmetric multiple description coding (AMDC). All combinations of descriptions may yield a different quality (regardless whether these qualities are acceptable or not). Similar to the symmetric case, redundancy may be added to make D_0 lower while keeping D_1 and D_2 constant.

The asymmetric case can be seen as a real intermediate case, since it spans all cases between the symmetric case, the individually-good case and the jointly-good case. In general, AMDC subsumes the three cases as special cases. Figure 3.16 clarifies this in $\rho \times (D1/D2)$ space.

In the remainder of this section we only consider symmetric MDC. Asymmetric MDC is further presented and discussed in Chapter 8. The cases presented above make clear that when we expect to have the lowest side distortions possible, we cannot obtain a substantially lower D_0. And when we expect the lowest central distortion D_0, we cannot have two low side distortions at the same time. We can, however, make a trade off between low central and low side distortion by controlling the amount of redundancy. At the expense of some excess rate ρ we can decrease the side distortions $D_{1,2}$ without increasing D_0.

If bandwidth is limited, the amount of redundancy has to be tuned accurately in order to minimize average distortion. Assume that the total rate is

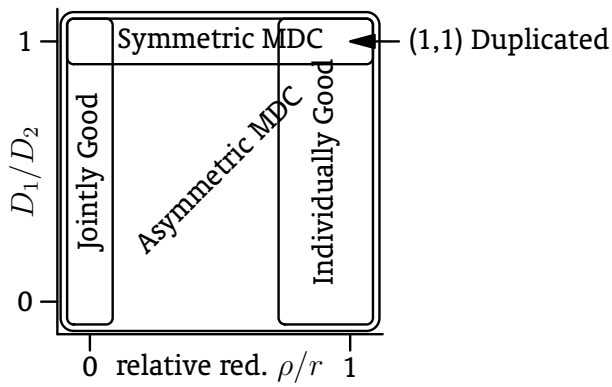


Figure 3.16 — *Special cases of Multiple Description Coding: jointly-good (includes layered coding), symmetric MDC, individually-good (includes duplication), and AMDC. We can discriminate between the special cases by looking at the D_1/D_2 ratio and on the relative amount of redundancy ρ/r.*

limited to $R_T = R_1 + R_2 = r + \rho$ and suppose channel 1 has a packet loss probability p_1 and channel 2 of p_2. The average distortion is then:

$$\begin{aligned} D_T \; &= p_1\, p_2 \;\; + \; (1-p_1)\, p_2\, D_1(r,\rho) + \\ &\quad\; p_1\, (1-p_2)\, D_2(r,\rho) + \\ &\quad\; (1-p_1)\, (1-p_2)\, D_0(r,\rho) \end{aligned} \tag{3.42}$$

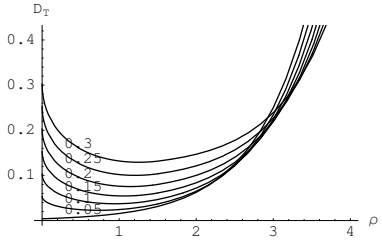
Minimizing D_T yields ρ_{opt}. We define:

$$R_T = R_1 + R_2 = r + \rho.$$

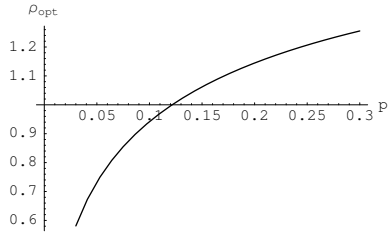
To get feeling for typical values of the redundancy parameter under different conditions we present an example for a Gaussian i.i.d. source, and $R_T = 4$ and $p_2 = p_1 = p$, Figure 3.17(a) illustrates the impact of increasing redundancy on the average (received) distortion for different packet-loss probabilities. Since we choose R_T constant, we trade redundancy ρ off with base rate r. The rather flat behavior at the minima means that D_T is not very sensitive to small variations of ρ.

For each packet-loss-rate we can find the optimal redundancy that minimizes the distortion D_T. In Figure 3.17(b) the value of the optimal redundancy as function of the packet-loss rate is plotted. Close inspection shows that a practical range of redundancies is from 5% to 30% for packet loss rates up to 20%. A remarkable result is that ρ_{opt} never goes beyond the ρ_T threshold in (3.40).

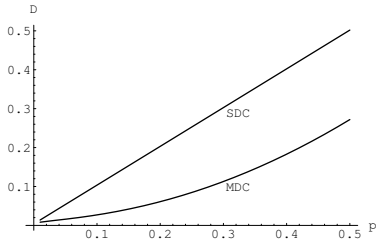
We compare single description coding (SDC) with MDC. In Figure 3.17(c) the average distortion is plotted for SDC $D_{\mathrm{SDC}} = (1-p)2^{-2R_T} + p$ and for MDC (Eq. 3.42) while using optimal redundancy ρ_{opt}. Finally, in Figure 3.17(d) the gain in dB when MDC is used instead of SDC as function of packet-loss rate is plotted. An improvement of several dB's can be obtained when MDC is used for packet-loss rates $p < 0.6$



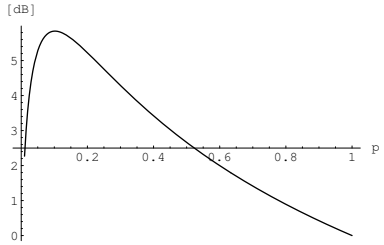
(a) Plot of (3.42) for different packet-loss probabilities



(b) Optimal ρ minimizing (3.42) for varying packet-loss probability



(c) Average distortion of MDC and SDC for varying packet-loss probability



(d) Gain in [dB] when two-description MDC is used instead of SDC

Figure 3.17 — *Optimizing redundancy in the case of $R_T = 4$ and $p_{1,2} = p$.*

3.4.3 MD Nested Quantization

MD Nested Quantization is a method where quantizers are designed to give suboptimal distortion. However, iff the two quantization indices are combined, resulting in a smaller quantization cell, a finer reconstruction can be obtained and hence a lower central distortion. Vaishampayan introduced nested scalar (MDSQ) [84] and nested vector quantization (MDVQ) [83, 25]. Nested quantization is an example of a MDC method where a controlled trade off between D_0 and D_1 is made.

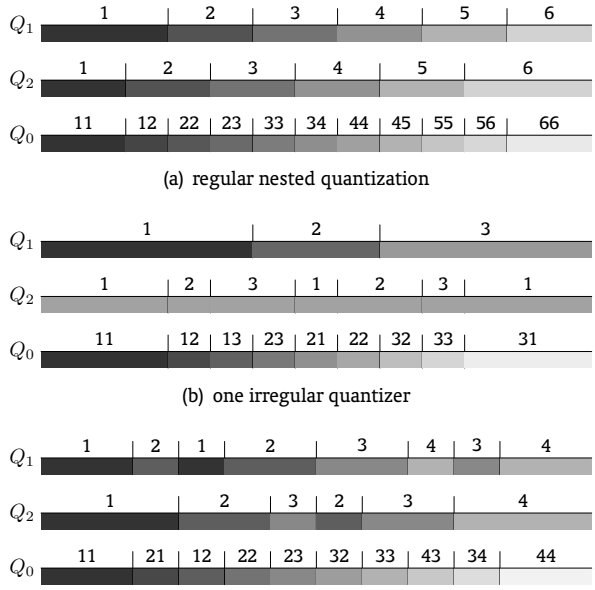


Figure 3.18 — *Multiple Description Nested Quantization*

Figure 3.18 depicts three examples of two quantizers that operate on the same source (X). The shown numbers are the index assignments for each cell and do not reflect reconstruction value. The greyvalue of the cell represents the reconstruction value of the corresponding index. Source signal X is quantized independently by 2 quantizers Q_1 and Q_2, producing $\hat{X}^{\{1\}}$ and $\hat{X}^{\{2\}}$. As always, the quantizers have to be designed to match the pdf of the source signal. Here, the MDC quantizers are designed to have a good reconstruction when only one quantizer index can be decoded but to have a better reconstruction when both quantizers indices are available. The quantization levels of Q_1 and Q_2 are in the same signal space, but are offset with respect to each other, hence the name Nested Quantization.

In Figure 3.18(a) two uniform quantizers (Q_1 and Q_2) that are shifted a little with respect to each other are shown. Receiving the index for one of the two gives a coarse approximation $\hat{X}^{\{1\}}, \hat{X}^{\{2\}}$. When the indices for both quantizers are received, the central decoder combines these indices and uses the intersection of the cells of the individual quantizers to reconstruct $\hat{X}^{\{1,2\}}$. The central decoder effectively decodes as if a central quantizer Q_0 was used. In this case, when both descriptions are received, this results in the MSE becoming four times as small. This case corresponds with the (highly redundant)

individually-good MDC case.

Figure 3.18(b) shows a coarse quantizer Q_1 and a finer but irregular quantizer Q_2. Receiving only the output index of Q_1 gives a coarse but acceptable approximation $\hat{X}^{\{1\}}$. Receiving only the index for Q_2 gives essentially no useful information about X. Receiving both indices gives a fine quantized version of X. This case is in fact a form of Layered Coding and corresponds to the 'jointly good' case. Q_1 produces the base layer and Q_2 produces the enhancement layer.

The third case presented in Figure 3.18(c) has two irregular but still useful quantizers. When only one description is received, there is substantial but acceptable uncertainty about X. Receiving both descriptions suddenly gives a very accurate approximation of X. This case corresponds to the symmetrical MDC case in which a controlled amount of redundancy is introduced to obtain a tradeoff between low side and low central distortions.

The nested quantization MDC method works on scalar data, but can also be extended to work on vectors. It offers control over the redundancy parameter. A disadvantage is that the different quantizer tables have to be communicated to the decoder and as such is less usable for non-stationary signals.

3.4.4 Multiple Description Correlating Transform

Since multiple descriptions describe the same data and are independently decodable, descriptions are correlated by definition. If one description is known, the others can be predicted to some extent. Wang, Orchard and Reibman [91] introduced the multiple description correlating transform (MDCT) that inserts a controlled amount of redundancy in otherwise uncorrelated pairs of random variables. Goyal improved this work by using Integer Transforms [33], which circumvents the sub optimality of non-square quantizer cells [35].

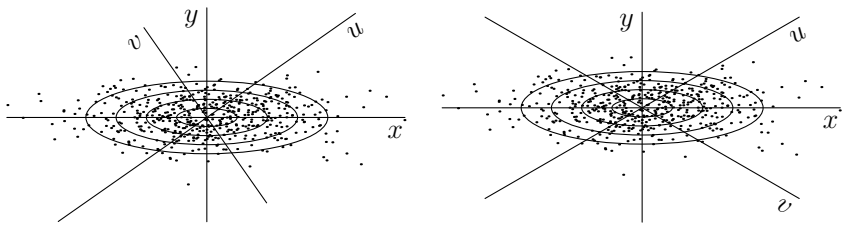
The basic idea is to correlate two independent zero-mean random variables x and y:

$$\begin{pmatrix} u \\ v \end{pmatrix} = T \begin{pmatrix} x \\ y \end{pmatrix}$$

to obtain two correlated variables u and v (See Figure 3.19(a)). If we know u we can make an estimate of v and *v.v.* Goyal derived the requirements of T for minimizing the distortion given a redundancy value of ρ.

The main result in this work of Goyal is the redundancy – side distortion curve for correlating transforms shown in Figure 3.20. Also shown is an approximation of Ozarows bound for two sources (line) and experimental results for MDSQ (circles). Goyal found that for independent Gaussian sources $[x, y]$ with standard deviations $\sigma_x > \sigma_y$:

$$D_1 \;=\; D_2 \;=\; \frac{1}{2}\sigma_2^2 + \frac{\sigma_1^2 - \sigma_2^2}{42^{2\rho}\left(2^{2\rho} + \sqrt{2^{4\rho} - 1}\right)} \tag{3.43}$$



(a) Orthogonal transform (b) Non-orthogonal transform

Figure 3.19 — *Operation of the correlating transform. Samples are projected from* $[x, y]$ *axes onto the new* $[u, v]$ *axes with* $[u, v]' = T[x, y]'$*. Goyal uses the non-orthogonal transform combined with integer transform since they give better performance.*

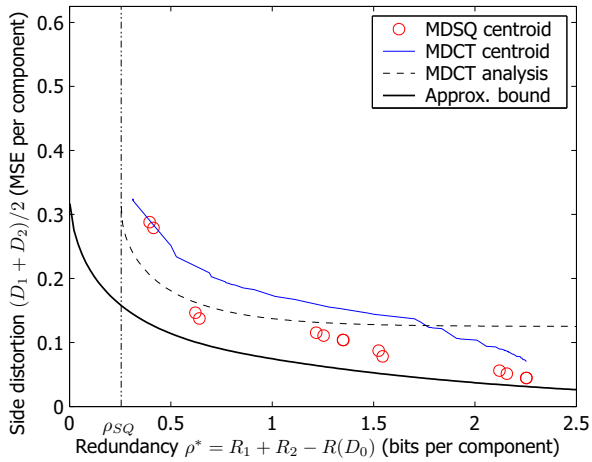


Figure 3.20 — *Redundancy – side distortion curve of the correlating transform (MDCT analysis). Also shown is the experimentally obtained RD points (solid line), Ozarows bound (thick black line). For comparison the experimental RD-points of MDSQ are also shown*

We can draw several conclusions from this curve. For increasing ρ, D_1 asymptotically goes to σ_2^2. This indicates that for MDC to be efficient the source should consist of independent signals of which the variances should differ sufficiently. Secondly, other methods such as MDSQ have better results in the high redundancy region. A benefit of using MDCT is that a continuous range of redundancies can be selected. Goyal found that for orthogonal Gaussian sources $[x, y]$ with standard deviations σ_x and σ_y, the optimal T is generally not a

rotation but a non-orthogonal transform such as in Figure 3.19(b).

A drawback of this method is that the source should consist of multiple independent and zero-mean random variables, of which the variances should differ sufficiently.

3.4.5 MD Channel Codes

With MDC, we protect the signal by introducing redundancy and correlation between each description, such that whenever some description is lost, the decoder is able to estimate the missing data and to reconstruct the source at a reasonable quality level. In transmission channels the same goal is achieved by applying and transmitting channel codes, such that missing blocks of data can be completely reconstructed when enough data is received. For instance with Reed-Solomon Codes, k symbols of data, are protected with $(n-k)$ parity symbols, making together n symbols. Whenever at maximum any $n-k$ of these symbols is lost, the original k symbols still can be recovered without any loss or distortion. The reconstruction quality when using channel codes does not gracefully degrade when more than $n-k$ symbols are lost.

MD Channel Codes is a form of MDC where the Distortion D_m when m descriptions are lost is minimal when $m \leq n-k$. When $m \geq n-k$ no data can be reconstructed and, resulting in maximum distortion. As such the resulting reconstruction quality is binary: all or nothing. Still, Channel Codes are often used for any type of data since it can be implemented in network adapters and scales easily to high numbers of packets or descriptions.

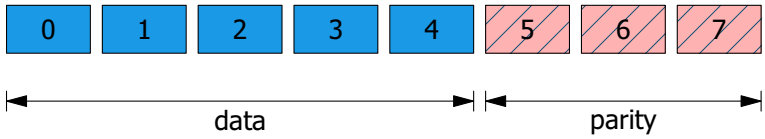


Figure 3.21 — *Example of an $(8,5)$ Reed Solomon Erasure Codes, whenever at least 5 out of 8 blocks or descriptions are received, the original 5 blocks of data ($0,1\ldots4$) can be reconstructed. Blocks $5,6,7$ are called code blocks or parity blocks.*

3.4.6 MD-FEC

The binary behavior of channel coders, described in the previous section, is in most cases unwanted, since often a more gradual or graceful quality degradation is demanded. Multiple description coding with forward error correction (MD-FEC) has been proposed as a method combining the ease of use of FEC

and layered coding [20]. The idea is similar to layered coding combined with unequal error protection (UEP) [93].

The goal is to increase quality with every description that is received. To achieve this, the data of every layer is protected by an erasure code. Suppose we generate M descriptions, the first layer then is protected by a $(M, 1)$ erasure code. The M parts of the code comprise the M descriptions. Now as long as we receive at least one description, the $(M, 1)$ code ensures that we can always decode the first layer.

The second layer is protected by a $(M, 2)$ code and so on for layer i, which is protected by a (M, i) code, until layer M, which is protected by a (M, M) code, and effectively is unprotected. For each layer, the M parts are equally distributed over the M descriptions. This results in having M descriptions, each of which contain channel codes from *all* layers. This structure of protecting, as shown in Figure 3.22, ensures that whenever m descriptions are received, all layers $1, \ldots, m$ can be decoded. Instead of MD-FEC, we rather speak of layered coding with lateral error correction (LC-LEC), since the error-correcting codes are working *laterally* over the descriptions and not sequentially over a block of data. Furthermore, it better expresses the fact that layered coding is used to generate multiple layers.

We compare pure LC with LC-LEC. A source that is layered-coded, results in a hierarchical set of layers, since each layer depends on lower layers. With symmetric MDC in general, and therefore also for LC-LEC, descriptions can be decoded independently. So the use of erasure codes in LC-LEC in fact transforms a set of hierarchical layers into independently decodable descriptions. This independence and increased flexibility comes with a price, namely redundancy, since the introduced parity data is redundant when all descriptions are received. The total amount of redundancy depends on the number of descriptions M and on the rates at which layer is encoded. If a small, low quality base layer is used, total redundancy is lower than when a large base layer is used.

We consider a M description code. We generate M Layers with an LC with rates $r_1, r_2, \ldots, r_M$ for each layer. This yields a distortion profile $\mathcal{D} = \{D_0, D_1, D_2, \ldots, D_M\}$, giving the distortion for when $0, 1, \ldots, M$ layers are decoded, respectively. By applying an (M, i) erasure code to each layer i we form descriptions with a rate of

$$R_D \sum_{i=1}^{M} \frac{r_i}{i}. \tag{3.44}$$

As a result of adding erasure codes, whenever m descriptions are received and decoded, this will yield a distortion D_m, no matter *which* descriptions are received. The total amount of redundancy we have inserted in our descriptions

is

$$\rho = \sum_{i=1}^{M} r_i \, (M - i), \tag{3.45}$$

where $r_0 = 0$. This amount of redundancy can be influenced by changing the rate of each layer, which will also result in a different distortion profile.

The optimal amount of redundancy depends on the required amount of error-resilience given the network capacity and packet-loss rate. An algorithm to find a good redundancy trade off is presented in Chapter 7 and [59]. In Chapter 7 LC-LEC is used in P2P streaming scenario to stream video to multiple clients with different bandwidths.

Another benefit of LC-LEC is that we can make asymmetric descriptions for any asymmetric set of channels, by changing the distribution of channel codes over the descriptions. In Chapter 8 this is discussed in more detail.

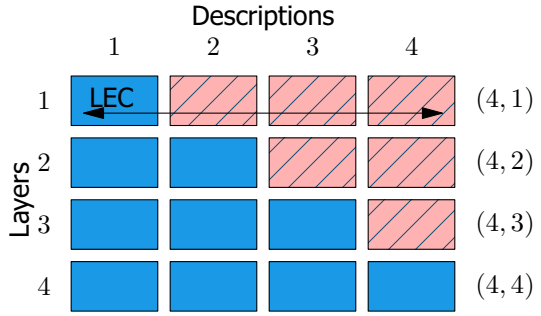


Figure 3.22 — *Illustration of a 4-description LC-LEC configuration. Each column comprises a description, consisting of parts from each layer. The LEC works across the descriptions.*

3.5 Discussion

In this chapter we presented three different strategies to implement the video coder in a network aware system. In the chapters in Part II, we use these different methods. The choice of the type of encoder is made on the basis of the type of network, the number of receivers (one or many) and other scenario constraints such as the platform limitations.

The first method, RAVC relies heavily on a behavior model of a (existing) video coder. RAVC is used in Chapters 4 and 6. RAVC depends on the availability of up-to-date network status information through a QoS mechanism. Besides adapting to changing rate and bit-error-rate, this method also takes other resources such as cpu-power and delay into account.

The second type of encoder layered coding offers a way to create multiple streams targeted at specific channel rates and loss rates, without requiring complex behavior models. The way the encoder settings are controlled depends on the scenario in which it used. We use the progressive JPEG2000 encoder in Chapter 5, where we also use behavior models and the network context to optimize the settings. LC is also used as the basis for implementing MD-FEC.

MDC as the third type of encoding discussed in this thesis also relies on having knowledge of the channel capacities and loss rates. As opposed to layered coding, symmetric MDC is more targeted to channels with similar transmission rate and packet-loss rates. In Chapter 7 the encoder (rate), we discuss a method where the encoder settings are controlled by a rate control algorithm that takes the client bandwidths of all clients into account. In Chapter 8 we focus on *asymmetric* MDC and present an allocation algorithm to match the AMDC encoding to a specific configuration of asymmetric channels.

Part II

Streaming Scenarios

In the following five chapters we have included the main parts of the following five papers.

Chapter 4 J. R. Taal, K. Langendoen, A. van der Schaaf, H. W. van Dijk, and R. L. Lagendijk, 'Adaptive end-to-end optimization of mobile video streaming using QoS negotiation,' in *ISCAS, special session on Multimedia over Wireless Networks*, vol. I, Scottsdale, AZ, May 2002, pp. 53–56. [73]

Chapter 5 J. R. Taal, Z. Chen, Y. He, and R. L. Lagendijk, 'Error resilient video compression using behavior models,' *EURASIP Journal on Applied Signal Processing*, no. 2, pp. 290–303, Feb. 2004. [75]

Chapter 6 I. Haratcherev, J. R. Taal, K. Langendoen, R. L. Lagendijk, and H. Sips, 'Optimized video streaming over 802.11 by cross-layer signaling,' *IEEE Communications Magazine*, vol. 44, no. 1, pp. 115–121, Jan. 2006. [41]

Chapter 7 J. R. Taal and R. L. Lagendijk, 'Fair rate allocation of scalable multiple description video for many clients,' in *Proc. of SPIE, Visual Communications and Image Processing*, vol. 5960, July 2005, pp. 2172–2183. [76]

Chapter 8 J. R. Taal and R. L. Lagendijk, 'Asymmetric multiple description coding using layered coding and lateral error correction,' in *Proc. twenty-seventh Symposium on Information Theory in the Benelux,*. Noordwijk: Werkgenootschap Informatie- en Communicatietheorie, June 2006, pp. 39–44. [77]

Four

Adaptive End-To-End Optimization Of Mobile Video Streaming Using QoS Negotiation

4.1 Introduction

Mobile systems often operate in a highly variable context. The two dominant factors causing context variability are the user and the mobile channel. The characteristics of mobile transmission strongly depend on the user's location and environment. This is reflected in variable throughput, reliability, and required transmission power. The user context is variable because mobile systems are often designed to support multiple interactive services, which impose different workloads on the system in different situations. Handling the variable context is not the only requirement for mobile systems. They must also be efficient, since resources (especially battery energy) are scarce in mobile systems. Handling variable workloads efficiently under variable conditions necessitates the use of collaborative adaptive modules.

The mobile system that we consider here supports video streaming over a wireless link. It consists of various adaptive modules, including a video encoder and protocols (see Figure 4.1). The video encoder is driven by a workload obtained from an application module, and the protocols induce a workload on a radio module. The application module directly experiences the user context variability, while the radio module is subjected to the fluctuating conditions of

This chapter was published as: J. R. Taal, K. Langendoen, A. van der Schaaf, H. W. van Dijk, and R. L. Lagendijk, 'Adaptive end-to-end optimization of mobile video streaming using QoS negotiation,' in *Proc. ISCAS, special session on Multimedia over Wireless Networks*, vol. I, Scottsdale, AZ, May 2002, pp. 53–56. [73]

the mobile channel.

In Figure 4.1 it is apparent that the video encoder and the protocol modules are not directly influenced by any context variability at either side of the system. However, if we optimize the system under global efficiency constraints, then the video encoding and protocol module will indirectly experience context fluctuations from neighboring modules when they adapt. Therefore, all modules will have to be context dependent, requiring their internal operation to be flexible and adaptive to local context fluctuations.

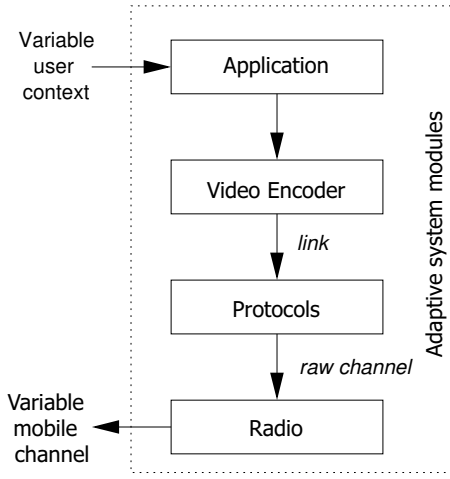


Figure 4.1 — *Mobile video communication system.*

Each module in an adaptive system usually contains many parameters that influence its behavior. System-wide optimization of the decision variables is difficult for two reasons. First, jointly optimizing many parameters yields computationally complex solution strategies. Second, every single module is a complex system in itself, whose application requires specific domain knowledge. To keep the optimization procedure manageable, we have to decompose the global optimization problem into several smaller problems. Ideally, we can perform independent optimization for each individual module. However, as we have argued, the optimization of one module depends on the context and inner workings of other modules. Therefore, the adaptation of individual modules cannot be optimized separately [31, 47]. Instead, context and implementation details of the components must be exposed and shared, but in an appropriate format.

The problem we address here is therefore essentially one of coordination. Figure 4.1 gives a computational view on the system, emphasizing functionality but hiding implementation aspects. Ideally the component-specific pa-

rameters must be tuned such that system behavior complies to its objectives while being constrained by its conditions. Taking an engineering view on the system, however, introduces additional conditions. Suppose we opt —for this experiment— for a straightforward implementation. We then effectively map the functionality of Figure 4.1 to the limited processing resources of a mobile terminal. In a simple battery-powered mobile terminal, the components that compose Figure 4.1 *share* CPU, memory, and battery capacity. Consequently, the distribution of *shared resources* over the components must be added to the coordination process. in [7]. Controlling QoS with more than a single parameter is a complex problem. Solutions do exist in literature, but they usually result in *ad hoc* control structures [10, 9].

In [90] we have introduced a generic QoS negotiation method, called ARC. This method is able to handle complex applications, such as video streaming, and facilitates evolution. Here we apply the distributed and non-iterative ARC framework to the problem of mobile video streaming.

4.2 QoS negotiation: ARC

The ARC QoS negotiation method is illustrated in Figure 4.2. In conformance with the hierarchical setup shown in Figure 4.1, each module acts both as a server to 'higher' layers and as a client to 'lower' layers. The hierarchical concatenation of modules is important, because subsequent modules determine each others context. A QoS interface deals with the interaction of one module acting as server and one module acting as client. A contract is negotiated at the interface, holding a number of abstract QoS parameters.

The process of QoS negotiation starts with the client issuing a request. The request is a partial specification of the expectations the client has about the performance of the underlying server. The server (now approximately aware of what is expected) responds with an offer, stating possible performance options. This informs the client about the context dependent capabilities of the server. The client can respond to the offer, either by selecting an option and issuing it as a contract, or restarting the negotiation by formulating a modified request. Once the contract is established, the client can put the appropriate workload on the server. The server, in turn, must inform the client on the status of the workload processing. This feedback of context dependent QoS information to the client is essential for fast local adaptation. If the QoS status of the server becomes unsatisfactory for the client (due to changes in context), then the contract must be re-negotiated. This form of adaptation is slower, but involves more precise mutual tuning of QoS parameters, which improves efficiency.

As an example of QoS negotiation consider Figure 4.3. The abstract QoS is here represented in a two-dimensional space, the two parameters denoting capacity and quality. Contrary to how QoS is often used, an ARC interface re-

flects capacity (i.e. costs) in addition to quality parameters. The request from the client in Figure 4.3 is a range selection from the QoS space. Now the server knows what the client is interested in, and responds with a number of detailed offers. Each offer is in the initial request range, but gives a tighter description of the QoS that the server can offer within the current context. The client then marks a QoS range that includes at least one offer and sets it as the contract. Specifying a range rather than a single point for a contract leaves room for adaptation within the contract boundaries. When the contract is established, the server tries with best effort to keep the actual QoS within the contract range. The actual QoS status is returned to the client as a single point in the QoS domain. Using an abstract QoS domain as common language between the client and the server effectively hides explicit implementation details from the negotiations.

The QoS interface is the result of a collaborative design by the server and the client; they share a consistent interpretation of the QoS parameters. For reasons of efficiency, run-time implementations need not be that explicit. Minimization of power dissipation is a system-wide *implicit* agreement. Another example of an implicit agreement is ranking of parameters which improves the integrity of the system. In case of a contract violation a server can continue operation in a predetermined way. In Figure 4.3, for instance, the server will degrade quality, utilize more capacity, or do both.

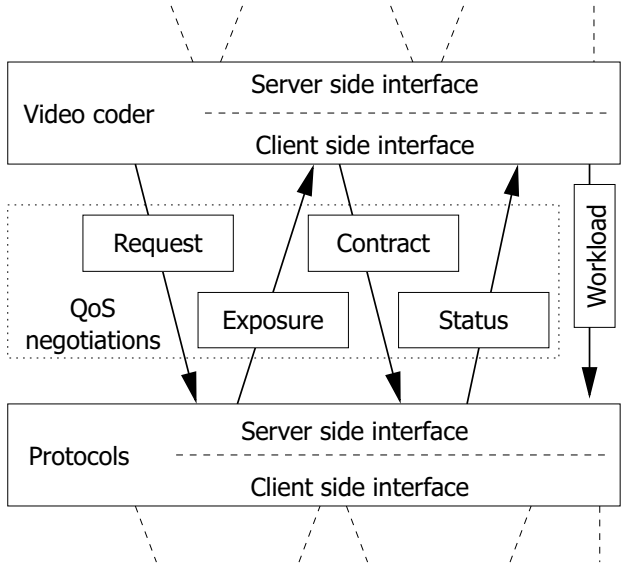


Figure 4.2 — *ARC protocol outline.*

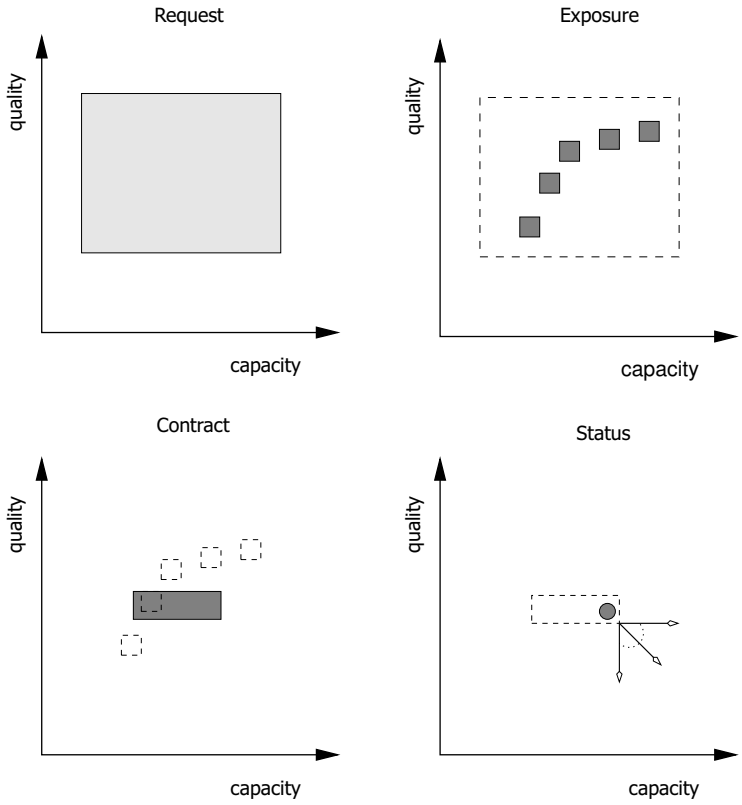


Figure 4.3 — *ARC operation spaces.*

4.3 Mobile Video Streaming Implementation

The implementation of the mobile video communication system involves a video encoder, a protocol component, and a radio transmission component. In our experiments we concentrate on the video encoder and protocol components, and assume without loss of generality that the applied radio component is non-adaptive. The applied channel model is time-varying though. Both the video encoder and protocol component support the ARC framework for doing QoS negotiations. Their fundamentals are described in this section.

The QoS parameters at the interface between the video encoder and protocols components are given in Table 4.1. For an detailed description of the design of this interface we refer to [87].

Table 4.1 — *QoS parameters (descending priority).*

Parameter	*Description*
latency (max)	The time required to transmit a single bit.
bit-error rate (max)	The net bit-error probability after FEC and ARQ.
CPU usage (max)	The allowed share of CPU capacity for protocols and transmission processes.
throughput (min)	The minimum net throughput.

4.3.1 Video Encoder

The video encoder implements a flexible H.263 encoder/decoder pair. Internally there are numerous video encoding parameters that can be tuned. In the context of ARC, we apply an abstract behavior description of the encoder, much like the work in [31, 47]. We used the entire model as presented in Section 3.2, which is not going to discussed here. Summarizing, we consider as dominant decision variables the frame-skip N_{fs}, the maximal motion vector length l_{me}, and the rate control's target bit-rate r. The variable user context is observed through the video sequence, namely through a model of the temporal predictability $\hat{\tau}$ and the (average) amount of variance σ_0^2.

The control model focuses on *a priori* estimation of the rate-distortion behavior as a function of l_{me}, N_{fs}, $\hat{\tau}$, and the choice whether or not to use motion compensation. The model expresses the prediction gain G as follows:

$$G(N_{\mathrm{fs}}, rmv) = \left(G_{\mathrm{me}} - (G_{\mathrm{me}} - G_0)e^{\frac{-l_{\mathrm{me}}}{\sigma_l}}\right) e^{\frac{-N_{\mathrm{fs}}}{N_{\mathrm{mc}}}} + 1 \tag{4.1}$$

where G_0, G_{me}, N_{mc}, and l_{me} are the model parameters of $\hat{\tau}$. G_0 and G_{me} are the prediction gains without and with motion compensation, respectively. N_{mc} is the motion coherence, describing the decay of prediction gain when the frame skip increases. σ_l is the average motion vector length, which characterizes the amount of motion in the sequence.

Given σ_X^2 we derive an estimate of the amount of variance to be encoded: $\sigma_d^2 = \sigma_X^2 / G(N_{\mathrm{fs}}, rmv)$. The distortion is estimated by a parameterized rate-distortion curve that describes the performance of the quantizer and the arithmetic coder. The estimated quantization noise is[1]

$$\sigma_q^2 = \sigma_d^2 \, 2^{2(b(e^{\frac{-r}{a}}-1)-r)} \tag{4.2}$$

The distortion after transmission and decoding is a PSNR value estimated from the quantization noise including effects of skipped frames and bit errors.

[1] This model is a less advanced RD-model than the one presented in Chapter 3.2, which was not available when this paper was written.

Skipped frames yield a PSNR based on the last received frame and $\hat{\tau}$. Bit errors degrade the PSNR assuming that one bit error destroys half of a Group of Blocks.

With our distortion model we evaluate different modes of operation. There is a trade-off between distortion and CPU utilization. The encoder offers the application a set of non-inferior points.

4.3.2 Protocols

The communication protocols run on top of a very simple radio that offers a TDMA scheme with fixed length transmission/receive slots to access the physical channel. Due to interference, fading, and other factors the data transmitted over the radio channel is subject to errors. The protocols employ two methods to counter the high bit-error rate (BER) of the physical channel: forward error correction (FEC) and automatic retransmit requests (ARQ).

Note that, contrary to many implementations, FEC is implemented in software. We use a Reed-Solomon protection scheme with four different code rates: 0%, 12.5%, 25%, and 50%. The code rate determines the maximum effective throughput that can be offered. We have run a number of off-line tests to determine the computational complexity and effective BER of the four code rates using white Gaussian noise. These results have been collected in a lookup table that is consulted during on-line execution.

For data that must be delivered reliably (i.e. without any error) packets are extended with a 32-bit checksum. When the receiver observes a checksum failure, it sends a retransmit request back to the sender, who will in turn resend the data. ARQ increases latency (L) and reduces the throughput (T) that can be obtained. We use the following model to quantify the efficiency loss due to retransmits:

$$T_{\text{ARQ}} = \frac{T_{\text{FEC}}}{1 + P_{\text{err}}} \tag{4.3}$$

$$L_{\text{ARQ}} = (1 + 2P_{\text{err}})L_{\text{FEC}} \tag{4.4}$$

where P_{err} is the probability that a packet is corrupted.

By combining the lookup tables for FEC and the ARQ model, the protocol layer can quickly evaluate what its best setting (code rate + enable/disable ARQ) is, given the current channel conditions and contract with the video encoder. When asked for an offer it prunes the inferior points out of the eight alternatives.

4.4 Experimental Evaluation

The experiment set up is as follows. A pre-recorded video stream (`carphone`) is encoded at a mobile terminal, transmitted over a (simulated) wireless link

and decoded at a base station. Latency requirements are such that *live* viewing is possible (< 0.5s). The radio has fixed settings: constant transmission power and constant modulation schemes during the length of the experiment. However interference at the physical channel will occur. Interference causes an increased bit error rate at the radio channel.

We present here the following three experiments. For all experiments the user requests the best quality possible within 100% CPU budget.

1. *Steady run*. There is constant interference on the mobile channel. Therefore neither of the components adapts during this experiment. Note the video encoder also does not adapt to changing characteristics of the incoming video source. We have run this experiment for a) a bad channel, $BER = 2\ 10^{-2}$ b) a medium channel, $BER = 10^{-2}$ and c) a good channel, $BER = 10^{-4}$. The raw channel bitrate is kept at $1.8\ 10^4$bit/s.

2. *Frozen run*. After 20 seconds from the start of the experiment the initial *medium* channel changes to a *bad* channel. The throughput is maintained at $1.8\ 10^4$bit/s. At $t = 47.5$s the channel changes to a *good* state. The protocols layer adapts instantaneously to the changed raw channel conditions. Initially it keeps the contracted BER at the link to the video encoder but it has to sacrifice throughput at the link. The video encoder establishes an initial contract assuming a (worst-case) BER of $2\ 10^{-2}$ right after the start of the experiments. For the remainder of the experiment, all internal settings (and contracts) are frozen. To maintain the agreed CPU budget and real-time objectives, some frames will be skipped, which decreases the delivered quality.

3. *Adaptive run*. The physical channel and protocols layer behave as in the frozen run above. This time, however, the video encoder initiates QoS negotiations and adapts to the changed conditions. Like in the frozen run the net effect is that the video encoder maintains the agreed real-time and CPU-budget constraints.

Figure 4.4 shows the results of the experiments. The top diagram shows the BER of the raw channel for the steady cases as well as the changing channel case. The diagram in the middle shows the effects on the throughput delivered by the protocols to the video encoder. The bottom diagram has four curves, three for the 'steady' runs and one for the adaptive run. The frozen run closely follows the 'bad' steady run, and is left out for clarity. The curves plot the quality (PSNR) per received frame. As can be expected the 'good channel' steady run has the highest quality. Quality variations over time are due to variations of the input source characteristics. Observe that the curve for the adaptive run switches between the three steady curves (medium→bad→good) when the channel conditions change. This shows that ARC-based negotiations succeed

in selecting appropriate settings of the video coder that outperform a coder that assumes worst-case conditions.

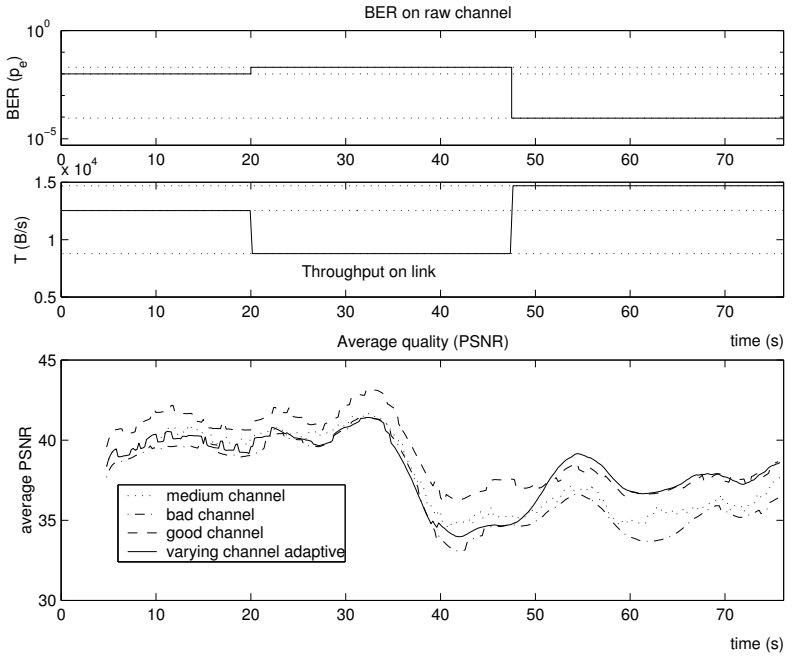


Figure 4.4 — *Experiment results.*

Compared to the *good* channel steady run the average quality of the frozen run is 1.97 dB less. The average quality of adaptive run is only 0.91 dB less than the steady run. In the adaptive run, 864 operation points were evaluated in three negotiations. The total time needed for these quality of service negotiations is 80ms. It is instructive to present the internal settings of the respective components for each of the experiments. Table 4.2 shows the results. The values shown for the frozen and adaptive run are averages, because the parameters are changing over time.

Until now we did not adapt to changes in source characteristics, but they vary drastically. An experiment in which the parameters were optimised taking into account the changing characteristics, improved the average quality with 1.1dB. This result was obtained using the *bad* channel and a CPU-budget of 30%. The user can trade quality for resources.

Table 4.2 — *Parameter settings.*

	steady				
Run	medium	bad	good	frozen	adaptive
Parameter	***Value***				
l_{me}	12.00	9.00	12.00	9.00	14.30
R (bpp)	0.67	0.47	0.79	0.47	0.51
N_{fs}	1.15	1.03	1.07	1.01	1.04
Video CPU-budget C_v	50%	45%	56%	52%	86%
FEC	25%	50%	0%	43%	38%
ARQ	0	0	0	0	0
Proto CPU-budget C_c	6%	14%	2%	8%	7%

4.5 Discussion

The experiments show that our ARC framework is able to improve the overall performance in cases where the channel conditions or video source characteristics fluctuate. Moreover the ARC framework makes it possible to keep the resource usage within bounds, even when the channel status is changing. It is able to make more efficient use of the available resources. The ARC-framework allows for a flexible implementation of modules. Therefore an ARC setup can operate in different environments: mobile or internet.

Stability problems may arise when the context changes occur too often, the optimiser might lag behind, and persists in making the wrong decisions. One way to avoid this problem, is to relax the contract margins. This allows the component to perform internal adaptations at the expense of being suboptimal.

Five

Error Resilient Video Compression using Behavior Models

5.1 Introduction

Wireless and Internet video applications are inherently subjected to bit errors and packet errors, respectively. This is especially so, when constraints on the end-to-end compression and transmission latency are imposed. Therefore, it is necessary to develop methods to optimize the video compression parameters and the rate allocation of these applications that take into account residual channel bit errors. Here we study the behavior of a predictive (inter-frame) video encoder, and model the encoders behavior using only the statistics of the original input data and of the underlying channel prone to bit errors. The resulting data-driven behavior models are then used to carry out Group of Pictures partitioning and to control the rate of the video encoder in such a way that the overall quality of the decoded video with compression and channel errors is optimized.

Although the current video compression techniques can be considered mature, there are still many challenges in the design and operational control of compression techniques for end-to-end quality optimization. This is in particular true in the context of unreliable transmission media such as the Internet and wireless links. Conventional compression techniques such as Joint Picture Experts Group (JPEG) and Motion Picture Experts Group (MPEG) were designed with error free transmission of the compressed bit stream in mind. With such unreliable media, not all bit or packet errors may be corrected by retransmis-

This chapter was published as: J. R. Taal, Z. Chen, Y. He, and R. L. Lagendijk, 'Error resilient video compression using behavior models,' *EURASIP Journal on Applied Signal Processing*, no. 2, pp. 290–303, Feb. 2004. [75]

sions or Forward Error Correction (FEC). Depending on the kind of channel coder, residual channel errors may be present in the bit stream after channel decoding. In most practical packet networks systems, packet retransmission corrects for some, but not all packet losses. Classic rate control, such as TM.5 in MPEG [51], can be used to control the video encoder according to the available bit rate offered by the channel coder, adaptation to the bit-error rate by inserting intra-coded blocks is nevertheless not incorporated in Test Model 5 rate control algorithm (TM.5). Other methods exist that control the insertion of intra-coded blocks [21].

Three classes of error-resilient source coding techniques may be distinguished that deal with error prone transmission channels. The first well-known approach is joint source-channel coding, which aims at intimate integration of the source and channel coding algorithms [23, 14]. Although this intimate integration brings several advantages to the end-to-end quality optimization, it comes at the price of a significant complexity increase. Furthermore, nearly all of these approaches only work with specific or non-standard network protocols and with a specific video encoder and/or decoder.

The second class represents many approaches where the source coder has no (or limited) control of the network layer. It is important to understand that these approaches can generally not be optimal, since the channel coder and the source coder are not jointly optimized. Since there is no joint optimization, the only thing the source coder can do is to adapt its own settings according the current behavior of the network layer. In many applications joint optimization is impossible because none of the standard network protocols (IP, TCP, UDP) support this. Even though the source coder has no or limited control over the network layer, the rate control algorithm can adapt to the available bit rate and to the amount of residual bit errors or packet losses. Such a control algorithm needs a model describing the effects of bit errors or packet losses on the overall distortion.

The third class, contains the approaches advocated in [71, 86]. In these approaches the best of both worlds are combined. In these approaches, the authors propose to limit the integration to joint parameter optimization, so that there is no algorithmic integration. In previous work at Delft University of Technology [90] an efficient overall framework was proposed for such joint parameter optimization from a Quality-of-Service (QoS) perspective. This framework requires high-level and abstract models describing the behavior of source and channel coding modules. However, this framework had not yet been tested with a real video coder and with a real behavior model.

We propose such a behavior model for describing source-coding characteristics, given some information about the channel coder. Although this model is designed to be used in a QoS setup, it may also be used to optimize the encoders settings when we only have knowledge of but no control over the current channel (as a second class approach).

With this behavior model we can predict the behavior of a source coder in terms of the image quality related to the channel coder parameters: the bit rate, the bit-error rate (BER) and the latency. To be applicable in a real-time and perhaps low power setup, the model itself should have a low complexity, and should not require that many frames have to reside in a buffer (low-latency).

We evaluate the behavior models with one type of progressive video coder. However, we believe that other coders can be described fairly easily with our methods as well, since we try to describe the encoders at the level of behavior rather than at a detailed algorithmic or implementation level. In Section 5.2, we first discuss our combined source channel coding system, the problem we wish to solve, and we describe the source and channel coders on a fairly high abstraction level. From these models, we can formulate end-to-end quality control as an optimization problem, which we discuss in Section 5.3. Section 5.4 describes in depth the construction of the proposed models. In Section 5.5 our models are validated in a simulation where a whole Group of Pictures (group of pictures (GOP)) was transmitted over an error prone channel. Section 5.6 concludes this chapter with a discussion.

5.2 Problem Formulation

To optimize the end-to-end quality of compressed video transmission, one needs to understand the individual components of the link. This understanding involves knowledge of the rate distortion performance and the error resilience of the video codec, of the error correcting capabilities of the channel codec, and possibly of parameters such as delay, jitter, and power consumption. One of the main challenges in attaining an optimized overall end-to-end quality is the determining of the influence of the individual parameters controlling the various components. Especially because the performances of various components depend on each other, the control of these parameters is not straightforward.

In [71, 16, 14, 30], extensive analyses of the interaction and trade-offs between source and channel coding parameters can be found. A trend in these approaches is that the underlying components are modeled at a fairly high abstraction level. The models are certainly independent of the actual hardware or software implementation but they also become more and more independent of the actual compression or source-coding algorithm used. This is in strong contrast to the abundance of joint source-channel coding approaches, which typically optimize a particular combination of a source and channel coder, utilizing specific internal algorithmic structures and parameter dependencies. Although these approaches have the potential to lead to the best performance, their advantages are inherently limited to the particular combination of coders and to the (source and channel) conditions under which the optimization was carried out.

Here, we refrain from full integration of source and channel codecs (i.e. the joint source-channel coding approach), but keep the source coder and channel coder as much separate as possible.

The interaction between source and channel coder, and in particular the communication of key parameters, is encapsulated in a QoS framework. The objective of the QoS framework is to structure the communication context parameters between OSI layers. In the scope of this research, the changing context can be the radio/Internet channel conditions and the application demands on the quality or the complexity of the video data to be encoded. Here we discuss only the main outline of the QoS interface. A more detailed description of the interface can be found in the literature [86, 90].

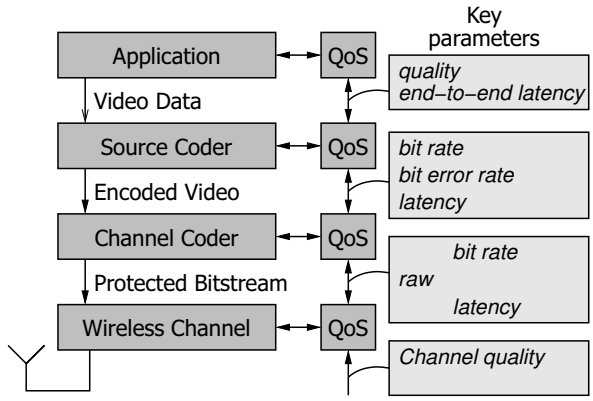


Figure 5.1 — *QoS ARC concept: Different (OSI)layers are not only communicating their payloads; they are also controlled by QoS controllers that mutually negotiate to optimize the overall performance.*

Figure 5.1 illustrates the QoS Interface concept [90]. The source and channel coders operate independently of each other, but are both under the control of QoS controllers. The source coder encodes the video data, thereby reducing the needed bit rate. The channel coder protects this data. It decreases the BER, thereby effectively reducing the bit rate available for source coding and increasing the latency. The QoS controller of the source coder communicates the key parameters – in this case the bit rate, BER, and latency – with the QoS controller of the channel coder. Based on the behavior description of the source and channel coding modules, the values of these parameters are optimized by the QoS controller. In a practical system this optimization takes into account context information about the application (e.g. maximum latency) and about the channel (e.g. throughput at the physical layer). The application may set constraints on the operation of the lower layers, for instance on the power

consumption or the delay. In this research we assumed that the only constraint set by the application is the end-to-end delay T_a.

In order to implement the QoS Interface/controller concept, three problems need to be solved:

- The key parameters must be optimized over different (OSI) layers. We have developed the 'Adaptive Resource Contracts' (ARC) approach for solving this problem. ARC exchanges the key parameters between two layers, such that after a negotiation phase, both layers agree on the values of these parameters. These key parameters represent the trade-offs that both layers have made to come to a joint solution of the optimization. A detailed discussion of ARC falls outside the scope. We refer to [86, 90, 73].

- The behavior of the source and channel coder should be modeled parametrically such that joint optimization of the key parameters can take place. At the same time, an 'internal' controller should be available that optimizes the performance of the source and channel coder independently, given the already jointly optimized key parameters

  The emphasis in this chapter is on the modeling of the video coder behavior.

- An optimization procedure should be designed for selecting the parameters internal to the video codec, given the behavior model and the key parameters. We do not emphasize this aspect of the QoS Interface here, as we believe that the required optimization procedure can be based on related work as that in [60].

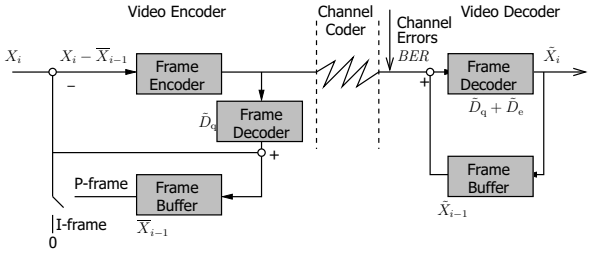


Figure 5.2 — *Simple video coding scheme. The 'Frame Encoder' and 'Frame Decoder' blocks represent the single frame encoder and decoder. The 'Frame Buffer' is needed for the predictively encoded P-frames.*

In our previous work and analyses [86, 90], the source coder was modeled as a *progressive encoder*, which means that with every additionally transmitted bit the quality of the received decoded information increases. Therefore the

most important information is encoded at the beginning of the data stream and the least important information is encoded at the end. In principle, we believe that any progressive encoder can be described with our models. To keep things simple from a compression point of view, we use the common P-frame coding structure (with one I-frame per GOP, multiple predictively encoded P-frames, and no bidirectional encoded frames). The actual encoding of the (difference) frames is done by a JPEG2000 [11] encoder, which suits our demand for progressive behavior. Figure 5.2 shows the typical block diagram of this JPEG2000-based P-frame coder. We excluded motion compensation of P-frames for simplicity reasons. The 'internal parameters' for this video encoder are the number of frames in a GOP N, and the bit rates r_i for the individual frames. X_i and X_{i-1} denote the current frame and the previous frame, $\overline{X}$ denotes a decoded frame, and $\tilde{X}$ denotes a decoded frame at the decoder side, possibly with distortions caused by residual channel errors. $\tilde{D}_{\mathrm{q}}$ and $\tilde{D}_{\mathrm{e}}$, denote the quantization distortion and the distortions caused by residual channel errors (named 'channel-induced distortion' hereafter), respectively.

In this work, the channel coder is defined as an abstract functional module with three interface parameters. The channel coder has knowledge of the current state of the channel which it is operating on. Therefore it can optimize its own internal settings using behavior models. Such a channel coder may use different techniques like FEC and automatic resent query (ARQ) to protect the data at the expense of added bit rate and increased delay (latency). The exact implementation is nevertheless irrelevant at this level. From here we will assume that the error protection is not perfect because of latency constraints; therefore the residual bit-error rate (BER) may be non-zero. The behavior models can be obtained by straightforward analysis of the channel coding process [71].

5.3 Source Encoder Optimization Criterion

At this point we assume that we have a behavior model for our video encoder. The development of this behavior model is the subject of Section 5.4. Given the behavior model, we can minimize the average end-to-end distortion $\hat{D}$ given the constraints imposed by the QoS Interface. In our work, the QoS Interface negotiates three key parameters between source and channel coder, namely $\{R, BER, T_c\}$, with

- R: the available bit rate for source coding (average number of bits per pixel);
- the residual bit-error rate BER: the average bit-error rate after channel decoding;
- T_c: the average time between handing a bit to the channel encoder, and receiving the same bit from the channel decoder.

The resulting source coding optimization problem now becomes the minimization of the distortion D, which can be formulated as follows:

$$\min_{I_{\mathrm{src}}} D\left(I_{\mathrm{src}} \mid \{R, BER, T_c\}\right). \tag{5.1}$$

Here I_{src} denotes the set of internal source coder parameters over which the performance of the encoder must be optimized, given the key parameters $\{R, BER, T_c\}$. The actual set of internal parameters to be considered depends on the encoder under consideration and the parameters included in the encoders behavior model. We consider the optimization of the following internal parameters:

- N, the length of the current GOP. Each GOP starts with an I-frame and is followed by $N-1$ predictively encoded P-frames;
- $\vec{r} = \{r_0, r_1, \ldots r_{N-1}\}$: the target bit rate for each individual frame in a GOP.

The encoder parameter N relates to the coding efficiency and the robustness of the compressed bit stream against remaining errors. The larger N, the higher the coding efficiency because more P-frames are encoded. At the same time the robustness of the stream is lower due to the propagation of decoded transmission errors.

On the other hand, in order to optimize the settings $\{N, \vec{r}\}$ for $N_{\max}$ frames, these $N_{\max}$ frames have to be buffered, thereby introducing a latency. In our approach the QoS Interface prescribes the maximum end-to-end latency T_a (sec) and we assume the channel coder will have an end-to-end latency of T_c (sec), from channel encoder to channel decoder, including transmission. Analysis of the whole transmission chain gives the following expression for the total end-to-end latency:

$$T_a = \frac{N-1}{f_r} + T_e + T_c + \frac{B}{R}, \tag{5.2}$$

where f_r is the frame rate of the video sequence that is encoded, T_e is the upper bound of the time it takes to encode a frame. Finally B/R is the transmission time for one frame B/R: the maximal number of bits to describe a frame, divided by the channel coding bit rate R.

We can now find an expression for the maximal number of frames that can be in the buffer, while still meeting the end-to-end latency constraints T_a. Clearly B/R is only known after allocating the rate for each frame. We suggest taking the worst-case value for B (i.e. calculated from the maximal bit rate setting). The same goes for T_E, where we suggest to take the worst-case encoding time per frame.

$$N_{\max} = 1 + (T_a - T_e - T_c - \frac{B}{R}) f_r, \tag{5.3}$$

In each frame i, two kinds of distortion are introduced: (1) the quantization error distortion, denoted by D_{q} and (2) the channel-induced distortion, caused by bit errors in the received bit stream, denoted by D_{e}. With our optimization problem we aim to minimize the average distortion, which is the sum of individual distortions of a GOP divided by the length of the group:

$$D_{\mathrm{GOP}} = \frac{1}{N} \sum_{i=0}^{N-1} \left\{ D_{\mathrm{q}}(r_i) + D_{\mathrm{e}}(r_i, BER) \right\}. \tag{5.4}$$

Following [71], we assume that D_{q} and D_{e} within one frame are mutually independent. Although (5.4) is a simple additive distortion model, the distortion of a frame is still dependent on that of the previous frames because of the P-frame prediction. Therefore, in our models we have to take into account the propagation of quantization and channel induced distortions.

Taking the above parameters into account, we can now rewrite (5.1) as the following bit-rate allocation problem:

$$\begin{aligned} \vec{r}_{\mathrm{opt}}, N_{\mathrm{opt}} &\leftarrow \min_{\vec{r},N} D_{\mathrm{GOP}}(\vec{r}, N|BER) \\ &= \min_{N} \left\{ \min_{\vec{r}} \frac{1}{N} \sum_{i=0}^{N-1} D_{\mathrm{q}}(r_i) + D_{\mathrm{e}}(r_i, BER) \right\} \\ \text{subject to } & \frac{1}{N} \sum_{i=1}^{N-1} r_i = R \\ & N \leq N_{\max}. \end{aligned} \tag{5.5}$$

The approach that we follow here is to optimize the bit-rate allocation problem (5.5) based on two frame-level parametric behavior models. The first (rate distortion) model parametrically describes the relation between the variance of the quantization distortion and the allocated bit rate based on the variance of the input frames. The second – channel induced distortion – model parametrically describes the relation between the variance of the degradations due to transmission and the decoding errors, based on the variance of the input frames and the *effective* bit error rate.[1]

5.4 Rate Distortion Model

In this section we first propose a behavior model for the rate-distortion characteristics D_{q} of video encoders, and then propose a model for distortion caused by residual channel errors including the error propagation, D_{e}.

[1] By 'effective bit-error rate' we mean the residual bit error rate, i.e. , the bit errors that are still present in the bit stream after channel decoding.

There are two approaches for modeling the rate distortion (R-D) behavior of sources. The first approach is the analytical approach, where mathematical relations are derived for the R-D functions assuming certain (stochastic) properties of the source signal and the coding system. Since these assumptions often do not hold in practice, the mismatch between the predicted rate distortion and the actual rate distortion is (heuristically) compensated for by empirical estimation. The second is the empirical approach, where the R-D functions are modeled through regression analysis of the empirically obtained R-D data. The rate distortion model proposed in [71] is an example of an empirical model of the distortion of an entire encoder for a given bit rate.

In our work we anticipate the real-time usage of the constructed abstract behavior models. At the same time the want to keep the complexity of the models low. This limits the amount of 'preprocessing' or 'analysis' that we may do on the frames to be encoded. Therefore we will base our behavior models on variance information only. In particular we will use:

- the variance of the frame under consideration, denoted by $\mathrm{var}\,[X_i]$, and
- the variance of the difference of two consecutive frames, denoted by $\mathrm{var}\,[X_i - X_{i-1}]$.

5.4.1 Rate Distortion Behavior Model of Intra-frames

It is well known that for memoryless Gaussian distributed sources X with variance $\mathrm{var}\,[X]$, the R-D function is given by:

$$r(D_{\mathrm{q}}) = \frac{1}{2}\log_2\left(\frac{\mathrm{var}\,[X]}{D_{\mathrm{q}}}\right), \tag{5.6}$$

or when we invert this function, by

$$D_{\mathrm{q}}(r) = \mathrm{var}\,[X]\,2^{-2r}. \tag{5.7}$$

Empirical observations show that for the most common audio and video signals under small distortions, the power function $-2r$ gives an accurate model for the behavior of a compression system, especially in terms of the quality gain per additional bit (in bit rate terms) spent. For instance, the power function $-2r$ leads to the well-known result that, at a sufficiently high bit rate, for most video compression systems we gain approximately 6 dB per additional bit per sample.

However, for more complicated compression systems and especially for larger distortions, the simple power function does not give us enough flexibility to describe the empirically observed R-D curves, which usually give more gain for the same increase in bit rate. Since there is basically no theory to rely

on for these cases without extremely detailed modeling of the compression algorithm, we instead propose to generalize (5.7) as follows

$$D_{\mathrm{q}}(r) = \mathrm{var}\,[X]\, 2^{f(r)}. \tag{5.8}$$

The function $f(r)$ gives us more freedom to model the rate distortion behavior, at the price of regression analysis or on-line parameter estimation on the basis of observed rate distortion realizations. The choice of the kind of the function used to model $f(r)$ is a pragmatic one. We have chosen a third-order polynomial function. A first- or second-order function was simply too imprecise, while a fourth-order model did not give a significant improvement and higher-order models would defeat our objective of finding simple and generic models. Clearly there is a trade-off between precision (high order) and generality (low order).

In Figure 5.3 we show the Rate Distortion curve of the experimentally obtained $D_{\mathrm{q}}(r)$ for the JPEG2000 compression of the first frame of the `carphone` sequence for bit rates between 0.05 and 1.1 (bpp). The solid line represents a third-order polynomial fit of $f(r)$ on the measured values. This fit is much better than the linear function $f(r) = -2r$. The following function was obtained for the first frame of the `carphone` sequence

$$D_{\mathrm{q}}(r) = \mathrm{var}\,[X]\, 2^{-4.46r^3+11.5r^2-12.7r-1.83}. \tag{5.9}$$

It is interesting to see how the R-D curve changes for different frames of the same scene or different scenes. Figure 5.4 shows the R-D curve for frame 1 and frame 60 of `carphone` and frame 1 of `foreman`. Observe that the `carphone` frames have very similar curves. The `foreman` curve is shifted, but is still similar to the other two. These observations strengthen our belief that the model is generally applicable for this type of coder. Of course the $f(r)$ needs to be fitted for a particular sequence, on the other hand, we believe that a default curve $f_0(r)$ can be used to bootstrap the estimation of model parameters for other video sequences. $f(r)$ can then be adapted with new R-D data as the encoding continues.

5.4.2 Rate Distortion Behavior Model of Inter-frames

For modeling the rate distortion behavior of P-frames, we propose to use a model similar as the one in (5.8), but with a different polynomial $g(r)$:

$$D_{\mathrm{q}}(r_i) = \mathrm{var}\left[X_i - \overline{X}_{i-1}\right] 2^{g(r_i)}. \tag{5.10}$$

Here $\overline{X}_{i-1}$ denotes the previously decoded frame $i-1$. Whereas with I-frames, a third-order polynomial was needed to predict $f(r)$ accurately enough, with P-frames, a second-order polynomial was sufficient to predict $g(r)$. The reason

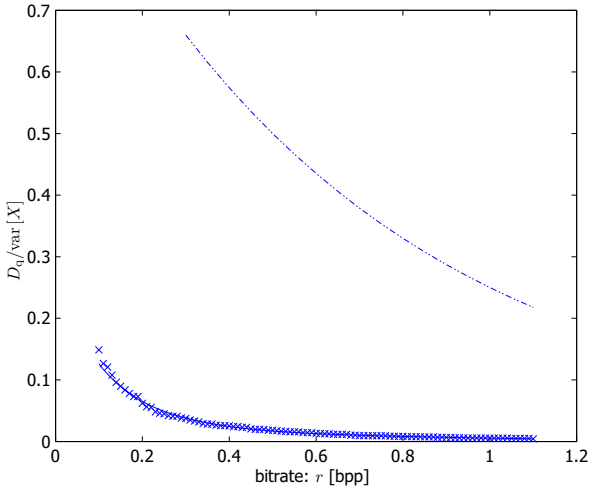


Figure 5.3 — *R-D curve for first frame in* `carphone`*. The crosses(X) are the measured normalized distortions* $\tilde{D}_q$ *and the line corresponds to the fitted function* $2^{f(r)}$*. The dotted line corresponds to the R-D model* 2^{-2r}

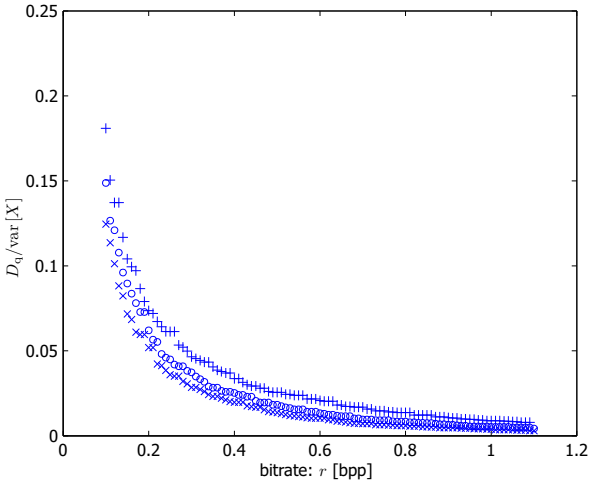


Figure 5.4 — *I-frame R-D curve for first frame of* `carphone` *(X), frame 60 of* `carphone` *(O) and the first frame of* `foreman` *(+)*

for this can be found in the fact that P-frames are less correlated than I-frames. Therefore $g(r)$ is more similar to the theoretical $-2r$ than $f(r)$.

In (5.10), $\mathrm{var}\left[X_i - \overline{X}_{i-1}\right]$ is the variance of the difference between the current frame i and the previously *encoded* frame $i-1$. Since the latter is only available *after* encoding (and thus after solving (5.5)), we need to approximate $\mathrm{var}\left[X_i - \overline{X}_{i-1}\right]$. Obviously we have:

$$\begin{aligned}\mathrm{var}\left[X_i - \overline{X}_{i-1}\right] &= E[(X_i - \overline{X}_{i-1})^2] \\ &= E[((X_i - X_{i-1}) - (\overline{X}_{i-1} - X_{i-1}))^2] \\ &= \mathrm{var}\left[X_i - X_{i-1}\right] + D_{\mathrm{q}}(r_{i-1}) - \\ &\quad 2E[(X_i - X_{i-1})(\overline{X}_{i-1} - X_{i-1})].\end{aligned} \tag{5.11}$$

The last term on the right-hand side of (5.11) cannot be easily estimated beforehand, and should therefore be approximated. We collapse this entire term into a quantity that only depends on the amount of quantization errors D_{q} from the previous frame, yielding:

$$\mathrm{var}\left[X_i - \overline{X}_{i-1}\right] = \mathrm{var}\left[X_i - X_{i-1}\right] + \kappa D_{\mathrm{q}}(r_{i-1}). \tag{5.12}$$

We expect the quantization noise of frame X_{i-1} to be only slightly correlated with the frame difference between frames X_{i-1} and X_i. Therefore we expect the value of κ to be somewhat smaller than one. Note that by combining (5.12) and (5.10), D_{q} is defined recursively, thereby making (5.5) a dependent optimization problem.

Figure 5.5 illustrates the relation between the frame difference variance $\mathrm{var}\left[X_1 - \overline{X}_0\right]$ and the quantization distortion of the first frame of `carphone` $\tilde{D_{\mathrm{q}}}$. The first frame is encoded at different bit rates. We observe a roughly linear relation, in this case with an approximate value of $\kappa = 0.86$.

We observed similar behavior for other sequences such as `susie` and `foreman` as well. We therefore postulate that (5.12) is an acceptable model for calculating the variance $\mathrm{var}\left[X_i - \overline{X}_{i-1}\right]$, as needed in (5.10).

The variance $X_i - \overline{X}_{i-1}$ in fact consists of two terms: the quantization distortion of the previous frames, and the frame difference between the current and the previous frame. These two terms might show different R-D behavior, i.e. , a separate $g(r)$ for both terms. However, we assume that both signals show the same behavior, since they are both frame difference signals by nature and not whole frames. The model for predicting the distortion of an P-frame now becomes:

$$D_{\mathrm{q}}(r_i) = \left(\mathrm{var}\left[X_i - X_{i-1}\right] + \kappa D_{\mathrm{q}}(r_{i-1})\right) 2^{g(r_i)}. \tag{5.13}$$

Figure 5.6 shows the experimentally obtained R-D curve, together with a fitted curve representing our model (5.13). Since this R-D curve should not only

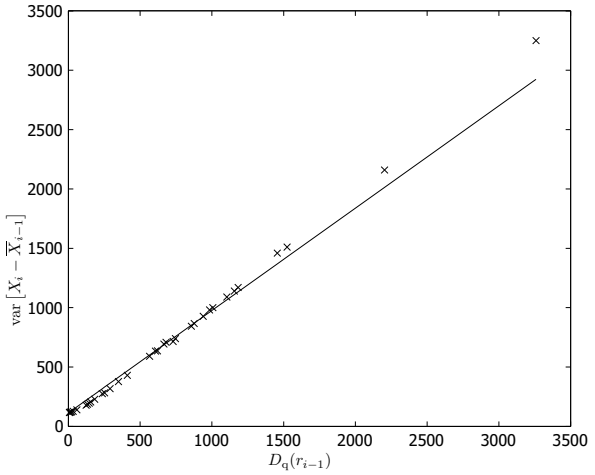


Figure 5.5 — *Relationship between the variance of frame difference* $\mathrm{var}\left[X_i - \overline{X}_{i-1}\right]$ *and the quantization distortion* $\tilde{D}_{\mathrm{q}}(r_{i-1})$*. The fitted line describes* $\mathrm{var}\left[X_i - \overline{X}_{i-1}\right] = \mathrm{var}\left[X_i - X_{i-1}\right] + \kappa D_{\mathrm{q}}(r_{i-1})$

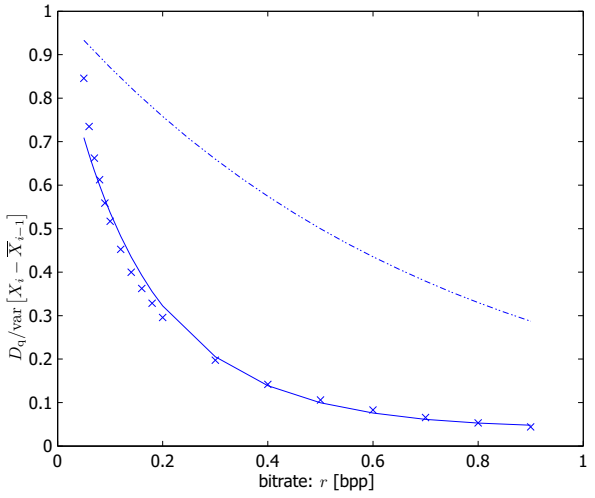


Figure 5.6 — *Average R-D curve for first P-frame of* `carphone`*. The crosses (X) are the measured normalized distortions* $\tilde{D}_{\mathrm{q}}(r_i)$ *and the line corresponds to the fitted function* $2^{g(r)}$*. The dotted line corresponds the R-D model* 2^{-2r}

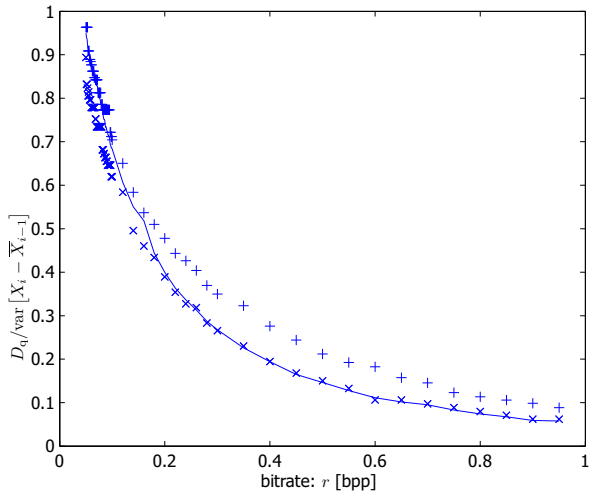


Figure 5.7 — *R-D curve for the first P-frame of* `carphone` *(x), the average R-D curve for the first ten frames of* `carphone` *(line), and the R-D curve for the first frame of* `foreman` *(+)*

be valid for varying bit rate r_i, but also for varying propagated quantization distortion $D_{\mathrm{q}}(r_{i-1})$, we also varied the bit rate of the previous frame r_{i-1}. Both rates were varied from 0.05 to 0.9. Each value of $D_q(r_i)$ is an average over all settings of r_{i-1}. For completeness the theoretic curve (5.7) is shown as well. The function that describes the R-D behavior for these frames is:

$$D_{\mathrm{q}}(r_i) = \left(\mathrm{var}\left[X_i - X_{i-1}\right] + \kappa D_{\mathrm{q}}(r_{i-1})\right) 2^{3.86r_i^2 - 8.15r_i - 0.26}. \qquad (5.14)$$

We then compared the curves for different frames. Figure 5.7 shows the R-D curve for the first frame difference of `carphone`, and the R-D curve for the first frame difference of `foreman`, as well as the average R-D curve for the first 10 frame differences of `carphone`. This shows again that these curves do not vary much for different video frames and different video sources.

5.4.3 Channel Induced Distortion Behavior Model

When the channel suffers from high error rates, the channel decoding will not be able to correct all bit errors. Therefore, to solve (5.5), we also need a model that describes the behavior of the video decoder in the presence of bit-stream errors.

First we define the channel-induced distortion to be the variance of the difference between the decoded frame ($\overline{X}$) at the encoder side and the decoded

frame at the decoder side ($\tilde{X}$):

$$\tilde{D}_{\mathrm{e}} = \mathrm{var}\left[\tilde{X} - \overline{X}\right]. \tag{5.15}$$

In [30] a model is proposed that describes the coders vulnerability to packet error losses:

$$D_{\mathrm{e}} = \sigma_{u_0}^2 \mathrm{PER}, \tag{5.16}$$

where $\sigma_{u_0}^2$ is an empirical constant and is found empirically and PER is the packet-error rate. Since we are dealing with bit errors and want to predict the impairment on a frame-per-frame basis, we are looking for a better model.

Modeling the impairments that are due to uncorrected bit errors may result in a detailed analysis of the compression technique used, see for instance [61]. Since we desire to have an abstract and high-level model with a limited number of parameters, we base our model on three empirical observations, namely:

1. For both I-frames and P-frames, the degree of image impairment due to uncorrected errors depends on the BER. If the individual image impairments caused by channel errors are independent, then the overall effect is the summation of individual impairments. At higher error rates, where separate errors cannot be considered independent anymore, we observed a decreasing influence of the BER.

   We notice that in a bit stream, a sequence of L bits will be decoded erroneously if one of the bits is incorrect due to a channel error. The probability of any bit being decoded erroneously is then

   $$P_{\mathrm{E}}(BER, L) = 1 - (1 - BER)^L. \tag{5.17}$$

   Note that this model describes the behavior related to dependencies between consecutive bits in the bit stream and does not assume any packetization. The value of L is therefore found by curve fitting and not by an analysis of the data stream structure. Clearly, the value of L will be influenced by the implementation specifics, such as re-sync markers. We interpret L as a value for the effective packet length, i.e. , the amount of data lost after a single bit error as if an entire data packet of length L is lost due to an uncorrected error. This model for P_{E} corresponds very well with the observed channel-induced distortion behavior, so we postulate:

   $$D_{\mathrm{e}} \sim P_{\mathrm{E}} = (1 - (1 - BER)^L), \tag{5.18}$$

   where parameter L was typically found to be in the order of 200 for I-frames and of 1000 for P-frames.

2. For I-frames, the degree of image impairment due to uncorrected errors highly depends on the amount of variance of the original signal, but also on the amount of quantization distortion.

$\mathrm{var}\,[X_i] - D_\mathrm{q}(r_i)$ represents in fact the amount of variance that is encoded: the higher the distortion $D_\mathrm{q}(r_i)$, the less information is encoded. We observed that if $D_\mathrm{q}(r_i)$ increases, the effect of residual channel errors decreases. Clearly at $r_i = 0$, nothing is encoded in this frame and the distortion equals the variance. At $r_i \gg 0$, $D_\mathrm{q} \approx 0$, there is no quantization distortion, all information is encoded and will be susceptible to bit errors. We therefore postulate

$$D_\mathrm{e}(r_i, BER) \sim \mathrm{var}\,[X_i] - D_\mathrm{q}(r_i). \tag{5.19}$$

3. For P-frames we did not observe a statistically significant correlation between the quantization distortion (i.e. the bit rate) and the image impairment due to channel errors. We assume that the image impairment is only related to the variance of the frame difference, thus here we do not take into account the quantization distortion.

$$D_\mathrm{e}(r_i, BER) \sim \mathrm{var}\,[X_i - X_{i-1}]\,. \tag{5.20}$$

These empirical observations lead us to postulate the following aggregated model of the channel-induced distortions for an I-frame:

$$D_\mathrm{e}(r_i, BER) = \mathrm{var}\left[\tilde{X}_i - \overline{X}_i\right] = \alpha P_\mathrm{E}(BER, L_\mathrm{I})\left(\mathrm{var}\,[X_i] - D_\mathrm{q}(r_i)\right), \tag{5.21}$$

and for one P-frame:

$$D_\mathrm{e}(r_i, BER) = \beta P_\mathrm{E}(BER, L_\mathrm{P})\mathrm{var}\,[X_i - X_{i-1}]\,. \tag{5.22}$$

Here $P_\mathrm{E}(BER, L)$ is given by (5.17). L_I and L_P are the effective packet lengths for I-frames and P-frames, respectively. The constants α and β determine to which extent an introduced bit error distorts the picture, and need to be found empirically.

For I-frames, $D_\mathrm{e}(r_i, BER)$ depends on BER and on the variance $\mathrm{var}\,[X_i] - D_\mathrm{q}(r_i)$. Two figures show the curve fitting on this two-dimensional function visible. Both figures show the results of encoding one frame at different bit rates (ranging from 0.05 to 2.0 bpp) and at different BERs (ranging from 10^{-3} to 10^{-6}), where bit errors were injected in the encoded bit stream randomly. Since we wish to predict the average behavior, we calculated the average distortions of 1000 runs for each setting.

1. Figure 5.8 shows the average $\tilde{D}_\mathrm{e}$ divided by $\mathrm{var}\,[X_i] - \tilde{D}_\mathrm{q}$ as a function of BER. The straight line corresponds to a line fitted with $P_\mathrm{E} = BER$ and $\alpha = 255.2$. We observe that it deviates at higher BER.

   The fitted curve corresponds to $P_\mathrm{E} = 1 - (1 - BER)^{L_\mathrm{I}}$ with an effective packet length $L_\mathrm{I} = 202$ and $\alpha = 1.29$ gives a better fit.

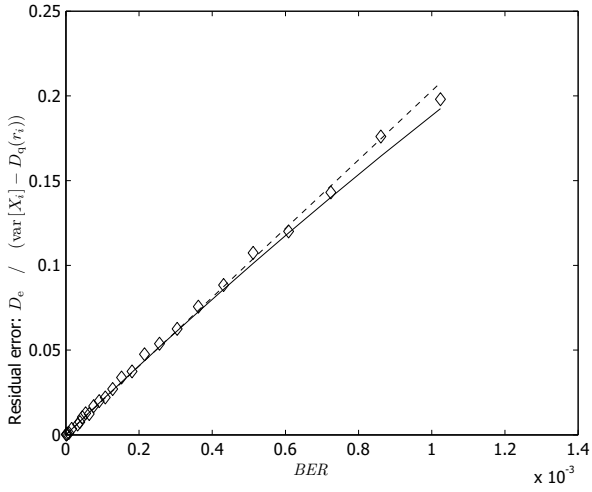


Figure 5.8 — *Plot of BER versus the normalized distortion $\frac{\tilde{D}_{\mathrm{e}}(r_i)}{\mathrm{var}[X_i]-D_{\mathrm{q}}(r_i)}$ for the first I-frame of* `carphone`*. The dashed line corresponds to the simple model $P_{\mathrm{E}} = BER, \alpha = 255.2$, the solid line to the model $P_{\mathrm{E}} = 1-(1-BER)^{202}$, $\alpha = 1.29$*

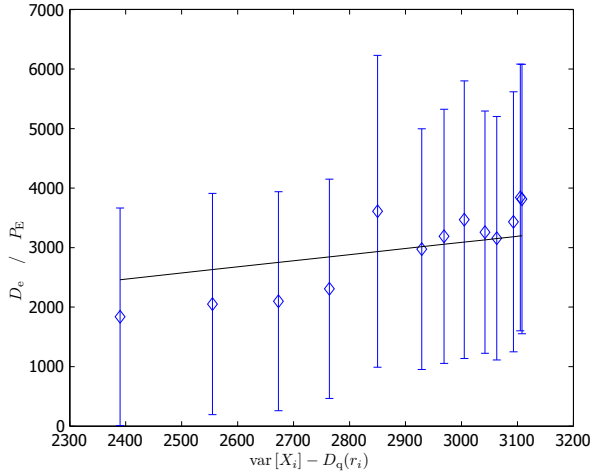


Figure 5.9 — *Plot of $\mathrm{var}\,[X_i] - \tilde{D}_{\mathrm{q}}(r_i)$ versus the normalized distortion $\frac{\tilde{D}_{\mathrm{e}}(r_i,BER)}{P_{\mathrm{E}}(BER,L_{\mathrm{I}})}$, for the first I-frame of* `carphone`*. The error bars represent the standard deviation over thousand runs of the experiment. The line is our model $\frac{D_{\mathrm{e}}(r_i,BER)}{P_{\mathrm{E}}(BER,L_{\mathrm{I}})} = \alpha(\mathrm{var}\,[X_i] - D_{\mathrm{q}}(r_i))$*

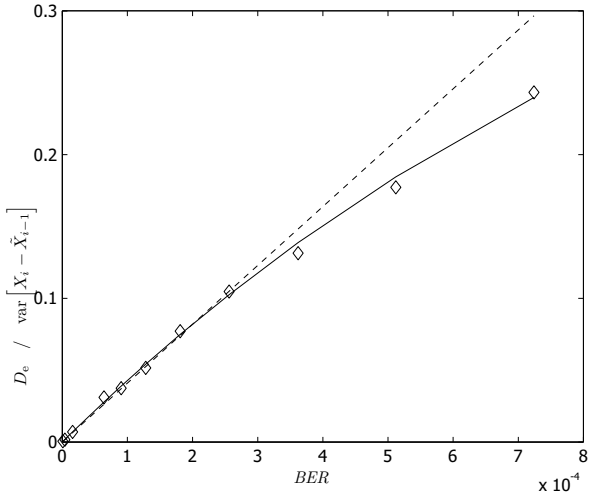


Figure 5.10 — *Plot of BER vs. the normalized channel-induced distortion $\frac{\widetilde{D_{\mathrm{e}}}(r_i,BER)}{\mathrm{var}[X_i-X_{i-1}]}$. The values are averages for the first ten P-frames of* `carphone`*. The dashed line corresponds to the model $P_{\mathrm{E}} = BER$, the solid line corresponds to the model $P_{\mathrm{E}} = 1-(1-BER)^{876}, \alpha = 0.51$*

2. Figure 5.9 shows $\tilde{D}_{\mathrm{e}}$ divided by $P_{\mathrm{E}}(BER, L_{\mathrm{I}} = 202)$ as a function of $\mathrm{var}\,[X_i] - \tilde{D}_{\mathrm{q}}$. The fitted line crosses the origin. Clearly this model does not fit these measurements extremely well, because the effect of $D_{\mathrm{q}}(r_i)$ is very unpredictable. Still we want to incorporate the trend in the effect that $D_{\mathrm{q}}(r_i)$ has on the error distortion. For other sources (`foreman`, `susie`) we observed a similar behavior.

Finally for P-frames, $D_{\mathrm{e}}(r_i, BER)$ only depends on BER and on the constant factor $\mathrm{var}\,[X_i - X_{i-1}]$. Figure 5.10 shows the average $\tilde{D}_{\mathrm{e}}$ divided by $\mathrm{var}\,[X_i - X_{i-1}]$ versus the BER. The resulting curve corresponds to $P_{\mathrm{E}} = 1-(1-BER)^{L_{\mathrm{P}}}$, with $L_{\mathrm{P}} = 876$. Here we found $\beta = 0.51$.

5.4.3.1 Error Propagation in Inter-frames

Due to the recursive structure of the P-frame coder, decoding errors introduced in a frame will cause temporal error propagation [46, 30]. Since (5.5) tries to minimize the distortion over a whole GOP, we have to take this propagation into account for each frame individually. In [30], a high-level model was proposed to describe the error propagation in motion-compensated DCT-based video encoders including a loop filter. We adopted the λ factor, which describes an exponential decay of the propagated error, but we discarded the γ factor,

which models propagation of errors in motion-compensated video, yielding:

$$D_{\mathrm{e}}(r_i, BER) = (1-\lambda)D_{\mathrm{e}}(r_{i-1}, BER)+\beta(1-(1-BER)^{L_{\mathrm{P}}})\mathrm{var}\left[X_i - X_{i-1}\right]. \tag{5.23}$$

Our observations are that this is an accurate model, although the propagated errors decay only slightly. For instance, for the `carphone` sequence we found that $\lambda = 0.02$ (not shown here). In a coder where loop filtering is used to combat error propagation, this factor is much higher [30].

5.5 Model Validation

We have now defined all models needed to solve Equation (5.5). Assuming we know the variances $\mathrm{var}\left[X_i\right]$, $\mathrm{var}\left[X_i - X_{i-1}\right]$, the parameters for the function $f(r)$ and $g(r)$, and the model parameters κ, L_{I}, L_{P}, α and β, we can minimize (5.5) using these models. Note that since in principle each frame can have its own R-D function, the function will get the additional parameter i to signify that.

$$D_{\mathrm{GOP}} = \frac{1}{N}\sum_{i=0}^{N-1}\left\{D_{\mathrm{q}}(r_i|i) + D_{\mathrm{e}}(r_i, BER|i)\right\}$$

$$\text{for } i = 0 \begin{cases} D_{\mathrm{q}}(r_0|i=0) & = \mathrm{var}\left[X_0\right] 2^{f(r_0|0)} \\ D_{\mathrm{e}}(r_0, BER|i=0) & = \alpha(1-(1-BER)^{L_{\mathrm{I}}})\left(\mathrm{var}\left[X_0\right] - D_{\mathrm{q}}(r_0|0)\right) \end{cases}$$

$$\text{for } i > 0 \begin{cases} D_{\mathrm{q}}(r_i|i) & = \left(\mathrm{var}\left[X_i - X_{i-1}\right] + \kappa D_{\mathrm{q}}(r_{i-1}|i-1)2^{g(r_i|i)}\right. \\ D_{\mathrm{e}}(r_i, BER|i) & = (1-\lambda)D_{\mathrm{e}}(r_{i-1}, BER|i)+ \\ & \quad \beta(1-(1-BER)^{L_{\mathrm{P}}})\mathrm{var}\left[X_i - X_{i-1}\right] \end{cases}$$

(5.24)

In this section we will verify these models by encoding a sequence of frames with different bit rate allocations and compare the measured distortion and the predicted distortion. Furthermore, we will introduce bit errors in the bit stream and verify the prediction of the distortion under error-prone channel conditions. As we mentioned in the introduction, we do not optimize (5.5) using the models (5.24) – as would be required in a real-time implementation. Instead we aim to show that it is possible to predict the overall distortion for a GOP under a wide range of channel conditions. We will show that a setting for N and r_i optimized with our behavior models (5.24) indeed yields a solution that is close to the measured minimum.

To validate our model, we will compare the measurements of the overall distortion of a GOP with the predictions made with our model (5.24). We used the JPEG2000 encoder/decoder as our video-coder (Figure 5.2), and encoded the

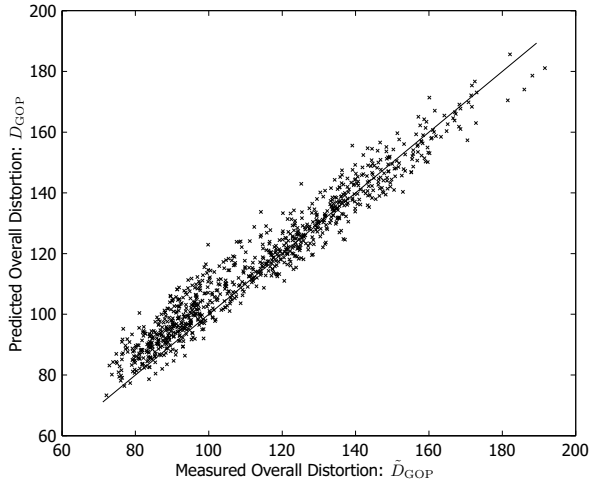


Figure 5.11 — *For each possible bit rate assignment, the cross(X) shows the measured distortion $\tilde{D}_{\mathrm{GOP}}$ horizontally and the predicted distortion D_{GOP} vertically. The line represents the points where the measurements would match the predicted distortion*

carphone sequence. In the first experiment a GOP of 10 frames was encoded with different bit rate allocations. No residual channel errors are introduced. In the second experiment, random bit errors were introduced in the encoded bit stream, to simulate an error prone channel. In the third experiment we addressed the issue of finding the optimal GOP length. In all these experiments we used the models (5.24) and the parameters we have obtained in Section 5.4 for the first ten frames of carphone. In the last experiment we used our models to optimize the settings for a whole sequence. We compare optimizing the settings with our models and with two other simple rate allocations. Furthermore we have investigated the gain that can be achieved if R-D curves are known for each individual frame instead of average R-D curves.

5.5.1 Optimal Rate allocation

In this experiment no residual channel errors were present ($BER = 0$) and the average bit rate available for each frame was 0.2 bpp. To each frame we assigned bit rates varying from $0.1, 0.2, 0.3$ to 1.1 bpp, while keeping the average bit rate constant at 0.2 bpp. The GOP length was set to 10. The total number of possible bit rate allocations with these constraints is 92378.

A GOP of 10 frames was encoded with each of these bit rate allocations. We then measured the overall distortion, denoted with $\tilde{D}_{\mathrm{GOP}}$, and compared

that with the predicted distortion D_{GOP} (using Equations 5.4, 5.9 and 5.14). Figure 5.11 shows the results. All points were plotted with the measured distortion $\tilde{D}_{\mathrm{GOP}}$ on the horizontal axis. The vertical axis shows the predicted distortion D_{GOP}. The straight line corresponds to the points where the prediction matches the measured values. Points under this line underestimate the measured overall distortion and the points above the line overestimate the measured overall distortion. The region we are interested in is located in the lower-left area, where the bottom-most point represents the bit rate allocation that minimizes our model D_{GOP} (5.24). The cloud shape gives good insight in the predictive strength of the model, since the points are never far off the corresponding measured distortion.

As we can see in Figure 5.11, the predicted distortion and the measured distortion correspond well over the whole range of bit rate allocations. Note that although it is not possible with these proposed behavior models to find the exact values of r_i yielding the minimal measured distortion (we only know the exact distortion after encoding and decoding), the predicted minimal distortion is close to the measured minimum distortion. We use the following metrics to express the performance of the model: the relative error

$$\varepsilon_1 = \mathrm{E}\left[\frac{D_{\mathrm{GOP}} - \tilde{D}_{\mathrm{GOP}}}{\tilde{D}_{\mathrm{GOP}}}\right] \cdot 100\%,$$

and the standard deviation of the relative error:

$$\varepsilon_2 = \mathrm{std}\left[\frac{D_{\mathrm{GOP}} - \tilde{D}_{\mathrm{GOP}}}{\tilde{D}_{\mathrm{GOP}}}\right] \cdot 100\%.$$

For this experiment, $\varepsilon_1 = 3.2\%$, which means that we slightly overestimated all distortions; $\varepsilon_2 = 5.7\%$, which means that on average our predictions were within $3.2 - 5.7 = -2.5\%$ and $3.2 + 5.7 = 8.9\%$ around the measured values.

We can interpret this in terms of $PSNR$: an increase of the error variance of 5.7% corresponds to a decrease of the $PSNR$ by $10 \log 1.089 = 0.37$ dB. This means that we predicted the average quality with 0.37 dB accuracy.

5.5.2 Rate Allocation for a Channel With Residual Errors

When residual channel errors were introduced, the same experiment yielded different results at different runs because of the randomness of bit errors. Therefore for each rate allocation, the coding should be done at least a thousand times, and the measured distortion values should be averaged. Analyzing each bit allocation with such accuracy is very demanding in terms of computing time, therefore we selected twenty cases uniformly distributed from the

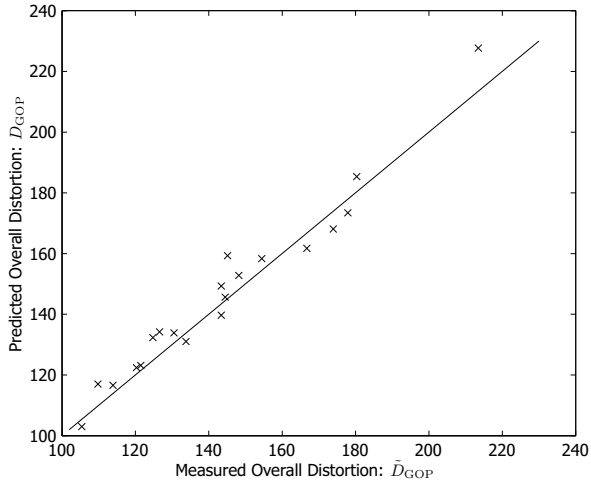


Figure 5.12 — *Selection of 20 bit rate assignments when $BER = 32 \cdot 10^{-6}$. For each case the cross(X) shows the measured distortion $\tilde{D}_{\mathrm{GOP}}$ horizontally and the predicted distortion D_{GOP} vertically. The line represents the points where the predicted distortion and the measured distortion would match.*

92378 rate allocations, to gain sufficient insight in the predictive power of the behavior models.

For this experiment we chose $BER = 32 \cdot 10^{-6}$. Figure 5.12 shows the measured average distortion $\tilde{D}_{\mathrm{GOP}}$ and the predicted distortion D_{GOP} for the 10-frame case. Now the relative error is $\varepsilon_1 = 2.0\%$, and $\varepsilon_2 = 3.7\%$.

Note that in these simulations, we did not use any special settings of a specific video coder, and used no error concealment techniques other than the standard JPEG2000 error resilience. Because of the combination of wavelet transforms and progressive bit plane coding in JPEG2000, in most cases the bit errors only caused minor distortions in the higher spatial frequencies. However, sometimes a lower spatial frequency coefficient was destroyed, yielding a higher distortion.

Any individual random distortion can differ greatly from the predicted one. Because large distortions are less likely to occur than small distortions, our model gives a boundary on the resulting distortion. We measured that for 88.0% of the cases, the measured distortion was lower than the predicted value.

We then changed our BER to $1024 \cdot 10^{-6}$. Figure 5.13 shows the measured and the predicted distortion. For this high BER, the relative performance metrics were still good: $\varepsilon_1 = 0.31\%$ and $\varepsilon_2 = 3.6\%$. Note that these relative metrics are similar to the case without channel errors. This means that on the

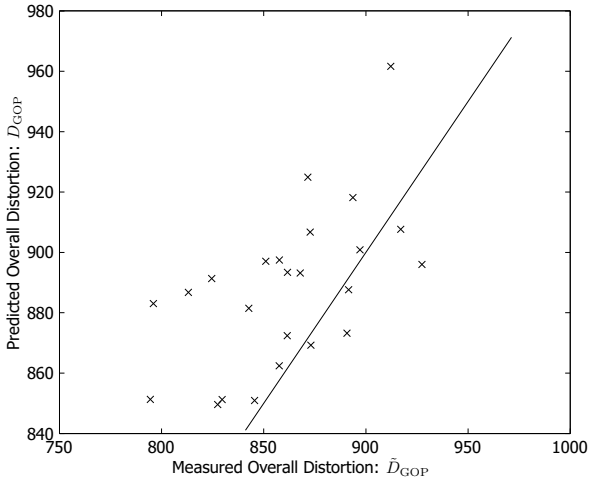


Figure 5.13 — *Selection of 20 bit rate assignments when $BER = 1024 \cdot 10^{-6}$. For each case the cross shows the measured distortion $\tilde{D}_{\mathrm{GOP}}$ horizontally and the predicted distortion D_{GOP} vertically. The line represents the points where the predicted distortion and the measured distortion would match.*

average, although the channel-error distortion is hard to predict, our model is still able to make good predictions of the average distortion even under error-prone conditions. Apparently the average D_{e} part of the total distortion is very predictable, this is probably due to the good error-resilience of the JPEG2000 encoder we used.

5.5.3 Selecting the Optimal GOP Length

In the previous experiments the optimal bit rate allocation was selected for each frame. This experiment deals with selecting the optimal GOP length N. The same constraints were used as in the previous experiment, but now the GOP length varied from 1 to 10.

Figure 5.14 shows for each GOP length from 1 to 10 the bit rate allocations for $BER = 0$. Observe that the average bit rate of 0.2 bpp per frame is spread out over each frame in the GOP, to obtain a minimal overall distortion D_{GOP}. The last case ($N = 10$) corresponds to the bottom-most point in Figure 5.11.

Figure 5.15 shows the predicted overall distortion D_{GOP} and measured overall distortion $\tilde{D}_{\mathrm{GOP}}$ for each of these bit rate allocations. Following our criterion (5.5), the optimal GOP length is $N = 8$. Since P-frames are used, we expect that using larger GOPs gives lower distortions. This is generally true, but in these experiments we did not cover the whole solution space since we

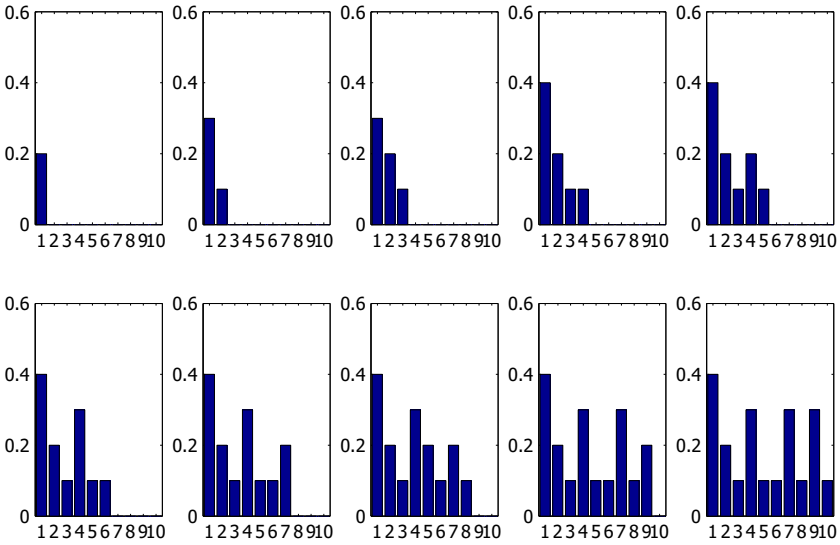


Figure 5.14 — *Bit rate allocations for $BER = 0$. Every plot corresponds to a GOP length running from $N = 1$ to 10. Within each plot, for each frame, the bit rate allocation that minimizes D_{GOP} is shown. The average bit rate is 0.2 bpp.*

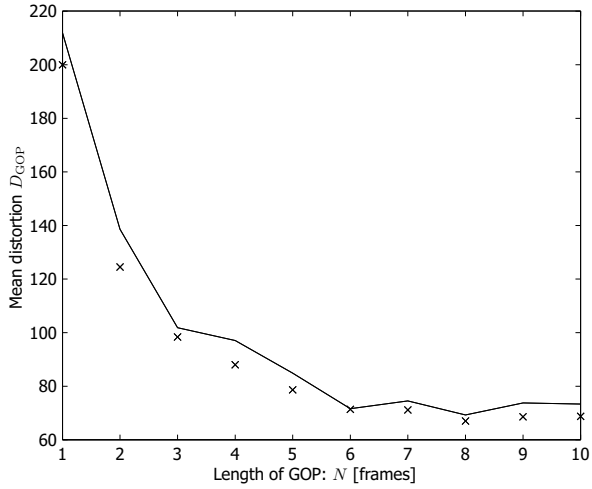


Figure 5.15 — *Minimized Distortion D_{GOP} (line) and $\tilde{D}_{\mathrm{GOP}}$ (x) for GOP lengths between 1 and 10 and for an average bit rate of $r = 0.2$ bpp.*

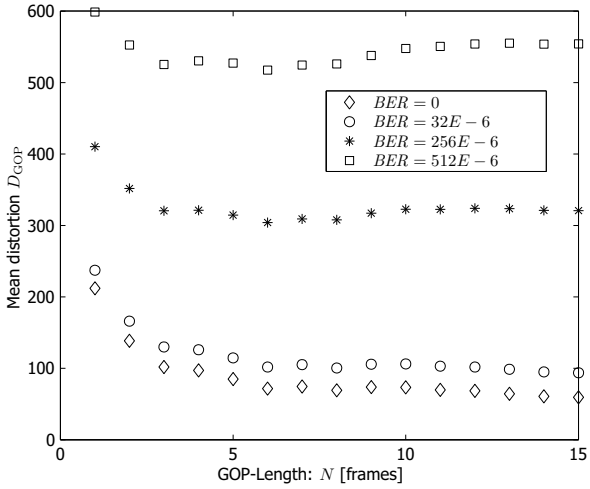


Figure 5.16 — *Minimized Distortion D_{GOP} for GOP lengths between 1 and 15, for different BERs and an average bit rate of $r = 0.2$ bpp.*

used increments of 0.1 bpp for the bit rates. With this limited resolution we may find suboptimal solutions.

Figure 5.16 shows the result of a simulation where N was varied from 1 to 15. In this simulation we only used our models to predict the distortion; the corresponding measurements were not carried out due to computational limitations (there are 600,000 combinations of rate allocations when bit rates 0.1, 0.2 to 1.6 are used). The distortions were again minimized with an average bit rate constraint of 0.2 bpp. The points correspond to the minimum achievable distortion D_{GOP} at each GOP length. We see that for $N > 6$ the average distortion did not substantially decrease anymore, so larger GOP lengths would not improve the quality greatly. Figure 5.16 also shows the results of the simulations for $BER = \{32 \cdot 10^{-6}, 256 \cdot 10^{-6}, 512 \cdot 10^{-6}\}$. Note that at some point the accumulated channel-induced distortion becomes higher than the gain we obtain from adding another P-frame. At this point, the internal controller should decide to encode a new I-frame to stop the error propagation.

5.5.4 Optimal Rate Allocation for Whole Sequences

In this experiment we used our models and our optimization criterion to optimize the settings for the whole sequence of `carphone`.

We have compared the measured distortion with two other simple rate allocation methods:

1. The rates and GOP length settings are obtained using our models and optimization criterion, with the constraints that $N_{\max} = 10$ and the average bit rate is 0.2.

2. Every frame has the same fixed bit rate $r = 0.2$. The GOP length is obtained using our models and optimization criterion.

3. Every frame has the same fixed bit rate $r = 0.2$. The GOP length has a fixed value of 10.

These methods were applied to the `carphone` and the `susie` sequences for $BER = 0$, $BER = 128 \cdot 10^{-6}$ and $BER = 512 \cdot 10^{-6}$. The results are shown in Table 5.1. For `carphone` method 1 is clearly better than method 3. Method 2 and method 3 perform more or less the same. When bit errors are introduced, method 1 still outperforms the other two. For `susie`, method 1 also outperforms the other two. When bit errors are present, method 2 (just adapting the GOP length), greatly outperforms method 3. We conclude that the performance of our method depends heavily on whether the characteristics of the source are changing over time or not. It seems that either optimizing the GOP length or the bit rates, decreases the distortion as opposed to method 3.

Finally, we have investigated whether using R-D parameters for each individual frame instead of average R-D parameters, indeed gives a significant increase of performance. We compared the case where for each individual frame the corresponding R-D function is used for optimization (case 1) and the case where one average R-D function is used for the whole sequence (case 2). For `carphone` we measured the following: for case 1, the average distortion $D = 76.5$, for case 2 this is $D = 91.0$. This means that significant gains can be expected when the R-D curves are known for each frame. Of course, in practice this is not possible. On the other hand, since consecutive frames look alike, we believe that an adaptive method to obtain the R-D curves from previous frames, could give significant gains. For `susie` we have similar results. For case 1 $D = 28.6$ and for case 2 $D = 47.9$.

Method	1	2	3
Case	***Distortion***		
`carphone`, $BER = 0$	76.6	91.1	90.6
`carphone`, $BER = 128 \cdot 10^{-6}$	136.8	161.3	161.1
`carphone`, $BER = 512 \cdot 10^{-6}$	397.7	408.4	410.4
`susie`, $BER = 0$	28.6	28.9	28.9
`susie`, $BER = 128 \cdot 10^{-6}$	47.4	49.6	59.5
`susie`, $BER = 512 \cdot 10^{-6}$	116.4	117.1	151.2

Table 5.1 — *Comparison between different rate allocation methods*

5.6 Discussion

In this chapter we introduced a behavior model that predicts the overall distortion of a group of pictures. It incorporates the structure and prediction scheme of most video coders to predict the overall distortion on a frame-per-frame basis. Furthermore, the model corrects for statistical dependencies between successive frames. Finally, our model provides a way to predict the channel-induced distortion when residual channel errors are present in the transmitted bit steam.

Although the deviation of the model-predicted distortion from the measured distortion can become substantial, with this model we can still compare different settings and select one likely to cause the smallest distortion.

Our models are designed to closely follow the behavior of the encoder given the characteristics of the video data, and to make an accurate prediction of the distortion for each frame. These predictions are made before the actual encoding of the entire group of pictures. To predict the average distortion, we need to know the variance of each frame and the variance of the frame difference of the consecutive original frames. We also need two parameterized R-D curves, and six other parameters ($\kappa, \alpha, \beta, L_{\mathrm{I}}, L_{\mathrm{P}}$ and λ).

In our experiments—some of which were shown here—we noticed that these parameters do not change greatly between consecutive GOPs, therefore they can be predicted recursively from the previous frames that have already been encoded. On the other hand we have shown that significant gains can be expected when the R-D parameters are obtained adaptively and no average R-D curves are used. The factors $\kappa, \alpha, \beta, L_{\mathrm{I}}, L_{\mathrm{P}}$ and λ do not depend greatly on the source data, but rather on the coder design, and thus may be fixed for a given video encoder.

After obtaining the frame differences, the distortion can be predicted before the actual encoding takes place. This makes the model suitable for rate control and CBR as well as for QoS controlled encoders. Although we focussed on rate allocation of entire frames rather than on macro blocks, all models can be generalized for use at macro block level.

Six

Optimized Video-Streaming over 802.11 by Cross-Layer Signaling

6.1 Introduction

Using wireless links for video streaming over the Internet is something that becomes more and more common today. This combination makes that the demanding world of real-time multimedia (which does not tolerate drop-outs for example) meets the quite imperfect – and capricious – dark universe of radio links. A lot of effort is required to team up these worlds, such that the stringent packet delay, jitter, and loss requirements of multimedia applications can be met by unstable and unreliable radio links.

Wireless links introduce bottlenecks for a number of reasons. First, communication over a wireless channel is simply not able to achieve the same quality (throughput, error rate, etc.) as its wired counterpart, which reduces the quality of the multimedia content that can be delivered. Second, in a mobile environment, the channel conditions can change rapidly due to changing distance between the stations (user mobility), Rayleigh fading, and interference. Since multimedia streaming applications must deliver their content in real-time, they are very sensitive to jitter in packet delivery caused by retransmissions in the underlying transport protocols. Third, multimedia streaming may be done over a shared medium like 802.11 and interfere with other users that are for instance downloading files.

With today's 802.11 products, the fundamental problems of wireless communication are aggravated by poor handling of the limited and imperfect re-

This chapter was published as: I. Haratcherev, J. R. Taal, K. Langendoen, R. L. Lagendijk, and H. J. Sips, 'Optimized video streaming over 802.11 by cross-layer signaling,' *IEEE Communications Magazine*, vol. 44, no. 1, pp. 115–121, Jan. 2006. [41]

sources (scarce spectrum, noisy medium) available to the radio. In particular, current transport protocols and device drivers do not actively control the user-available parameters of the 802.11 MAC layer; they use default values instead.

In this article, we present an architecture for adaptive video streaming over 802.11. We will show by a number of experiments that the real-time video-streaming quality of the total system can be drastically improved by applying *link adaptation* and *cross-layer signaling* techniques. Link adaptation is a technique to handle the effects due to changes in the channel conditions and is typically being employed at the link (MAC) level [8, 24, 45, 89]. Basically, link adaptation is the process of automatically adjusting a number of radio/MAC parameters, so that optimal quality of packet transmission is achieved. Cross-layer signaling can be used to pass link quality information to a video encoder such that, for example, the degree of compression is changed and consequently the data rate [12, 53]. This mechanism can also be used to pass changes in throughput estimations to the application layer in case of medium sharing.

This article is organized as follows. In Section 6.2, we give an outline of our communication architecture and the use-scenarios that it is intended to cover. Sections 6.3 and 6.4 describe the basics of link adaptation and give an overview of the various approaches to automatic rate control in 802.11. Section 6.5 and 6.6 describe the use of cross-signaling techniques in video-streaming without and with medium sharing, respectively.

6.2 Communication system architecture

In designing an adaptive communication system, it is important to know what the inherent problems in wireless communication links are and their severity. Then, given the constraints that the application imposes, such as the performance quality variations that it can handle, an appropriate adaptive control mechanism can be designed. This control mechanism should keep the quality variations within the specified bounds, but not to the extend that the control becomes too complex.

Figure 6.1 depicts the architecture of our adaptive wireless communication system for video streaming. In this architecture both the the *Radio/MAC* and the *Video encoder* are rate-adjustable, meaning that the transmission rate of the radio/MAC and video coding rate of the video encoder can be dynamically changed (components radio rate and video rate control, respectively). The video encoder talks with the radio/MAC through the UDP/IP component layer of the system.

The rate control components obtain their information from the *Channel State Predictor* (CSP) and the *Medium Sharing Predictor* (MSP) components. The Channel State Predictor produces *Channel State Information* (CSI) that is used for link adaption by the rate controller driving the radio/MAC component (path 1).

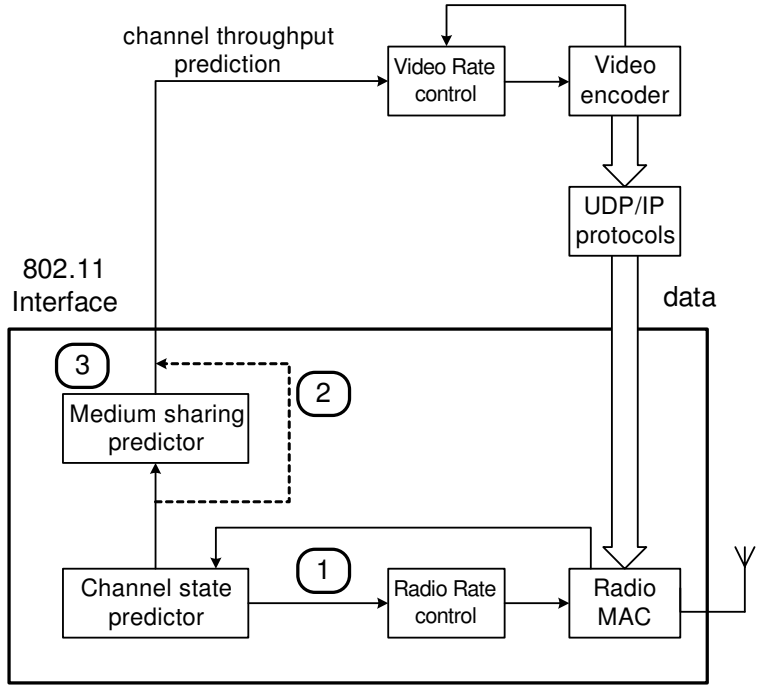


Figure 6.1 — *(1) Channel state prediction used for link adaptation; (2) Channel throughput prediction without considering medium sharing; (3) Channel throughput prediction considering medium sharing.*

This CSI is also supplied to the video encoder through a function that maps raw radio throughput to effective throughput, accounting for headers, etc (path 2). With *medium sharing*, the real link throughput is calculated from the CSI by a MSP (path 3), accounting for background traffic (and protocol overheads as before).

In Table 6.1 we show the communication system complexity as function of a number of wireless use scenarios. First, consider the first row of the table (no medium sharing). In the first two entries we have scenarios where the channel is static, i.e. there is no movement or any change in the link quality. In that case, we do not need any channel state prediction and we do not have to change any parameter in the radio. A typical example of such a scenario is a satellite link. The last two entries in the first row of the table depict scenarios that are dynamic. Here we definitely need channel state prediction to be able to adapt to the channel quality variations. Failing to do so will have catastrophic

effects, no matter how smart the network layers above are, just because a broken radio link means that no packets arrive at all.

Channel type	Static channel		Dynamic channel (movement)	
Quality	low	high	low	high
Sharing type	***Paths***			
No medium sharing	-	-	1	1+2
Medium Sharing	(3)	3	1+3	1+3

Table 6.1 — *Configuration for different video-streaming scenarios; the numbers refer to the paths in Fig. 6.1.*

Moreover, in the case of high quality video requirements, we also need the video encoder to have some information about the available throughput (paths $1 + 2$).

The picture changes completely when the wireless channel is shared (second row in Table 6.1). In [40], we have shown that the throughput can drop an order of magnitude, even with only a single additional user that is downloading. Therefore, we need to employ medium sharing prediction in addition to channel state prediction, for all cases (static/dynamic channel, low/high video quality).

6.3 Link adaptation basics

The (IEEE) 802.11 standard defines several MAC-level parameters (settings) that are available for tuning at the side of the wireless network interface card (NIC). The most important parameter available for adjustment is the transmit rate. In IEEE 802.11 WiFi Standard (802.11a), for example, the transmit rate can be set to 6, 9, 12, 18, 24, 36, 48 and 54 Mb/s. Each rate corresponds to a different modulation scheme with its own trade-off between data throughput and distance between the stations. This can be seen in Figure 6.2 - for clarity only the last four modulation schemes are shown.

The figure shows the performance in terms of the throughput for these modulation schemes versus the signal-to-noise ratio (SNR). Note that distance is related to SNR as $SNR \sim \frac{1}{dist^{\alpha}}$. More complex modulation schemes like 64-(QAM) 3/4 offer a larger throughput, but also have increased sensitivity to channel noise, and thus provide a shorter operating range. Usually, one wants to extend the operating range as much as possible and, at the same time, maximize the throughput. This can be done by proper (automatic) selection of the rate (modulation scheme) that gives the maximum throughput for certain con-

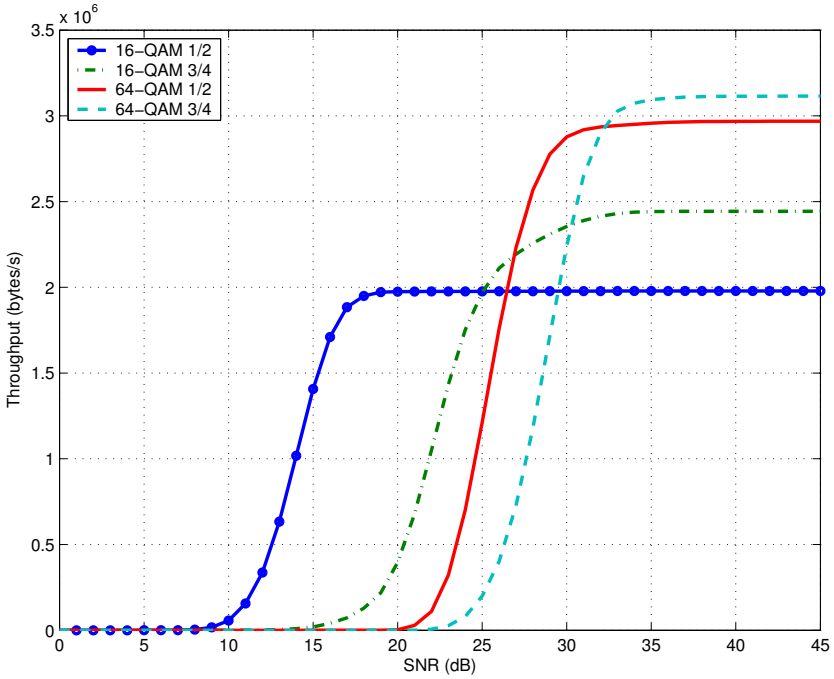


Figure 6.2 — *Throughput vs. (radio) SNR for some 802.11a modulation schemes.*

ditions. Obtaining a reliable SNR measurement from the channel is difficult, and may require indirect solutions as will be explained in Section 6.4. Knowing the optimal rate, we then compute the effective throughput at the user-level, so the video codec can adjust its parameters to avoid overloading the communication link when channel conditions degrade. We use a simple model to convert the radio rate setting to available user-space data throughput. Our model has been derived from the IEEE 802.11 standards [44] and is of the form:

$$T = \frac{8RL}{8L + bR + c}$$

where T is the throughput in Mbits/s, L is the length of a packet in bytes, R is the data rate setting in Mbits/s, and b and c are coefficients that depend on the 802.11 supplement. For 802.11a, $b = 161.5$ and $c = 156$. For IEEE 802.11 WiFi Standard (802.11b), $b = 754$ in the case of long preamble, or $b = 562$ in the case of short preamble, and $c = 112$.

6.4 Automatic MAC rate control

The IEEE 802.11 standard [44] and its supplements do not specify any algorithm for automatic rate selection. To our knowledge, all of the known vendors of 802.11 equipment use statistics-based approaches for rate control, which are slow to respond to changes in link conditions. In the research community, another class of rate control algorithms has been studied. These control algorithms use SNR-related information as a feedback to improve the sensitivity to changes in link conditions. We will discuss both approaches below, as well as our hybrid solution that combines the advantages of the statistics-based and SNR-based rate controllers.

6.4.1 Statistics-based automatic rate control

An easy way to obtain the necessary information on the link conditions is to maintain statistics about the transmitted data like the frame-error rate (FER), acknowledged transmissions, and the achieved throughput. Since these statistics are directly related to the effective user-level data throughput, they inherently guarantee that this throughput is maximized on the long-term. Three basic types of statistics-based rate control can be distinguished: *throughput-based*, *FER-based*, and *retry-based* rate control. The throughput-based approach is the one that uses the most global type of statistic and is the slowest method. The retry-based control uses the most local statistic (number of retries per frame), and is the fastest method.

In the throughput-based rate control a constant small fraction (10%) of the data is sent at the two adjacent transmit rates to the current one (an adjacent rate is the next higher or lower one available). Then, at the end of a specified decision window, the performance of all three rates is evaluated and the rate having the highest throughput during the decision window is selected. To collect meaningful statistics, the decision window has to be quite large (i.e. about one second), hence, the response to changes is rather slow.

In the FER-based rate control, the Frame Error Rate (FER) of the data stream transmitted over the link is used to select an appropriate rate [13]. The FER can easily be determined since under 802.11, all successfully received data frames are explicitly acknowledged by sending an (ASK) frame to the sender; hence, a missing ASK is a strong indication of a lost data frame. By counting the number of received ASK frames and the number of transmitted data frames during a rather short time window, the FER can be computed as the ratio of the two. The width of the time window and the rate-switching thresholds are critical for the performance of the FER-based algorithm. The optimal settings of the parameters are dependent on the link and the application, but are generally fixed at design time limiting the effectiveness of FER-based rate controllers.

An improvement over the FER-based approach is to downscale immediately when the MAC is struggling to transmit a frame correctly over the link. That is, to select the next lower rate after a small number of unsuccessful retransmissions (usually 5-10 retries) [45, 89]. This approach has to be implemented in hardware, since precise control of the rate setting in between retransmissions (of the same frame) is required. The advantage of this retry-based approach is that it combines a very short response time (a few frames) for handling deteriorating link conditions (downscaling) with a low sensitivity to traffic rates. The price to be paid is that the control algorithm is rather pessimistic. Relatively short error bursts cause long drops in throughput because upscaling to higher rates takes much longer than downscaling due to the need to collect a meaningful FER and to prevent oscillation.

6.4.2 SNR-based automatic rate control

A fundamental limit of indirect, statistics-based feedback is that it classifies link conditions as either 'good' or 'bad'. This binary information provides some notion about the direction in which to adapt the rate setting, but does not suffice to select the appropriate rate at once. This leads to a slow step-by-step accommodation to large changes in conditions, and introduces the risk of oscillation in stable conditions. A better approach is to use direct measurements of the link conditions.

The Signal-to-Noise Ratio (SNR) is directly related to the bit-error rate in the link and, hence, to the FER. Consequently, the SNR is linked to the packet delay and jitter, and the throughput, and holds the potential of providing rich feedback for automatic rate control [8]. Knowing the current SNR and the throughput-vs-SNR curves for each rate setting (e.g. Figure 6.2) would solve the rate-selection problem instantly.

Despite the advantages, SNR-based rate control has not been applied in practice so far, because of the following three problems:

1. in reality, for certain link conditions the relation between the optimal rate and SNR is highly variable. This is due to the imperfectness of the models describing the radio channel.

2. it is not trivial to obtain a reliable estimate of the SNR of a link. Many radio interfaces only provide an uncalibrated signal strength indication (SSI)).

3. the rate controller, which is at the sending side, needs in fact the SNR observed at the receiving side.

Most work on using SNR information for automatic rate control is based on simulation and does not consider the practical difficulties of obtaining good

SNR estimates. It concentrates on the way in which the noisy and drifting SNR (problem 1) can be used to determine the correct rate setting [8, 81]. Holland et al. [43] do address the issue of how to communicate back SNR values (problem 3), but their rate selection algorithm still relies on a straight SNR threshold technique. Another approach is discussed in [24], where the assumption is made that the channel is symmetric, meaning that the SNR observed at either station is very similar for any given point in time. This assumption allows Pavon et al. to use the SNR of the last ASK frame as an indicator of the SNR at the other side, and to use it for selecting the rate of the next data frame to be sent.

6.4.3 Hybrid automatic rate control

Both the statistics-based and the SNR-based approaches have their advantages and disadvantages. The statistics-based approach gives robust performance and inherently maximizes the throughput in the long term. However, the main drawback is its slow response to changing link conditions, which can be a source of problems for real-time applications. The SNR-based rate control can respond very fast, but due to the uncertain and fluctuating relation between SNR information and BER of the link, it lacks stability and reliability. Therefore, a logical step forward is to combine the two approaches in a hybrid algorithm that will provide both robustness and fast response.

We have implemented such an SNR-based hybrid rate control. The core of this hybrid algorithm is a traditional statistics-based (throughput-based) controller. The decisions of the core controller can be overridden by a second feedback loop. This loop bounds the acceptable range of the (signal strength indication of acknowledged frames (SSIA)) values for each rate, based on the specific knee in the throughput-vs-SNR curve (cf. Figure 6.2). The SSIA is used instead of the SNR, since most radio interfaces provide only uncalibrated SSI information. To account for the drift and the lack of calibration of the SSIA readings, we employ an adaptive adjustment logic that updates the values according to the recent history of channel conditions. For more details see [38].

Our hybrid rate controller should not be affected by collisions caused by transmissions from other stations. First, the throughput-based controller bases its decisions on overall throughput measured for adjacent rate settings over relatively long periods of time. So, collisions will affect all the throughputs simultaneously, hence the relationship between them will not change, and consequently it will not change the controller decisions. Second, if there is a collision, there will be no ASK, so no SSIA will be returned by the radio chipset to the driver. Thus, the state of the second feedback loop of the rate controller will not change, and consequently the final decisions of the rate controller will also not be affected by collisions.

We are investigating the possibilities of implementing advanced estimation techniques, such as Kalman filtering, in the radio rate controller, as well. Although this might improve the quality of our predictions (of both SSIA values and throughput), the benefit of such techniques should be carefully studied. In our case the advantage is doubtful since it comes at the price of a heavy computational load. We perform predictions on driver/firmware level, therefore, such a computational burden is probably unacceptable.

We have compared the decoded video quality for different rate control algorithms in an experiment. In this experiment we used two laptops of which one has been put in a fixed position on a desk. The other laptop was carried out of the room, moved a few times up and down through the hallway, and then returned to the initial position next to the other laptop. By walking up and down the hallway we ensured that the conditions continuously improved or deteriorated, so we can inspect the behavior of the various rate control algorithms at different circumstances. During the whole experiment the first laptop was streaming an H.263 encoded video of the `carphone` sequence in (QCIF) format to the second laptop which decoded and recorded the received video.

A classical and easy measure for image quality is the PSNR) measure. The downside of this measure is that it not necessarily corresponds to the human perception and that it is not well-suited for moving pictures where sometimes frames are missing. To overcome this shortcoming we employed two measures: 1) the average PSNR: The average of the PSNRs of all frames, and 2) the human perceptual quality. To obtain a perceptual measurement we have shown each received video to fourteen people who had to give a mark between 0 and 5 (0=bad, 5=good). Both measures together should give a good indication of the effects of losing packets on the quality of the video.

We have compared the performance of the following three rate control algorithms: perfect (a fictitious algorithm with no lost packets), hybrid-based, and statistics-based. Table 6.2 shows for each algorithm the packet loss ratio, the perceptual quality, and the average PSNR.

Algorithm	***packet loss***	***perceptual quality***	***average PSNR (dB)***
Perfect	0.00%	5.0	37.34
Hybrid	0.15%	3.0	36.59
Statistics-based	7.01%	0.4	29.33

Table 6.2 — *Packet loss and PSNR measurements and perceptual quality rating for three different algorithms.*

The results in Table 6.2 show that the statistics based algorithm loses above forty-five times more packets than the hybrid algorithm. The video quality for the hybrid algorithm is therefore 7.26 dB higher than for the statistics based

algorithm. The perceptual rating confirms this difference. As can be observed from the perceptual measurements, even low packet-loss ratios already give significant lower qualities. This is due to the propagation of errors in consecutive frames. A typical effect of these errors is shown in Figure 6.3.

Figure 6.3 — *Standard rate control (left) and hybrid rate control (right).*

6.5 Cross-layer signaling

At the application layer, the video encoder can also adapt to the link quality by changing the compression degree for example, and thus modifying the data rate. This adaptation requires that the video encoder is able to sense the link quality, for example, by getting a feedback information from the decoder side. However, such a scheme is ineffective when the round-trip delays are too long. Using this approach also introduces additional overhead.

Our solution is use an adaptive video encoder on top of the hybrid rate control algorithm [39]. The video encoder will adapt based on the CSI provided by the channel state predictor (see Figure 6.1) and an additional forecast about what the link conditions are going to be in the next couple of tens of milliseconds.

The video codec we used is a H.263 codec that has been modified to support interaction with the link layer. Our version of the H.263 encoder supports a video rate control algorithm (VRCA) that tries to achieve a certain rate, by adjusting the quantization step size. The quantization step size is the main parameter that controls the compression of the video. This VRCA has been designed for CBR encoding, but it can also be used to dynamically change the bit rate that is produced by the encoder. The resulting bit rate not only depends on the selected quantization step size, but also on the statistics of the picture itself. Therefore, the VRCA can not set the bit rate beforehand and then expect that this bit rate will be exactly achieved.

The VRCA is implemented as a simple feedback control loop that consists of setting an initial quantization step size, encoding part of the picture, measuring the resulting intermediate bitrate, changing the step size accordingly and then continuing with the next part of the picture. In total there are nine parts of a picture frame for which the quantization step size can be adjusted, which generally is enough to be able to achieve a certain preset rate. With this algorithm, we are able to change the target bit rate for each individual frame, meaning that we have a maximum delay of 40 ms (25 frames/s) to respond to changes in the channel.

In this way, we are able to constantly adapt the target video encoding rate for the VRCA according to information from the MAC rate control algorithm. By coupling the rate control algorithms of the MAC and the video coder in this manner we can efficiently use the available transmission rate to maximize the picture quality.

The following experiment shows the effectiveness of our cross-layer signaling approach. One of the laptops was again placed in a fixed position and the other one was following a predetermined track. The track consisted of three parts -'lead-in', which is reaching from the room to a specific start position in the hallway, and waiting until certain time elapses (10s). Then the laptop was moved up and down three times the hallway (60s). Finally, the laptop was placed back ('lead-out') again into the room where the fixed laptop lies (20s).

We performed the above experiment for cross-signaling and no cross-signaling. In the case of no cross-layer signaling, we have set the target-rate for the VRCA to the average rate as was obtained from the VRCA in the coupled case.

In Figure 6.4 the quality (PSNR) is shown for the whole experiment (90s). In the left part ($0 - 10$s) the channel conditions are excellent, hence the high quality in both cases. The middle part is best described as having conditions changing from good to bad a few times. The right part has good conditions again.

As we can see in Figure 6.4, the cross-layer signaling case yields a higher PSNR than the no cross-signaling case during the whole experiment. Table 6.3 summarizes the number of skipped frames (by the encoder), the number of lost packets, and the average PSNR in the period between 10-70s. Looking at the average PSNR in the $10 - 70$s period, we conclude that the quality can be dramatically improved by informing the video codec of the actual present rate and a prediction for the near future.

6.6 Medium Sharing Prediction

The scenario described in the previous section assumes that the 802.11 medium is not shared by other users. If other users are also sharing the same medium the throughput can drop significantly [40]. To be able to cope with this sit-

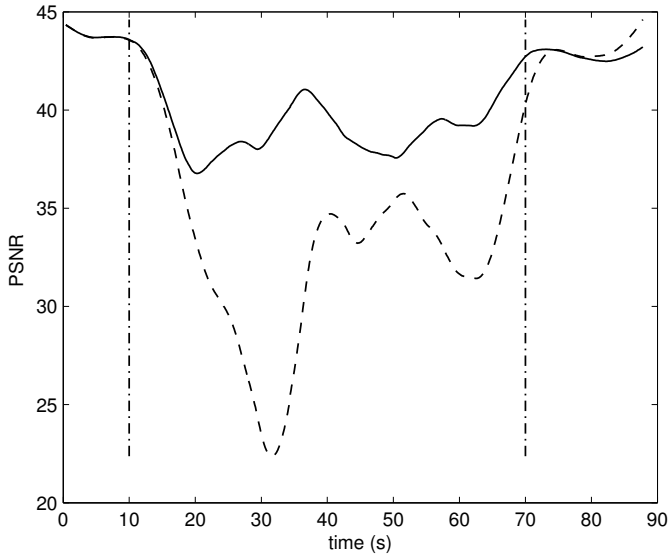


Figure 6.4 — *PSNR: the solid line shows how the quality (PSNR) changes during the experiment for cross-layer signaling. The dashed line shows the results for no-cross-layer signaling.*

Case	*# Skipped frames*	*# Lost Packets*	*PSNR Middle*
cross-layer signaling	50	13	39.3
no cross-layer signaling	216	23	33.1

Table 6.3 — *Cross-layer signaling and no signaling in rate control. The column 'PSNR' shows the average PSNR over changing conditions period between 10 and 70 seconds.*

uation, we need a throughput predictor. Transmitting over the air is usually a bursty process (streaming, downloading) and the time between switching from an active state and back to inactive state is in the order of seconds. As a consequence, a throughput prediction could be based on the statistics of the observed throughput for each radio rate setting during previous transmissions. As the medium utilization changes slowly, this prediction should work fine for the period that we are interested in (a couple of tens of milliseconds), provided we have a way to detect when other users start using the radio channel.

To test the viability of having a good performance throughput estimator (the Medium Sharing Predictor component in Figure 6.1), we have implemented a throughput prediction emulator. This emulator is in fact a lookup table with

throughput values for each rate setting, which we have measured beforehand in the presence of background traffic. The values are propagated as prediction values to the video encoder at the times we know the radio channel is used by other stations. In this way we can observe what the performance improvement would be in case of perfect throughput prediction.

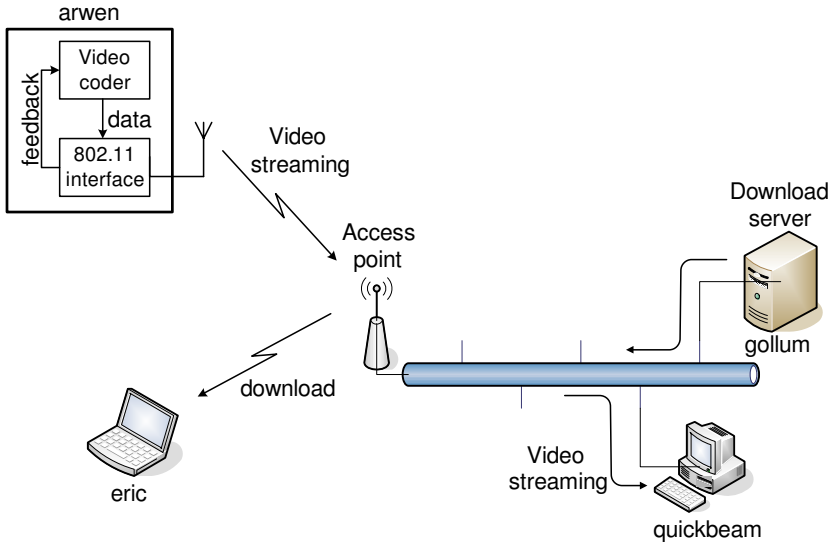


Figure 6.5 — *Our experimental setup.*

To show this, we have done the following experiment. The setup consists of two desktops ('golum' and 'quickbeam') on an Ethernet link, a 802.11a access point, and two laptops ('arwen' and 'eric') – both running Linux (Figure 6.5). The laptops are equipped with 802.11a cards based on the Atheros AR5000 chipset, and the card driver uses the advanced hybrid rate control algorithm. The experiments were carried out by streaming a video file between 'arwen' and 'quickbeam' while 'arwen' was moving, following a predetermined track. The track consisted of three parts - a 'lead-in' of walking from the room where the access point lies, to a specific start position in the hallway. Then a move followed taking the laptop up and down the hallway for about 20s ('action'). Finally, the laptop was moved back ('lead-out') again into the room. While the video streaming took place, 'eric' downloaded a file from 'gollum' (during the 'action' part) for about 10 seconds.

To evaluate the performance of the cross-layer signaling system in a shared medium (SM) scenario, we have examined two cases:

No-signaling The video encoder has no indication of the actual throughput,

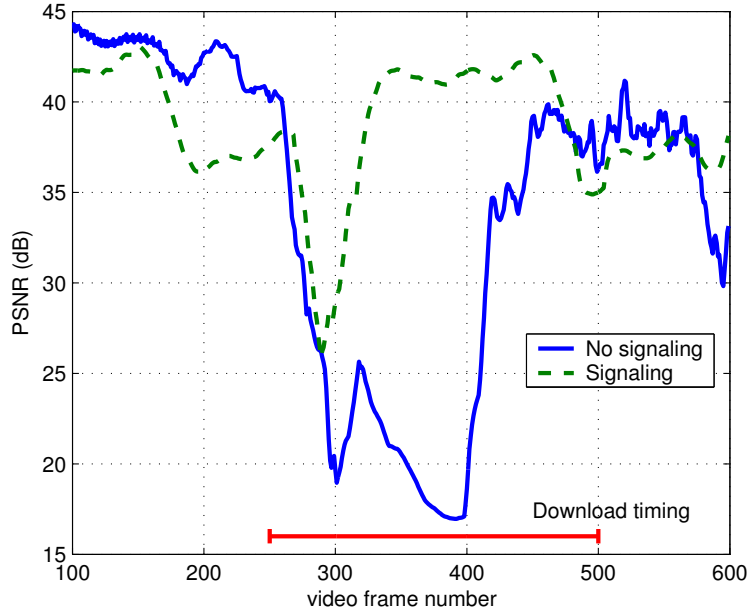


Figure 6.6 — *PSNR of the received video: The solid line shows the results for the No-signaling case using a fixed rate of the video encoder. The dashed line shows the quality when SM-signaling is on.*

and we set the target-rate for the VRCA to a fixed value, which is the *minimum* throughput that would be obtained in the case there is no medium sharing (about 4 Mbits/s).

SM-signaling The video encoder is informed about the effects of other users sharing the medium. The rate control loops of the MAC-layer and the video encoder are coupled through the throughput prediction feedback information provided by the MSP-component emulator (Figure 6.1).

Figure 6.6 shows the quality (PSNR) for the second ('action') part of the experiment. In the SM-signaling case just a few frames were lost, since the video encoder properly reduced the video-rate, following the throughput prediction feedback. In the No-signaling case, the higher video-rate selected by the video encoder caused the wireless interface to choke during the time the download took place, and the video freezed as a result of multiple frame losses. This 'freeze' can be localized by the regions with low PSNR. At about video frame 400, the video data managed to get through again, since the link conditions improved.

In the periods where there is no background traffic (video frames from 100 to around 250, and from around 500 to 600), the PSNR of SM-signaling curve is slightly lower than that of the No-signaling curve. This is because we have used the throughput prediction emulation for the duration of the whole experiment. This is to evaluate the loss of quality, when there is pessimistic misprediction of the medium sharing (fewer stations to share the medium with than predicted).

The mean PSNR over the whole measurement period of No-signaling is 38.1dB, and for SM-signaling it is 39.1dB. Focusing on the period with medium sharing (background download) we find that the mean PSNR is 28.3dB for No-signaling and 38.6dB for SM-signaling. In the SM-signaling case a significant improvement (over 10dB) is achieved when the medium is shared.

Seven

Fair Rate Allocation of Scalable Multiple Description Video for Many Clients

7.1 Introduction

Peer-to-Peer (P2P) networks and their file swapping P2P applications have become popularized in the past years because of their high download bandwidths and inherent server off-loading. This success has stimulated research into using peer-to-peer networks as infrastructures for streaming video over the Internet [67, 15]. In addition to being clients, the peers in the network then also serve as *application-level* multicast nodes. Although IP-level multicast offers efficient distribution from server to clients using routers, this mechanism is not widely spread nor often used. For application-level multicast (application level multicast (ALM)), however, no infrastructural changes are required, making fast and flexible deployment possible. Furthermore, the way the multicast is carried out can be specifically tailored to video streaming applications.

If a P2P network implements ALM, we can use intermediate nodes in the network to forward data to other nodes. The data forwarding model that we consider to be most suitable for video streaming over peer-to-peer networks is known as 'bartering'. Chunks of data are exchanged between nodes in a 'as fair as possible' manner. Bartering has two advantages. First, the clients exchange the data without server intervention, thus off-loading the server. Second, the clients are encouraged to participate in sharing the downloaded data

This chapter was published as: J. R. Taal and R. L. Lagendijk, 'Fair rate allocation of scalable multiple description video for many clients,' in *Proc. of SPIE, Visual Communications and Image Processing*, vol. 5960, July 2005, pp. 2172–2183. [76]

with other peers, which effectively combats the freeriding problem [58]. The underlying P2P network takes care of downloading all missing chunks from other nodes, and forwarding available chunks of data to nodes that still miss that data.

To apply the bartering model to streaming video, two possibilities exist. The first is to chop up the compressed video stream into data packets, and assign these packets to different network chunks. For example, chunk 1 contains video data packet $1, 3, 5, \ldots$, and chunk 2 contains video data packets $2, 4, 6, \ldots$. An important drawback of this approach is that since all packets are crucial for proper video decoding, missing chunks will result in major video degradation.

The second – and much more attractive – approach is to apply bartering on multiple descriptions (MD) of the (compressed) video source rather than straightforwardly chopping up a compressed bit stream. In this case, a video encoder generates several (multiple description) streams, that are all independently decodable. The different descriptions are put into the different chunks. If more chunks, and hence more descriptions, are received, the decoded video quality improves. The advantages of the MD approach is that video streaming becomes robust against P2P bandwidth variations or failing P2P nodes, which in both cases cause random chunks of data to be unavailable to a video decoder.

We use the multiple description coding approach MD-FEC proposed by Puri and Ramchandran [59] (see Figure 7.1). First, the video is encoded using an M-layer video encoder. Each layer k is encoded at a rate R_k and is then protected by an (M, k) (Reed-Solomon) erasure code. The thus obtained protected data is evenly distributed over the M descriptions. Finally, the individual descriptions form the chunks that are bartered by the P2P nodes. If any m out of M descriptions are available, the first m layers can successfully be decoded, resulting in a quality Q_m.

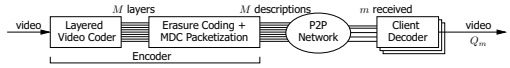


Figure 7.1 — *Block Diagram of the multiple description video distribution over a P2P network*

In the practical application of the above MD-FEC approach, several (encoding) parameters have to be chosen, namely the number of layers (and descriptions) and the bit rate at which the individual layers are encoded. These choices depend strongly on the quality that is desirable and achievable for clients with different bandwidth. In our earlier work [74] we have proposed to use a probabilistic client-bandwidth distribution model that describes clients bandwidth due to physical network limitations (e.g. the various ISDN, cable, ADSL, and LAN connections) and due to network congestion. In this way, the probability for a P2P peer receiving a certain number of descriptions can be calculated as a

function of the above mentioned parameters. Using an overall criterion, such as the average compression distortion over all clients, the optimal number of layers/descriptions and the bit rate per layer can be found.

In the next section we will discuss related work. We present our approach to MDC video streaming in P2P network in Section 7.3, and derive a model for the behavior of our MD-FEC system. Also we derive an RD-model for the layered Dirac codec. In Section 7.4 we first present three different criteria to find the optimal encoding parameters. We will introduce these criteria from the perspective of 'fairness' to the clients. After all, multiple description coding is inherently less efficient than single description coding, hence some clients 'pay' bandwidth or quality for the benefit of others. For instance, rather counter-intuitive, a criterion that minimizes the average MSE distortion results in mainly minimizing the distortion for clients with low bandwidth, whereas clients with high bandwidth gain little. We will also discuss the optimization of the encoding parameters for the selected fairness criteria. Finally, in Section 7.5, we show results of optimizing the fairness criteria with specific settings and assumptions on the client distribution. We conclude with a discussion.

7.2 Related Work

7.2.1 Peer-to-Peer Networks

A peer-to-peer network consists of a subset of all nodes present on a network (Internet). Just by having a connection to one or multiple nodes that are part of the P2P network, these nodes are (virtually) connected to the whole P2P network. The actual IP connection to these nodes may go through other nodes or routers that themselves are not part of the P2P network. Therefore the term *overlay network* is also often used in place of P2P network. Often there is no central registration site or server which controls or tracks the peers connected to the P2P network. This means that all functionalities of the P2P network have to be implemented in a distributed fashion. A commonly observed property of P2P networks is that the larger the P2P network, the more efficient, or the higher the performance. This is contrary to the single server solutions where the number of clients that can be handled, is often bounded.

A commonly used data exchange technique in P2P networks is 'bartering'. Chunks of data are bartered (traded) amongst peers. This greatly reduces the fan-out of a server. BitTorrent [57], a system that supports a large number of parallel file downloads, splits up the file into a number of chunks. Clients that already have downloaded certain chunks, forward these chunks to other clients, such that the server is relieved. Furthermore, the clients that are forwarding chunks are allowed to download more chunks simultaneously, resulting in a higher download speed. This tit-for-tat approach is good for the P2P

network as a whole. Since peers are encouraged to forward chunks for other clients, this greatly simplifies control of the network.

7.2.2 Multiple Description Coding

In multiple description coding (MDC) a certain amount of redundancy is added to transmitted (compressed) data, such that when one or more descriptions are lost, we are still able to recover the source data with an acceptable amount of distortion. Where error-correcting codes are typically used to correct bit and burst errors, MDC is used to handle situations in which losing entire packets or descriptions is likely.

There is a substantial body of literature on MDC of images. The MDC approaches can be divided into four categories, namely:

- MDSQ, in which nested quantizers or lattice vector quantizers are used to generate the multiple description. If one description is lost, the decoder effectively uses a coarser reconstruction [82],

- MDCT, in which uncorrelated signal components are transformed using a correlating transform, introducing a controlled amount of redundancy. If a correlated component is lost, the remaining component(s) can be used to recover the source data with an acceptable distortion [32],

- SBMDC. Source-based MDC: The autocorrelation naturally present in the source data is exploited to generate correlated descriptions [5],

- MD-FEC. This is combination of layered (or progressive) encoding and erasure coding. First a number of layers are generated. Then erasure codes of different strength are applied to each layer, such that, if m out of M descriptions is received, m layers can be successfully decoded [55].

The MD encoding of video is more complicated than MD encoding of random i.i.d. data. This is due to the complex autocorrelation structure of video, but also to the non-Gaussian distribution of the data. Furthermore, special care needs to be taken to prevent accumulation of errors in inter-frame decoding. An example of a MDC-SB like video encoder is the following [6]. The encoder generates an 'even' stream, containing even video frames $0, 2, 4, \ldots$ and an independently decodable 'odd' stream containing all odd video frames. If a description is missing, it can be estimated from the other description(s) because of the (inherent) correlation between successive video frames. A clear disadvantage of this (and any MDC) approach is the less efficient compression as correlation between odd and even frames is not exploited.

7.2.3 Application Layer Multicast (ALM)

The combination of multiple description coding and application layer multicast has been addressed by Castro *et al.* [15]. Their *Splitstream* system splits the video stream into two descriptions, similar to the above presented example where one description contains the even frames and the other description contains the odd frames. Splitstream constructs two different multicast trees with the same server and containing the same clients, but with different routes. When one of the nodes temporarily fails, each sub-trees is deprived of one description, but since the two subtrees are based on disjoint paths, it is very likely that the other description can still be received. Other proposals [56, 37, 27, 79] also describe streaming of media P2P networks, but these works concentrate mainly on P2P tree formation and handling of joins and leaves.

7.3 MDC Video Streaming in Peer-to-Peer Networks

7.3.1 Peer-to-Peer TV Architecture

In our previous work [58] we presented the basic design of a peer-to-peer television ((→P2P-TV)) system. In →P2P-TV, a separate P2P layer takes care of peer-finding, building multicast trees, handling joins and departures of nodes and handling (temporarily) failing peers. Furthermore, the P2P layer measures bandwidth latencies, peer failures etc, such that the video always can be efficiently distributed through the multicast tree.

The P2P video distribution system within →P2P-TV supports any peer to be the server of a video stream. At the same time, any peer in the network can subscribe to the stream. Figure 7.2 depicts an example of such a P2P network. The Server S is part of the P2P network and produces several descriptions ($D_1, \ldots, D_N$) of the video data. Client C_i receives these through peers $P_A, \ldots, P_Z, P_1, \ldots, P_N$. Another client C_j also receives these descriptions from another set of peers. In this example C_i forwards one or more descriptions to C_j. This general model enables simple packet forwarding and bartering to distribute the video descriptions through the P2P network to all clients. We assume that the total of incoming and outgoing bandwidth for each node C_i is limited to B_i. Usually, B_i forms the bandwidth bottleneck, because the rest of the network is normally able to support a total flow of B_i to node C_i. Since the total flow is split up in separate description, following different paths through the network, this is often a valid assumption. We also assume that the total bandwidth for server S, B_S is sufficient to distribute at least one copy of each description.

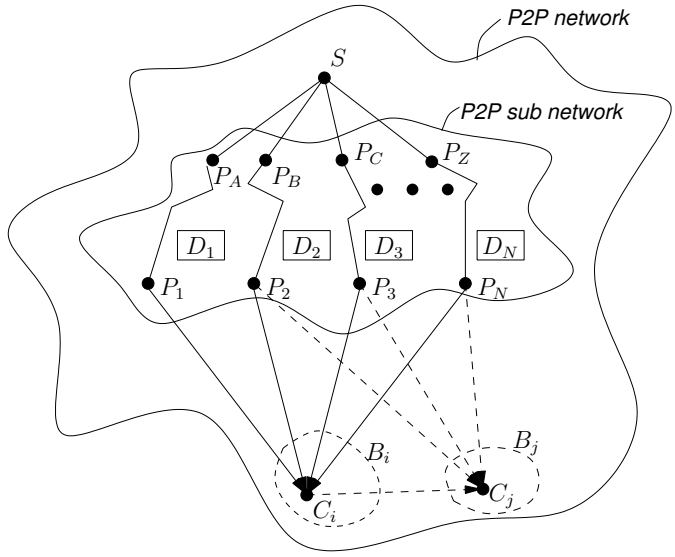


Figure 7.2 — *Example of an overlay P2P network.*

7.3.2 Layered MDC Video Encoder

As we mentioned in the introduction of this chapter, we use the Multiple Description Coding approach proposed by Puri and Ramchandran [59] for creating the descriptions that are streamed over the application layer multicast trees in the P2P network. We prefer their MD-FEC approach over other MDC approaches for two reasons. First, this MDC approach is flexible in the number of descriptions to generate. If an efficient layered encoder is present, any number of descriptions can be generated. Second, the behavior of the MD-FEC system is easy to model if the rate-distortion performance of the layered encoder are known. Hence, end-to-end rate-distortion optimization of the video streaming systems becomes possible.

The video coder we use generates an inter-coded base layer. The $(M-1)$ enhancement layers do not employ temporally predictive coding. On one hand this overcomes the effect of accumulating errors, since missing enhancement data has no effects on the next frame. On the other hand, we lose the efficiency of temporal predictive coding for the enhancement layers.

For decoding layer l, first all lower layers $1, \ldots, l-1$ should be decoded successfully. The layered coder is followed by a packetizer, as depicted in Figure 7.1. We generate multiple descriptions by combining erasure codes of each layer into packets. This way we ensure that when a client receives only a subset of the descriptions, it can still successfully decode a number of layers.

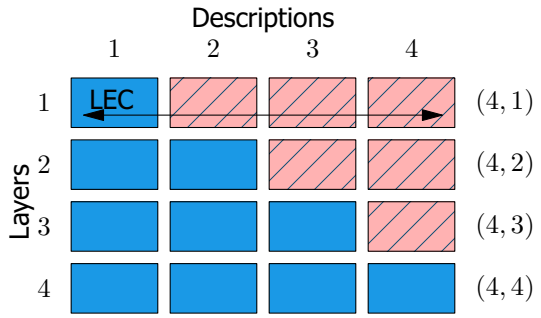


Figure 7.3 — *Illustration of a 4-description MD-FEC configuration. Each column comprises a description, consisting of parts from each layer.*

In general, an (M, k) block-code is able to correct $\frac{M-k}{2}$ random errors and $M-k$ erasures of which the positions are known. As shown in Figure 7.3, each description contains information of all layers using erasure codes. The data of layer l is first split up in l equal sized data blocks and then the remaining $M-l$ blocks are filled with the respective erasure codes. Any number l of these blocks is then sufficient to recover the original l data blocks of this layer. As a special, case layer 1 is in fact just copied in all descriptions. The last layer is only split up such that all descriptions are needed to reassemble that layer. The total number of layers M is equal to the number of descriptions. We have control over the redundancy added by erasure codes, since the rate of each layer is still free to choose. For instance, by using a small base layer rate and a large enhancement layer rate, we can induce a low redundancy.

The base layer is encoded at a rate R_1. The total rate of the first n layers is denoted by R_n. Hence, each successive layer i has a rate $R_i - R_{i-1}$. The rate R_D of each of the M descriptions is then given by:

$$R_{\mathrm{D}} = \sum_{l=1}^{M} \frac{R_l - R_{l-1}}{l} \tag{7.1}$$

$$R_{\mathrm{D}} = \sum_{l=1}^{M} \alpha_l R_l, \tag{7.2}$$

where $R_0 = 0$, and

$$\alpha_l = \frac{1}{l(l+1)} \quad \text{for } l = 1, 2, M-1, \text{ and } \alpha_M = 1/M. \tag{7.3}$$

A node or client in the P2P network with bandwidth R_c receives $m = \lfloor \frac{R_c}{R_{\mathrm{D}}} \rfloor$ descriptions. After decoding these m descriptions, the client experiences a

compression distortion of $D(R_1, R_2 \ldots, R_m)$ for $m = 1, \ldots, M$. In principle, the rate-distortion function for layer m depends on *all* rates R_k of the individual layers $k = 1, \ldots, m$ as layered compression induces losses relative to a single layer coder that are dependent on the rates of the (lower) layers. Hence, for each layer we may have a different rates-distortion function. By definition, $D(0) = \sigma_x^2$.

7.3.3 Layered Dirac Video Encoder

In our work we used *Dirac*, an open-source video coder developed by the BBC [26]. Dirac is wavelet-based and its performance is comparable to H.264. In Figure 7.4 the bit rate - distortion (RD) curve of Dirac is shown for the Foreman sequence. We also plotted several RD-curves for two-layer encoding. The branch-point in each curve is the rate R_1 for the first layer. The efficiency loss for the first enhancement layer is clearly visible. For higher layers (not shown) we have observed an insignificant loss of coding efficiency. Consequently, the rate distortion function becomes only dependent on the rate of the base layer R_1 and the rate of the layer m under consideration, i.e. , $D(R_1, R_2 \ldots, R_m) = D(R_1, R_m)$.

In order to be able to find an optimal rate allocation, we need to have an analytic model of the RD curves. First we model the single layer curve.

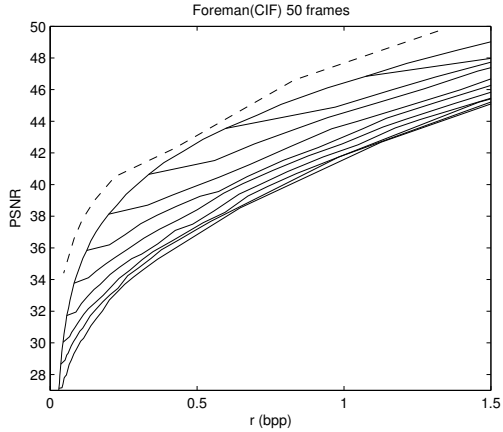
$$D(R) = \sigma_x^2 \, 2^{-2(R + a_0(1 - 2^{-b_0 R}))}. \tag{7.4}$$

This model embeds the information-theoretical bound for Gaussian i.i.d. sources ($a_0 = 0$), but also models the enhanced coding efficiency for autocorrelated sources such as video. For $R \to \infty$, the slope of the curve becomes the well-known 6dB per bit. For smaller R, the curve has a larger slope, which slowly decays to the 6dB bound. The top curve in Figure 7.4(b) shows the resulting model.

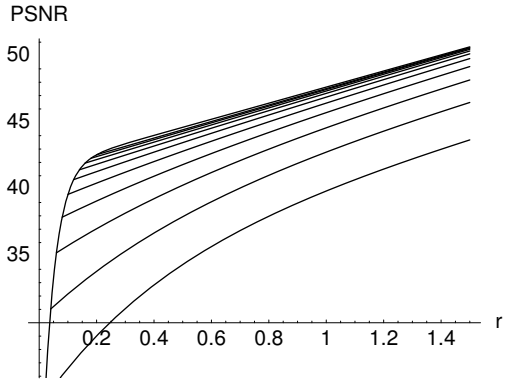
We see in Figure 7.4(a), that the slope of the two-layer curves $D(R_1, R_2)$ is smaller than for a single layer curve $D(R_1)$. By letting model parameters a and b depend on R_1, we can model the RD behavior for any combination of R_1 and R_2. Since we observed no additional loss in efficiency for layers 3,4..., the model for the m-layer encoder is based on the 2-layer encoder:

$$\begin{aligned} D(R_1, R_2 \ldots, R_m) =& D(R_1, R_m) = \\ =& \sigma_x^2 \, D(R_1) \, 2^{-2((R_m - R_1) + a(R_1)(1 - 2^{-b(R_1)(R_m - R_1)}))} \quad \text{for } m \leq M. \end{aligned} \tag{7.5}$$

Functions $a(R_1)$ and $b(R_1)$ are fit to the experimentally observed rate-distortion behavior of the coder.



(a)



(b)

Figure 7.4 — *(a) Rate-Distortion curves for the layered Dirac encoder with the 30Hz CIF-Format Foreman sequence. The dashed line is for the H.264 encoder. (b) Rate-Distortion model*

7.4 Fair MDC Streaming

7.4.1 Rate Allocation Problem

If we have to serve a large number of different clients, all with a different bandwidth R_c, we have to trade-off quality and redundancy of the descriptions. Making an optimal trade-off is not trivial as clients having different bandwidth will receive a different number of descriptions, and hence experience a different quality and effective bandwidth. On one hand, we wish to offer every client a quality that is as high as possible. On the other hand because of scalability of the system, we cannot offer each client individually an optimal stream. Furthermore, clients have to accept the fact that they have to forward packets to other peers, i.e. that they have to donate bandwidth to the P2P network, especially when we introduce redundancy by using MDC.

In order to deal with this dilemma in a fair way, two ingredients are needed. In the first place we need to know (or model) the distribution of the bandwidths R_c available to the clients. The distribution is modeled by the probability density function (PDF) $f_{R_c}(R_c)$. Secondly, we need to establish a criterion that expresses what we mean by 'fair', and which also lends itself for optimization. The fairness criterion will generally be a function of the clients bandwidth R_c and the bit rates R_k allocated to the layers of the video encoder. The fairness criterion is therefore denoted by $FC(R_c; R_1, R_2 \ldots, R_M)$. An in-depth discussion on the somewhat socio-economical question of how to distribute resources over a heterogeneous population is, however, outside the scope. In the following subsections we will introduce and discuss different examples of fairness criteria.

Given the distribution of the clients bandwidth and the fairness criterion, the optimal compression parameters $M, R_1, \ldots, R_M$ can then be found by maximizing

$$\{M, R_1, R_2 \ldots, R_M\} = \underset{M,R_1,R_2\ldots,R_M}{\operatorname{argmax}} \int_0^\infty f_{R_c}(r)\ FC(r; R_1, R_2 \ldots, R_M)dr \quad (7.6)$$

In most cases the rate allocation problem (finding optimal values for $R_1, R_2 \ldots, R_M$ given R_D and M) can only be solved numerically. The values of R_D and M, however, can either be optimized numerically but often are chosen practically. The P2P network, for instance, may only support a limited number of descriptions. Or, if we observe that most of the client bandwidths are smaller B_max, we could choise to select $R_D = B_\mathrm{max}/(M+1)$ so that the client-bandwidth spectrum is equally divided. In the remainder of this chapter we only discuss optimizing the rates $R_1, R_2 \ldots, R_M$ and we assume that we either have al-

ready selected values for M and R_{D}, or that these values are to be found in an outer optimization loop.

When R_{D} and M are fixed, the maximization function becomes:

$$\{R_1, R_2 \ldots, R_M\} = \underset{R_1,R_2\ldots,R_M}{\operatorname{argmax}} \sum_{j=0}^{M} \int_{b_j}^{b_{j+1}} f_{R_c}(r)\ FC(r; R_1, R_2 \ldots, R_j) dr \quad (7.7)$$

where,

$$\begin{aligned} b_i &= i\,R_D \quad \text{for } i = 0, 1, \ldots, M \\ b_{M+1} &= \infty. \end{aligned}$$

7.4.2 Minimal MSE Fairness Criterion

The first – and the most straightforward – criterion we consider is to average the distortion $D(R_1, R_2 \ldots, R_m) = D(R_1, R_m)$ over all clients:

$$\begin{aligned} \hat{D}(R_1, R_2 \ldots, R_M) &= \sum_{j=0}^{M} \int_{b_j}^{b_{j+1}} f_{R_c}(r) D(R_1, R_j)\, dr \\ &= \sum_{j=0}^{M} C_j\ D(R_1, R_j), \end{aligned} \quad (7.8)$$

where the number of clients receiving $i+1$ out of M descriptions is C_i, computed as:

$$C_i = \int_{b_i}^{b_{i+1}} f_{R_c}(r)\, dr \quad \text{for } i = 0, 1, \ldots, M. \quad (7.9)$$

Given a fixed number of description M and a fixed rate per description R_{D}, this criterion can be solved numerically using the Lagrange multiplier method, as discussed in the work of Puri *et al.* [59]

$$L(R_1, R_2 \ldots, R_M, \lambda) = \sum_{j=0}^{M} C_j\ D(R_1, R_j) + \lambda \left(\sum_{j=1}^{M} \alpha_j R_j - R_{\mathrm{D}} \right) \quad (7.10)$$

After equating the partial derivatives to zero, we obtain a set of equations for which the roots can be found numerically:

$$\frac{1}{\alpha_1}\sum_{j=1}^{M} C_j \frac{\partial D(R_1, R_j)}{\partial R_1} + \lambda = 0 \tag{7.11}$$

$$\frac{C_i}{\alpha_i}\frac{\partial D(R_1, R_i)}{\partial R_i} + \lambda = 0 \quad \text{for } i = 2, \ldots, M \tag{7.12}$$

$$\sum_{l=1}^{M} \alpha_l R_l - R_{\mathrm{D}} = 0 \tag{7.13}$$

In [59], a method is presented to solve these simultaneous equations such that after optimization $R_i < R_{i+1}$ holds for all rates. Unfortunately, the required conditions can only be verified for a single layer coder that is described by a single RD-function. In our case, however, we have to deal with a layered coder that is described by multiple RD-functions. Consequently, the derivative of $D(R_1, R_2 \ldots, R_m)$ with respect to R_i also depends on $R_j, 1 \leq j < i$. For that reason, we cannot use method in [59], and as a result we cannot guarantee that after optimization all $R_i < R_{i+1}$. Currently we deal with monotonicity of R_i as a postprocessing step *after* optimization. The resulting optimization procedure is illustrated in Figure 7.5.

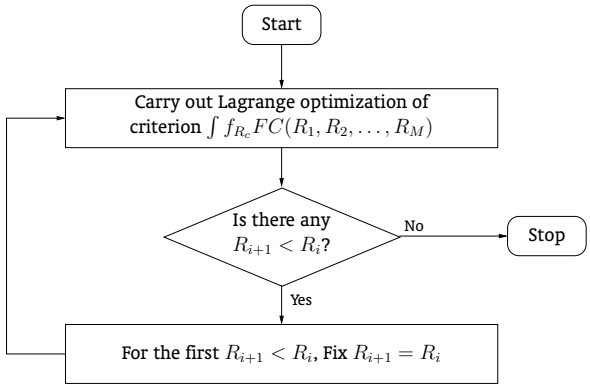


Figure 7.5 — *Optimization Algorithm which ensures an monotonic increasing sequence R_i*

7.4.3 Maximal Average PSNR Fairness Criterion

A drawback of the MSE metric described above is that the (inverse) magnitude of the MSE is not a good measure of quality. When the *average* MSE is

minimized, most effort is put in minimizing the highest MSE values. Little effort is put in minimizing the lower MSE values, although these still can gain significant amounts of quality. An obvious alternative choice for the fairness criterion is to use the performance measure often used in video compression, namely peak-SNR. When we average the PSNR over all clients, the following criterion is obtained:

$$\widehat{PSNR}(R_1, R_2 \ldots, R_M) = \sum_{j=0}^{M} \int_{b_j}^{b_{j+1}} f_{R_c}(r)\ PSNR(R_1, R_j)\, dr \quad (7.14)$$

where $PSNR(R_1, R_j) = 10 \log_{10} \frac{255^2}{D(R_1, R_j)}$. We can optimize the above criterion in a similar way as the minimum MSE criterion. The Lagrangian function is then given by

$$L(R_1, R_2 \ldots, R_M, \lambda) = \sum_{j=0}^{M} C_j PSNR(R_1, R_j) + \lambda \left(\sum_{j=0}^{M} \alpha_j R_j - R_{\mathrm{D}} \right). \quad (7.15)$$

After partial differentiation to R_i we obtain the following set of equations:

$$\frac{1}{\alpha_1} \sum_{j=1}^{M} C_j \frac{1}{D(R_1, R_j)} \frac{\partial D(R_1, R_j)}{\partial R_1} + \lambda = 0 \quad (7.16)$$

$$\frac{C_i}{\alpha_i} \frac{1}{D(R_1, R_i)} \frac{\partial D(R_1, R_i)}{\partial R_i} + \lambda = 0 \quad \text{for } i = 2, \ldots, M \quad (7.17)$$

$$\sum_{l=1}^{M} \alpha_l R_l - R_{\mathrm{D}} = 0 \quad (7.18)$$

The procedure outlined in Figure 7.5 can also be used for solving the above set of equations.

7.4.4 Weighted PSNR Loss Fairness Criterion

From the client's point of view, the client is paying for a certain bandwidth R_c and wishes to use this bandwidth as efficient as possible, in particular, the client wishes to obtain maximal quality of the received compressed video stream. Unfortunately, because of the multiple description coding, there is an inherent reduction in quality. As long as there is a balance between the MDC-induced quality reduction and the benefit for the entire P2P video distribution

system (of which the client is an integral part), the client is willing to take part in the system.

The third proposed fairness criterion reflects the MDC-induced quality reduction for the individual clients. We consider the system to be fair when the (psnr) quality of all clients is close to the performance obtained *if* a single description coding system (SDC) had been used. The following *weighted PSNR loss* (WPL) criterion measures the difference between MDC and SDC performance. The p-parameter controls the way in which differences are weighted:

$$\mathrm{WPL}_p(R_1, R_2 \ldots, R_M) = \left| PSNR_{\mathrm{SDC}}(R_c) - PSNR_{\mathrm{MDC}}(R_1, R_2 \ldots, R_{\lfloor \frac{R_c}{R_\mathrm{D}} \rfloor}) \right|^p$$

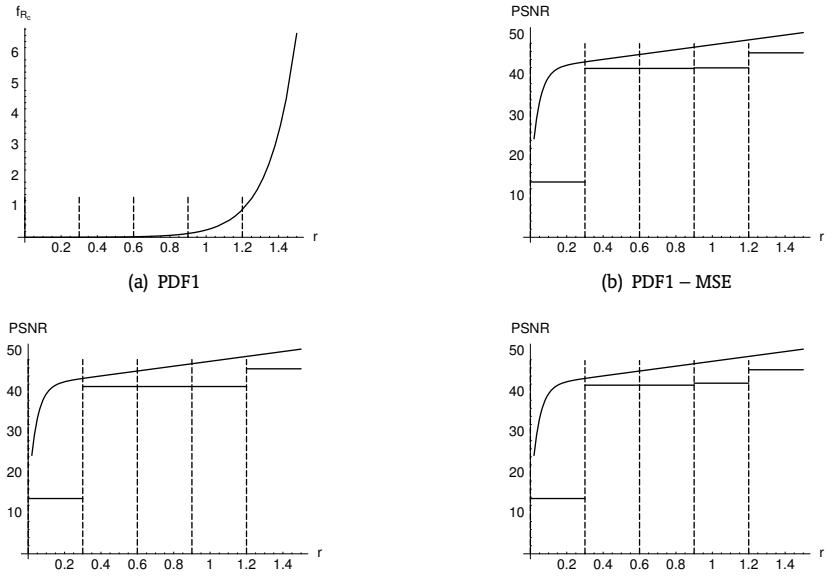
After substitution of this criterion in Eq. (7.7) and simplification of the resulting expression, we obtain:

$$\widehat{\mathrm{WPL}_p}(R_1, R_2 \ldots, R_M) = \sum_{j=0}^{M} \int_{b_j}^{b_{j+1}} f_{R_c}(r) \left| 10 \log_{10} \frac{D(r)}{D(R_1, R_j)} \right|^p dr \,. \quad (7.19)$$

With $p = 1$, this criterion becomes identical to the maximal average PSNR criterion Eq. (7.14). For $p > 1$ we put more emphasis on larger quality reductions. Minimizing the criterion then results in a solution where the difference between MDC case and the SDC case is more balanced over all clients. Note that for $p \to \infty$, we effectively minimize the maximum quality reduction.

The Lagrangian for the weighted PSNR loss with parameter p (WPL-p) criterion is

$$L(\lambda, R_1, R_2 \ldots, R_M) = \sum_{j=0}^{M} \int_{b_j}^{b_{j+1}} f_{R_c}(r) \left| 10 \log_{10} \frac{D(r)}{D(R_1, R_j)} \right|^p dr + \lambda \left(\sum_{j=1}^{M} \alpha_j R_j - R_\mathrm{D} \right), \quad (7.20)$$



(a) PDF1 (b) PDF1 – MSE

(c) PDF1 – PSNR (d) PDF1 – WPL-4

Figure 7.6 — *Results for the optimization of the MSE, PSNR and WPL-p criteria for client distribution PDF1)*

resulting in the following set of equations:

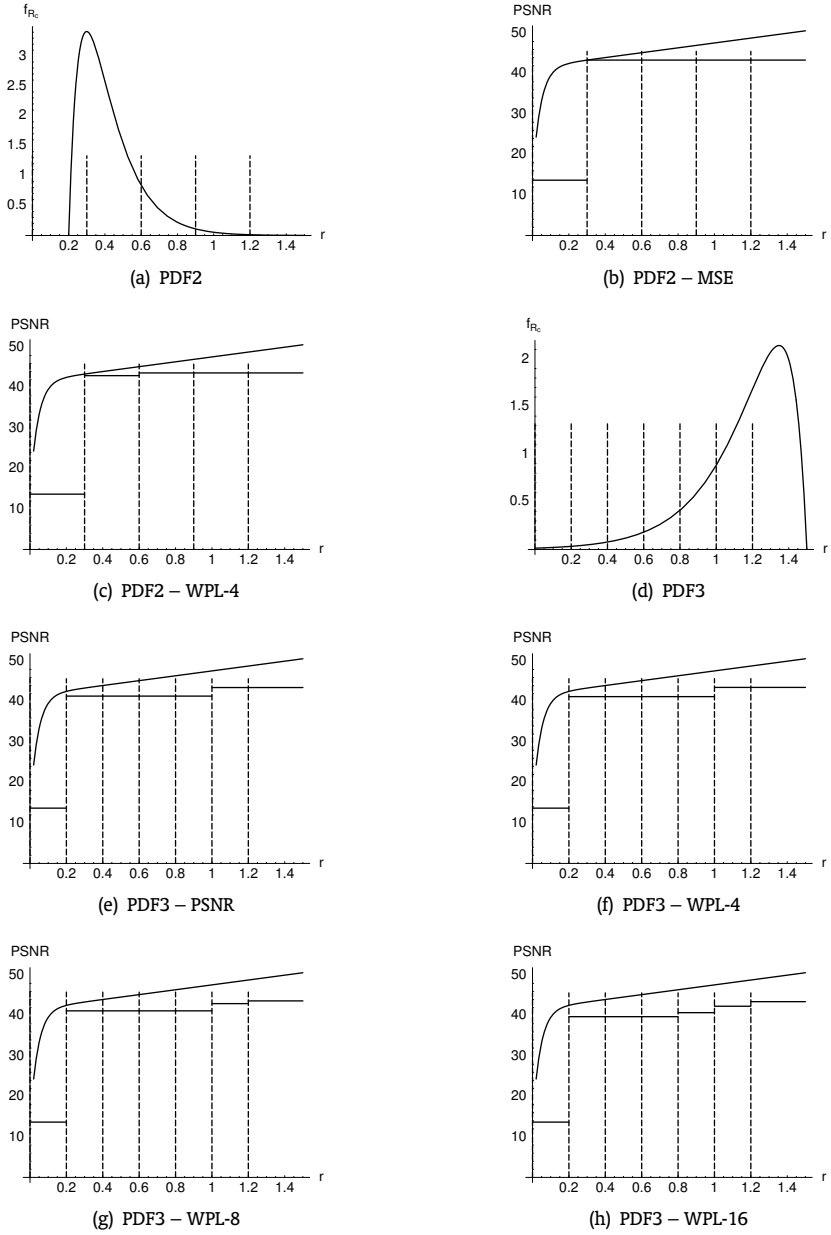
$$\frac{1}{\alpha_1}\sum_{j=1}^{M}\frac{-p\left(\int_{b_j}^{b_{j+1}} f_{R_c}(r)\log\frac{D(r)}{D(R_1,R_j)}^{p-1}dr\right)}{D(R_1,R_j)}\frac{\partial D(R_1,R_j)}{\partial R_1}+\lambda \quad = 0 \tag{7.21}$$

$$\frac{1}{\alpha_i}\frac{-p\left(\int_{b_i}^{b_{i+1}} f_{R_c}(r)\log\frac{D(r)}{D(R_1,R_i)}^{p-1}dr\right)}{D(R_1,R_i)}\frac{\partial D(R_1,R_i)}{\partial R_i}+\lambda = \quad 0 \tag{7.22}$$

$$\text{for } i=2,\ldots,M$$

$$\sum_{l=1}^{M}\alpha_l R_l - R_{\mathrm{D}} \quad = 0 \tag{7.23}$$

Similar to the other two criteria, the procedure outlined in Figure 7.5 is used for solving the above set of equations.



(a) PDF2

(b) PDF2 – MSE

(c) PDF2 – WPL-4

(d) PDF3

(e) PDF3 – PSNR

(f) PDF3 – WPL-4

(g) PDF3 – WPL-8

(h) PDF3 – WPL-16

Figure 7.7 — *Results for the optimization of the MSE, PSNR and WPL-p criteria for several client distributions (PDF2 and PDF3)*

7.5 Results and Experiments

In this section we present the results of the optimization of above criteria given several different client-bandwidth distributions and different choices for R_{D} and M. In these simulations we used the general RD model as derived in Section 7.3.3.

7.5.1 Rate Optimization

The first distribution we consider is shown in Figure 7.6(a). We expressed the clients bandwidths in bit-per-pixel, also note that $f_{R_c}(r > 1.5) = 0$. Furthermore we fix $M = 4$ and $R_{\mathrm{D}} = 0.3$. Using the MSE criterion, this resulted in rate assignment

$$\vec{r} = \{R_1, R_2, R_3, R_4\} = \{0.15, 0.15, 0.18, 0.74\}$$

. The resulting quality depending on the number of descriptions that is received (and therefore depending on the client bandwidth) is shown in Figure 7.6(b). This plot shows MDC quality depending on the number of received descriptions in the lower curve. For reference we also plotted the SDC quality $D(r)$ (upper curve). A first observation is that the first layer already obtains a fairly high quality. This has two reasons: a) the RD-function is very steep at lower rates, making it 'cheap' to obtain low distortions at low rates; b) since the RD-model correctly models the penalty in coding efficiency in the enhancement layers, a small base layer implies lower qualities for the higher layers as well. Another observation is that only for clients with bandwidth $R_c > 1.2$, the system really increases the base quality.

When we use the PSNR criterion, the optimal assignment is

$$\vec{r} = \{0.14, 0.14, 0.14, 0.78\}$$

(Fig. 7.6(c)). Although barely noticeable, this criterion favors the higher bandwidths with a little bit higher quality. When we apply the WPL-4 criterion however,

$$\vec{r} = \{0.15, 0.15, 0.32, 0.69\}$$

, this results in a more 'fair' distribution of the quality over all clients. Clients receiving three descriptions do obtain a little higher quality than when receiving only two descriptions, making it worthwhile to participate in de P2P network.

Now we consider the distribution in Figure 7.7(a). This distribution puts emphasis on clients having bandwidths between 0.3 and 0.6. Since $R_{\mathrm{D}} = 0.3$, all clients with $R_c < 0.3$ receive no descriptions at all. For the MSE criterion ($\vec{r} = \{0.3, 0.3, 0.3, 0.3\}$) gives a high quality to all clients, but effectively results in a single-layer, single-description solution. With the WPL-4 criterion on the other hand,

$$\vec{r} = \{0.20, 0.31, 0.45, 0.45\}$$

and even clients with a $R_c > 0.6$ still gain quality at a small expense of quality for the lower bandwidth clients (Fig. 7.7(c).

For the distribution shown in Figure 7.7(d), we fixed $M = 6$ and $R_{\mathrm{D}} = 0.2$. To gain insight in in the effect of the p-parameter, we varied p from 1 (effectively the PSNR criterion) to $p = 16$. Figures 7.7(e)-7.7(h) display the results. Remember that a higher p-value puts more emphasis on minimizing larger offsets between the SDC an MDC quality. Using a large p, the criterion goes to a greater extent to prevent large offsets, which results in a system that really offers different qualities for different bandwidths.

7.5.2 Streaming experiment

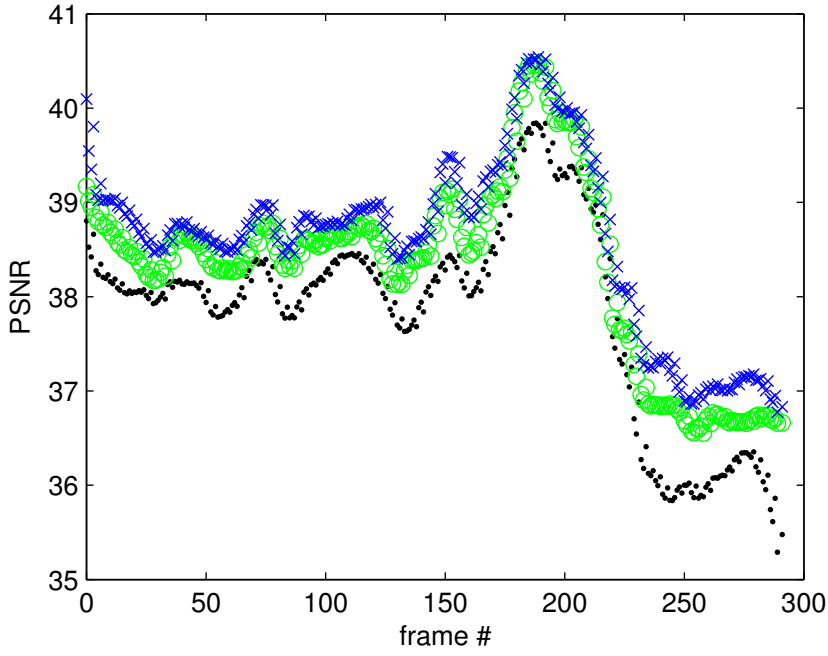


Figure 7.8 — *Video streaming experiment with $\vec{r} = \{0.20, 0.31, 0.45, 0.45\}$. Each curve shows the obtained PSNR per frame, when one (·), two (○), three or four(×) descriptions are received.*

We have performed a streaming experiment on the short Foreman sequence. For distribution PDF1 and fairness criterion WPL-4 we obtained after optimization the rate-setting $\vec{r} = \{0.20, 0.31, 0.45, 0.45\}$. We encoded the sequence with our layered-Dirac coder and obtained the results as plotted in Figure 7.8.

Note that our implementation does not employ an inner rate-control loop, hence the greatly varying PSNR quality of different frames. On average however, we clearly obtain different quality levels as requested by our rate setting.

Eight

Asymmetric Multiple Description Coding using Layered Coding and Lateral Error Correction

8.1 Introduction

Multiple Description Coding is a source coding method where a source is encoded into a limited number of descriptions such that, whenever some descriptions are lost, the quality gracefully degrades. In general some amount of redundancy has to be added in order to increase the error resilience and to enhance the gracefulness. In [59] the theoretical bounds for the case of two descriptions is presented. Goyal [32] gave an excellent analysis and discussion of these bounds and also presented a method to correlate i.i.d. sources in order to increase the error resilience. Most of the presented methods for MDC are however cases of symmetric MDC: Each description is equally important and equivalent (similar but not the same). In general, however, MDC can be asymmetric: descriptions are not equally important and may be prioritized: Having description *one* may give better quality than having only description *two*. Having both results in the highest quality. An extreme case of asymmetric MDC is Layered Coding, where the enhancement layer is useless (gives worst quality) when the base layer is not received. We can also think of intermediate cases where there is still some unbalance between the descriptions but each description will at least give some quality, although descriptions are still pri-

This chapter was published as: J. R. Taal and R. L. Lagendijk, 'Asymmetric multiple description coding using layered coding and lateral error correction,' in *Proc. twenty-seventh Symposium on Information Theory in the Benelux,*. Noordwijk: Werkgenootschap Informatie- en Communicatietheorie, June 2006, pp. 39–44. [77]

oritized. With nested quantizers and correlating transforms it is possible to generate these asymmetric descriptions, although most papers concentrate on balanced/symmetric MDC [83].

In this paper we investigate the encoding of asymmetric descriptions using layered coding and channel codes. We refer to [20, 76] for a discussion of symmetric MDC using layered coding and channel codes. In that case M MD-packets are assembled by dividing the bytes of each layer over the descriptions. For each layer i, a (M, i) Reed-Solomon code is computed, and these bytes are also divided amongst each MD packet. This way, when receiving an arbitrary combination of k MD packets, the original first k layers can be reconstructed. This method effectively removes the prioritization from the layered coding and introduces redundancy and hence error resilience. Each extra description will increase quality. Later (Section 8.3) we will show an example of this symmetric case as a special case of general (asymmetric) MD coding.

The main idea of our Asymmetric Multiple Description Coding method (AMDC) is to generate unbalanced descriptions by including different amounts of channel codes into each description. Thereby forming descriptions that contain more base information and descriptions that contain more enhancement information. It should be clear that we need to know the channel characteristics at forehand in order to fit or design the AMDC coder, the (un)balance of the channels should be matched by an unbalance of the descriptions. In the remainder of this paper we first present the general design of AMDC using channel codes and layered coding (Section 8.2). In Section 8.3 we first present other coding techniques as special cases of AMDC and compare them with our rate-constrained AMDC method as discussed in Section 8.4. In Section 8.6 we present an algorithm to find solutions for a rate constrained optimization of our AMDC method.

8.2 Asymmetric Multiple Description Coding

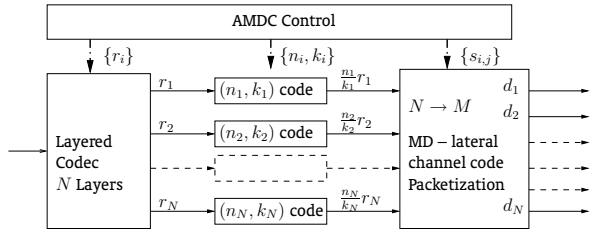


Figure 8.1 — *Asymmetric MDC coder + lateral error correction code block diagram*

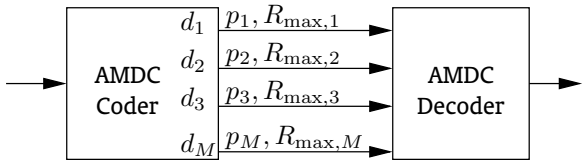


Figure 8.2 — *Model of AMDC streaming over asymmetric channels. The AMDC coder generates M descriptions which are transmitted over M channels. Each channel j has a, possibly different, packet loss rate p_j and transmission rate $R_{\max,j}$*

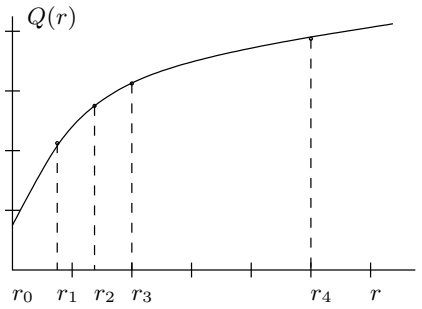


Figure 8.3 — *Example of Layered Coding RD-Curve. When using an FGS encoder. Layers can be generated at arbitrary rate points r_i.*

Layered Coding RD-Curve]

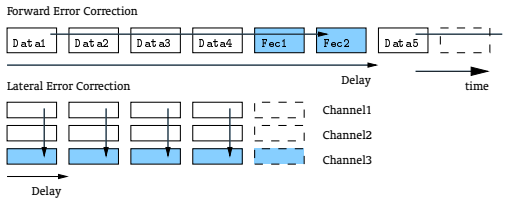


Figure 8.4 — *This (6,4) Forward Error Correction Reed-Solomon code operates on subsequent packets, therefore introducing a delay. Lateral Error Correction and MDC operate on parallel streams such that the delay is not increased.*

In this section we present the general design of AMDC. What does asymmetric MDC look like. In general, all descriptions contain some base information (low resolution or coarse information, like a base layer) and some refinement information. When not all descriptions contain the same amounts of these, we speak of AMDC. One way to obtain an AMDC code, is to extend the MD-FEC system with longer (Lateral) Error Correction codes (lateral error correction (LEC)) and where each description contains one ore more of these LEC blocks. Figure 8.1 shows a block diagram. The source is first encoded using a layered coder into L layers using rates $\{r_1, r_2, \ldots, r_L\}$ (See Figure 8.3). On each layer i a different (n_i, k_i) Reed Solomon Channel Code is applied (See Figure 8.4. The Packetization Block then Combines all data (layer data and channel codes) into MD Packets according to the packetization scheme (Matrix S). The AMDC control block controls the layered coding rates r_i, the channel codes and packetization matrix S.

In Figure 8.4, we explain the difference between Forward Error Correction (FEC), which operates on sequential packets, and Lateral Error Correction (LEC), which operates on parallel packets (or descriptions).

The general design of our AMDC method with L layers and M descriptions is as follows:

$$\begin{array}{c} \\ \\ \left[\begin{array}{cccc} s_{1,1} & s_{1,2} & \ldots & s_{1,M} \\ s_{2,1} & s_{2,2} & \ldots & s_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ s_{L,1} & s_{L,2} & \ldots & s_{L,M} \end{array}\right] \end{array} \begin{array}{c} S_{\text{asym}} \\ \begin{array}{cccc} d_1 & d_2 & \ldots & d_M \end{array} \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \text{LEC} \\ (n_i, k_i) \\ (n_1, k_1) \\ (n_2, k_2) \\ \vdots \\ (n_L, k_L) \end{array} . \tag{8.1}$$

The L rows of this Matrix correspond to the L layers as generated by the layered coder. The LEC column shows which lateral channel code is applied to each layer. Matrix S show the allocation/distribution of each of the n_i code blocks within each layer i over each description (columns). Each layer i is protected with a (n_i, k_i) code, where $n_i = \sum_{j=1}^{M} s_{i,j}$. When we accumulate the sizes of the code blocks in each description, we have

$$R_j = \sum_{i=1}^{N} (r_i - r_{i-1}) \frac{s_{i,j}}{k_i} \tag{8.2}$$

being the rate of description j, where $r_0 = 0$.

In order to evaluate the performance of this coding scheme we need to know for each combination of packet losses what layers can be successfully decoded. We define a 'description pattern' $V = \{v_1, v_2, \ldots, v_M\}$ where $v_j = 1$ when description j is received and zero otherwise. Set V contains all 2^M possible V's. In general, each V has a certain probability of occurring $\Pr\{V\}$.

Given pattern V, layer i can be decoded when all lower layers are decodable *and* its channel code can be decoded:

$$\text{Decodable?} \quad \sum_{j=1}^{M} v_j s_{h,j} \geq k_h \qquad \forall \quad 1 \leq h \leq i. \tag{8.3}$$

We define $l(V)$ as the number of layers that can be decoded given pattern V. We express the distortion obtained in this case as $D(r_{l(V)})$. As a more general case we use from now on a quality metric $Q(r)$.

Suppose each description pattern has a probability $\Pr\{V\}$, then we can define the following average quality measure.

$$\hat{Q} = \sum_{\forall V \in \mathsf{V}} \Pr\{V\} Q(r_{l(V)}). \tag{8.4}$$

For independent channels with loss probabilities p_j, we have

$$\Pr\{V\} = {p_j}^{1-v_j}(1-p_j)^{v_j}$$

In the next section we show some examples (solutions) of this problem for the case of three descriptions.

8.3 Special Cases of AMDC and examples

In order to get insight of the behavior and design of our AMDC coding method we will first show well-known coding methods as special cases of AMDC.One extreme case of asymmetric MDC is (pure) Layered Coding (hence no LEC), which only yields acceptable distortions when all layers $0, \ldots$ up till an arbitrary layer i are received. Layered Coding can be seen as a degenerate case of MDC. Another extreme case is symmetric or balanced MDC.

Layered Coding as extreme AMDC — With the first case of layered coding, we transmit each layer on a different channel, where the most important layer is transmitted on the safest channel. Suppose we confine ourselves to the case with 3 channels and 3 descriptions and since layered coding is considered a special case of AMDC, also only 3 layers. Furthermore we assume that the channels have increasing packet loss rates ($p_1 < p_2 < p_3$. In this case using our AMDC framework, layered coding looks like (Eq. 8.5):

This case does not include the often used Layered Coding with layered coding and unequal error protection (LC-UEP). With LC-UEP each layer has a different amount of error protection, according to the priority of each layer. It is clear that the UEP is based is forward error correction (Fig.8.4 and is thereby introducing possibly unwanted delay.

Symmetric MDC case — Symmetric MDC is special in the sense that matrix S only contains ones and $\vec{k} = \{1, 2, 3, \ldots\}$ (Eq.(8.6).

$$
\begin{array}{cccc}
 & S_{\text{layered}} & & \text{LEC} \\
d_1 & d_2 & d_3 & (n_i, k_i) \\
\left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] & & & \begin{array}{c}(1,1)\\(1,1)\\(1,1)\end{array}
\end{array} \tag{8.5}
$$

$$
\begin{array}{ccccc}
 & & S_{\text{sym}} & & \text{LEC} \\
 & d_1 & d_2 & d_3 & (n_i, k_i) \\
 & \left[\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array}\right] & & & \begin{array}{c}(3,1)\\(3,2)\\(3,3)\end{array}
\end{array} . \tag{8.6}
$$

8.4 General AMDC constrained by channel rates

The most general optimization criterion can be defined when the rate on each channel is limited and potentially different. Furthermore we assume that also the packet loss rates on each channel are different (See Figure 8.2).

$$
\max_{\substack{S_{i,j},k_i,r_i \\ \forall 1 \le ,i \le L \\ \forall 1 \le j \le M}} \sum_{\forall V \in \mathsf{V}} \Pr\{V\} Q(r_{l(V)}) \tag{8.7}
$$

such that

$$
\sum_{i=1}^{L} \frac{s_{i,j}}{k_i}(r_i - r_{i-1}) \le R_{\max,j} \quad \forall 1 \le j \le M. \tag{8.8}
$$

Optimizing matrix S and $\vec{k}$ in order to maximize the average quality criterion Q is an integer programming problem since values for $s_{i,j}$ and k_i only take integer values and is very complex since the number of parameters tends to get very large. Furthermore, the behavior of Function (8.3) is non-linear and non-continuous. When also the rates for all layers r_i have to be optimized, the problem is a mixed integer programming problem. For now we concentrate on investigating the behavior of fairly simple AMDC cases.

If we constrain again to a system with three layers and three descriptions, each layer i is coded with a different (n_i, k_i) channel code. In this example layer one is (channel) coded with a $(6, 3)$ code, layer 2 with $(6, 4)$ code and layer 3 with $(6, 3)$. The following table shows the distribution of the n codewords over all descriptions:

$$
\begin{array}{ccccc}
 & & S_{\text{asym,6}} & & \text{LEC} \\
 & d_1 & d_2 & d_3 & (n_i, k_i) \\
 & \left[\begin{array}{ccc} 3 & 2 & 1 \\ 2 & 2 & 2 \\ 1 & 2 & 3 \end{array}\right] & & & \begin{array}{c}(6,3)\\(6,4)\\(6,5)\end{array}
\end{array} . \tag{8.9}
$$

This simple example already gives some interesting behavior. If a client only receives description one, he has — according to the channel code — sufficient information to decode layer one. When he receives descriptions two

and three, he could decode layer one as well as layer two. Receiving all three descriptions results in all descriptions being decodable.

8.5 Comparison of results for $M = 3$ and simple quality metric

Let us compare the AMDC method to SDC, layered coding and symmetric MDC. The quality metric we use here is simply counting the number of layers that can successfully be decoded, Furthermore we assume that every layer has a rate $r_i = 1$. Suppose descriptions d_1, d_2, d_3 have loss probabilities of $0.05, 0.1, 0.3$, respectively. For the asymmetric case, on average 2.89 layers can be decoded at a total rate of 5.5 (See Table 8.1). For the symmetric case, the total rate is also 5.5, but on average only 2.55 layers can be decoded. By changing the codes and reallocating the codewords over the descriptions we have adapted to the network conditions, which results in more layers being decodable (and hence in higher quality) and in lower rates. The Layered coding technique where each layer is send over a different channel, only 2.4 layers can be decoded. For a complete view, we also simulated the single description coding case, where all data is sent over the safest channel. This gives a pretty high average of 2.85, but needs much higher rate on this channel. In a rate-constrained case, where description rates R_j are constrained to 1.8, by using AMDC we achieve better error resilience and are able to balance the load over all channels.

Method	R_{total}	$\{R_1, R_2, R_3\}$	n
SDC	3.0	$\{3.0, 0, 0\}$	2.85
Layered	3.0	$\{1.0, 1.0, 1.0\}$	2.40
Symmetric MDC	5.5	$\{1.8, 1.8, 1.8\}$	2.55
Asymmetric MDC	5.5	$\{1.8, 1.8, 1.8\}$	2.89

Table 8.1 — *Comparison of AMDC cases for $M = 3$.*

8.6 Optimization Algorithm

Since we believe there is no efficient analytic solution to the optimization criterion, we developed an algorithm (Alg: 8.5) based on simulated annealing that tries to find a good S matrices. We do this for the simple case using the layer-counting metric and for equal layer rates $r_i = r_j \forall i \neq j$. We try to find a Pareto set S of S matrices, containing non-inferior solutions S using function

$$Pareto(\mathsf{S}, \mathsf{V}, \tilde{k}) : \{S_i \in \mathsf{S}; \hat{Q}(S_i, \vec{k}) > \hat{Q}(S_j, \vec{k}) \quad \wedge \\ R_i > R_j \quad \forall S_j \in \mathsf{S}, i \neq j\}. \tag{8.10}$$

The algorithm runs an arbitrary number of passes. With each pass, the existing set S is extended with multiple random variations of each matrix $S \in \mathsf{S}$. Of this new set the Pareto points are extracted. With each pass i set S_i is superior to S_{i-1}. The algorithm is initialized with a template matrix S which can be a random or structured matrix. The purge step is included to purge solutions that are not required, and can be used to bound the rates. By feeding the algorithm sufficiently large f_0 and $\vec{k}$ the algorithm has enough freedom to find good candidates. By shaping the initial S matrix and k-values towards a reasonable initial solution the optimization process can be improved. In figure 8.6 we show some results of the algorithm. The X-axis shows the total rate $\sum_1^M R_j$, the Y-axis the average Quality under the given conditions. The lines show the Pareto points found in each pass. The upper-left-most line corresponds the the last pass.

$\mathsf{S}_{\mathrm{par}} = EXTEND(M, \mathsf{S}_0, \vec{k}, \mathsf{V}, \vec{p}, Npasses)$

Require: Number of descriptions M, template matrix S_0, $\vec{k}$, V, channel probabilities $\vec{p}$, total number of passes.
Ensure: $\mathsf{S}_{\mathrm{par}} \equiv Pareto(\mathsf{S}_0)$
  for $pass = 1$ to $Npasses$ **do**
    $N \leftarrow N_0/2^{\mathrm{pass}-1}$, $f \leftarrow f_0/2^{\mathrm{pass}-1}$,
    $\mathsf{T} \leftarrow \mathsf{S}_{pass-1}$
    for all $S \in \mathsf{S}_{pass-1}$ **do**
      for $n = 1$ to N **do**
        $\mathsf{T} \Leftarrow S + f \cdot \mathrm{random}_{M \times M}$,
      end for
    end for
    $\mathsf{T} \leftarrow Prune(\mathsf{T})$
    $\mathsf{S}_{pass} \leftarrow Pareto(\mathsf{T}, \mathsf{V}, \vec{k})$
  end for
  $\mathsf{S}_{\mathrm{par}} \leftarrow \mathsf{S}_{Npasses}$

Figure 8.5 — *Algorithm to find a Pareto set of S matrices, which all result in a different rate R_m and Quality Q.*

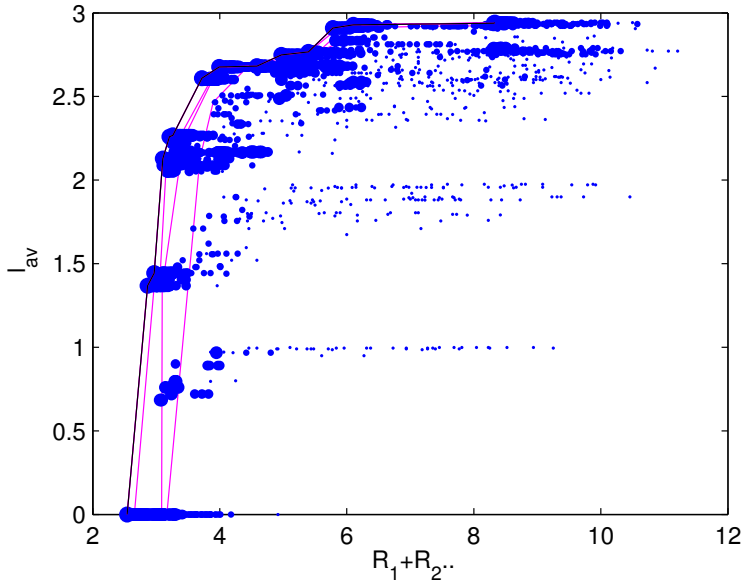


Figure 8.6 — *Example of results of the algorithm (5 Pass, $N_0 = 128$, $f = 16$, $\vec{k} = \{16, 32, 48, 64\}$, $\vec{p} = \{0.05, 0.1, 0.2\}$). The dots correspond to all points found in each pass. The lines connect the Pareto points found in each pass of the algorithm*

Nine

Discussion

9.1 Common Solutions and Results

The video streaming scenarios presented in this thesis have in common that video has to be streamed over an error-prone channel. However, the presented solutions differ in encoding paradigm, the level of cooperation and the type of network. In all cases though, the underlying optimization problem could be reduced to variants of a rate-allocation problem, each demanding a different approach. Another observation is that in all cases, behavior models of the video coder in question are used to find the optima, although the complexity of these model range from RD models—based on variance—to high complexity models taking more source characteristics into account. Although specific coders were used in each scenario for tuning the models, the models are generic enough to apply to other coding methods as long as they are based on a motion-compensated predictive control-loop.

From Chapters 5, 4 and 6, we may generalize the results in the sense that, whenever network behavior is taken into account in the design and in the settings of a video coder, the video quality is increased, as opposed to a worst-case or medium-case approach, especially when the network characteristics are very dynamic.

In Chapter 6 a tight cooperation of video coder and 802.11a MAC layer was established. Where the original MAC was not designed for streaming applications but to maximize average throughput, the modified MAC supported streaming, by making a trade off between small delays and high throughput. Especially when the MAC layer and Video Coder exchange bit rate, delay and packet-loss parameters, the MAC is able to take these into account, while the video coder tries to be resilient to packet losses, resulting in higher average quality.

P2P networks introduce path diversity by offering different overlay net-

works. We can exploit path diversity by transmitting multiple descriptions over different independent paths, the possible losses in the network have limited effect on the quality (Chapters 7 and 8). In this case, a part of the error-resilience comes from the network itself. This 'Betting on more than one horse' paradigm is in fact what makes a P2P network an interesting medium for streaming video to many clients.

9.2 Diversity or Adaptivity

We conclude from our papers that different encoding paradigms combined with different network types with different types of cooperation, enable adaptive and error-resilient video streaming in dynamic networks. In these cases we either need *adaptivity* with cooperation between layers or *diversity* requiring less cooperation, to deal with streaming on error-prone networks. *Adaptivity* emerges when using parametrized implementations with behavior models and a method for finding the optimal settings given the constraints from other layers. Both LC and MDC create *diversity*, in the sense that, more than one description (or layer) is generated, and each description (or layer) is associated with a different quality level and rate. On the network access side, *diversity* is created by, for instance, P2P networks with multiple overlay networks and by an NAL that offers different channels with a different throughput/reliability trade off.

If we summarize the three scenarios in this thesis:

1. Low-latency real-time streaming of video to low-power mobile devices — with full QoS cooperation using ARC negotiations;
2. Low-latency real-time streaming of video over WiFi networks — with limited QoS cooperation using the bottom-up approach;
3. Streaming of video to many clients using P2P networks — using MDC and different channels but with limited network adaptivity,

where each solution scores differently with respect to adaptivity and diversity and in each case a different level of cooperation is used between Video Coder and NAL, we can make the a table as in Table 9.1. For all scenarios the total score is more or less constant. It seems that adaptivity can be traded off with diversity, resulting in having less QoS cooperation. Having less cooperation, does not mean that there is a mismatch between Video Coder and NAL, the introduced diversity ensures that Video Coder and NAL are matched automatically by the behavior of the underlying network.

Scenario	1	2	3
Scenario	***Scores***		
Video Coder Adaptivity	++	+	+
Video Coder Diversity	0	0	++
NAL Adaptivity	+	++	0
NAL Diversity	0	0	++
Level of QoS Cooperation	++	+	0

Table 9.1 — *Scores for each scenario solution w.r.t. Adaptivity and Diversity.*

9.3 Recommendations

The presented ideas and techniques rely on implementations of Single, Scalable and MD Video Coders. For real implementation and use of Layered and MDC video streaming applications on Internet, these encoding technologies still need to become faster in order to achieve the real time constraint. Furthermore, for every new encoder, behavior models are required, or existing ones have to be tuned. Real-time adaptation of these models to follow the real-time source-characteristics is an interesting and useful research topic.

In the field of MDC, especially AMDC deserves more research and attention, because of its general applicability on any sort of network and for its flexibility of generating anything between prioritized and unprioritized descriptions. MDC in general will benefit from having more methods for generating more that two descriptions with good control over the amount of redundancy.

The inter-layer communication advocated in this thesis also deserves more attention. The market of Video Streaming will benefit from having a standard for inter-layer communication and negotiation, in the style of ARC (Chapter 4), although full QoS negotiations may not always be required.

The approaches for P2P streaming presented here are focused on the video coder and the control of it. The P2P network itself and the algorithms for building distribution trees and the algorithms for real time streaming, greatly influence the quality level of the received video for each client. Since streaming P2P networks are still at their infancy, these aspects require more research and attention.

9.4 Outlook

Video streaming to mobile devices will become common, enabled by the higher speeds offered by Universal Mobile Telecommunications System (UMTS), WiFi and follow-ups. The speed increase of these networks also comes with more varying speeds and packet-losses. To deal with these, a QoS system will help. Whether inter-layer QoS will become commonly used is a matter of will in the

Internet community, from both developers and operators. This kind of evolution is maybe more to be expected from open-source developers than from standardization bodies. A more viable option is maybe the 'bottom-up' approach of informing higher layers of the performance.

In the coming years, video streaming over Internet will get a stronger foothold and in some cases replace Television broadcasting. Especially for less-popular content and programs. The bulk of 'long tail' content deserves to be distributed but is more and more left out in favor of big shows, sports events, the news etc [4]. P2P may help in disclosing the long tail content by setting up P2P networks for fans of the relatively unpopular content.

Using MDC on P2P networks is becoming a viable option, when good and standardized MDC encoders and players become available. The demand for Layered Coding becomes higher when more different devices are used for display, ranging from mobiles to high definition television (HDTV), and the rate variation between devices increases (100kbps for mobiles to 20Mbps for ADSL/2 users). The use of the scalable H.264/SVC standard will probably be boosted when fast and free implementations become available on Internet, enabling its use in P2P networks. Since the LC-LEC MDC method is based on Layered Coding, H.264/SVC may significantly boost the use of MDC in streaming applications.

The uprise of P2P networks also shows another development: decentralization. Where content delivery networks (CDNs) have centralized control and are well-organized. P2P networks rely more on self-organization and centralized control. The latter is the more complex option, but has the advantage of having no single-point of failure and not having to rely on a single party. As long as Internet traffic itself is not controlled and censored by governments, P2P enables the freedom of speech and people can start their own Internet Television (TV) station using P2P networks.

Bibliography

[1] R. Ahlswede, "The rate-distortion region for multiple descriptions without excess rate," *IEEE Transactions on Information Theory*, vol. 31, no. 6, pp. 721–726, 1985.

[2] T. Ahmed, A. Mehaoua, R. Boutaba, and Y. Iraqi, "Adaptive packet video streaming over IP networks: a cross-layer approach," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 2, pp. 385–401, 2005.

[3] S. Aign and K. Fazel, "Temporal and spatial error concealment techniques for hierarchical MPEG-2 video codec," in *IEEE International Conference on Communications*, vol. 3, 1995.

[4] C. Anderson, *The Long Tail*. New York: Hyperion, 2006.

[5] J. Apostolopoulos, "Error-resilient video compression through the use of multiple states," in *Proceedings Int. Conf. on Image Processing*, 2000.

[6] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee, "On multiple description streaming with content delivery networks," in *Proc. IEEE INFOCOM*, vol. 50, July 2002, p. 3.

[7] C. Aurrecoechea, A. T. Campbell, and L. Hauw, "A survey of QoS architectures," in *7th Int. Workshop on Quality of Service (IWQoS'99)*, June 1999.

[8] K. Balachandran, S. Kadaba, and S. Nanda, "Channel quality estimation and rate adaptation for cellular mobile radio," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 7, pp. 1244–1256, July 1999.

[9] M. Bechler, H. Ritter, and J. Schiller, "Quality of service in mobile and wireless networks: The need for proactive and adaptive applications," in *Hawaii Int. Conf. on System Sciences (HICSS-33)*, Jan. 2000.

[10] G. L. Bodic, J. Irvine, and J. Dunlop, "Resource cost and QoS achievement in a contract-based resource manager for mobile communications systems," in *Proceedings of Eurocomm*, May 2000, pp. 392–397.

[11] M. Boliek, C. Christopoulos, and E. Majani, "Jpeg2000 part i final draft international standard," ISO/IEC, Tech. Rep., 2002.

[12] J.-C. Bolot and T. Turletti, "Experience with control mechanisms for packet video in the internet," *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 1, pp. 4–15, 1998.

[13] B. Braswell and J. McEachen, "Modeling Data Rate Agility in the IEEE 802.11a WLAN Protocol," in *OPNETWORK 2001*, Mar. 2001.

[14] M. Bystrom and J. Modestino, "Combined source-channel coding for transmission of video over aslow-fading Rician channel," in *Image Processing, Proceedings. International Conference on*, vol. 2, 1998, pp. 147–151.

[15] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: high-bandwidth multicast in cooperative environments," in *Proc. of the 19th ACM symposium on Operating systems principles*, 2003, pp. 298–313.

[16] Y. S. Chan and J. W. Modestino, "Transport of scalable video over cdma wireless networks: A joint source coding and power control approach," in *Proceedings International Conference on Image Processing*, vol. 2, 2001, pp. 973–976.

[17] Y. Chen, K. Yu, J. Li, and S. Li, "An error concealment algorithm for entire frame loss in video transmission," in *Proc. Picture Coding Symposium*, 2004.

[18] C. Cheung and L. Po, "Novel cross-diamond-hexagonal search algorithms for fast block motion estimation," *IEEE Transactions on Multimedia*, vol. 7, no. 1, pp. 16–22, 2005.

[19] A. Chimienti, C. Ferraris, and D. Pau, "A complexity-bounded motion estimation algorithm," *IEEE Transactions on Image Processing*, vol. 11, no. 4, pp. 387–392, 2002.

[20] P. A. Chou, H. J. . Wang, and V. N. Padmanabhan, "Layered multiple description coding," in *Proc. Packet Video Workshop*, Apr. 2003.

[21] G. Côté, S. Shirani, and F. Kossentini, "Optimal mode selection and synchronization for robust video communications over error prone networks." *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 952–968, 2000. [Online]. Available: citeseer.nj.nec.com/263417.html

[22] N. Damera-Venkata, T. Kite, W. Geisler, B. Evans, and A. Bovik, "Image quality assessment based on a degradation model," *Image Processing, IEEE Transactions on*, vol. 9, no. 4, pp. 636–650, 2000.

[23] G. M. Davis and J. M. Danskin, "Joint source and channel coding for image transmission over lossy packet networks," *Proceedings of SPIE*, vol. 2847, p. 376, 1996.

[24] J. del Prado Pavon and S. Choi, "Link adaptation strategy for IEEE 802.11 WLAN via received signal strength measurement," in *IEEE International Conference on Communications, 2003 (ICC '03)*, vol. 2, Anchorage, Alaska, USA, May 2003, pp. 1108–1113.

[25] S. Diggavi, N. Sloane, and V. Vaishampayan, "Asymmetric multiple description lattice vector quantizers," *Information Theory, IEEE Transactions on*, vol. 48, no. 1, pp. 174–191, 2002.

[26] "The DIRAC project," http://www.bbc.co.uk/rd/projects/dirac/.

[27] T. Do, K. A. Hua, and M. Tantaoui, "P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment," in *to appear in the Proc. of the IEEE Int. Conf. on Communications (ICC 2004)*, jun 2004.

[28] A. El Gamal and T. Cover, "Achievable rates for multiple descriptions," *IEEE Trans. Inform. Theory*, vol. 28, no. 6, pp. 851–857, 1982.

[29] W. H. R. Equitz and T. M. Cover, "Successive refinement of information," *IEEE Transactions On Information Theory*, vol. 37, no. 2, pp. 269–275, Mar. 1991.

[30] N. Färber, K. Stuhlmüller, and B. Girod, "Analysis of error propagation in hybrid video coding with application to error resilience," in *Proceedings International Conference on Image Processing*, vol. 2, 1999, pp. 550–554.

[31] B. Girod and N. Färber, "Wireless video," in *Compressed Video Over Networks*, M.-T. Sun and A. Reibman, Eds. Marcel Dekker, 1999, ch. 12.

[32] V. Goyal and J. Kovacevíc, "Generalized multiple description coding with correlating transforms," *IEEE Trans. onf Information Theory*, vol. 47, no. 6, pp. 2199–2224, September 1999.

[33] V. Goyal, "Transform coding with integer-to-integer transforms," *Information Theory, IEEE Transactions on*, vol. 46, no. 2, pp. 465–473, 2000.

[34] ——, "Multiple description coding: compression meets the network," *Signal Processing Magazine, IEEE*, vol. 18, no. 5, pp. 74–93, 2001.

[35] V. Goyal and J. Kovacevic, "Generalized multiple description coding with correlating transforms," *Information Theory, IEEE Transactions on*, vol. 47, no. 6, pp. 2199–2224, 2001.

[36] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2325–2383, Oct. 1998.

[37] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2Cast: peer-to-peer patching scheme for VoD service," in *Proc. of the 12th Int. Conf. on World Wide Web*, 2003, pp. 301–309.

[38] I. Haratcherev, K. Langendoen, I. Lagendijk, and H. Sips, "Hybrid Rate Control for IEEE 802.11," in *ACM International Workshop on Mobility Management and Wireless Access Protocols (MobiWac)*, Philadelphia, PA, USA, Oct. 2004, pp. 10–18.

[39] I. Haratcherev, J. R. Taal, K. Langendoen, R. L. Lagendijk, and H. J. Sips, "Fast 802.11 link adaptation for real-time video streaming by cross-layer signaling," in *Proc. International Symposium on Circuits and Systems*, Kobe, Japan, May 2005, pp. 3523–3526.

[40] ——, "Link adaptation and cross-layer signaling for wireless video-streaming in a shared medium," in *WirelessCom2005*, Maui, Hawaii, June 2005.

[41] I. Haratcherev, J. R. Taal, K. Langendoen, R. Lagendijk, and H. Sips, "Optimized video streaming over 802.11 by cross-layer signaling," *IEEE Communications Magazine*, vol. 44, no. 1, pp. 115–121, Jan. 2006.

[42] R. Holenstein, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, 2001.

[43] G. Holland, N. H. Vaidya, and P. Bahl, "A rate-adaptive MAC protocol for multi-hop wireless networks," in *ACM MOBICOM'01*, Rome, Italy, July 2001.

[44] IEEE, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications for 802.11a and 802.11b," 1999.

[45] A. Kamerman and L. Monteban, "WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band," *Bell Labs Technical Journal*, pp. 118–133, Summer 1997.

[46] B.-J. Kim, Z. Xiong, and W. A. Pearlman, "Low bit-rate scalable video coding with 3-d set partitioning in hierarchical trees (3-d spiht)," *Ieee Transactions On Circuits And Systems For Video Technology*, vol. 10, no. 8, pp. 1374–1387, 2000.

[47] T.-H. Lan and A. Tewfik, "Power optimized mode selection for H.263 video coding and wireless communications," in *IEEE Conference on Image Processing*, 1998.

[48] Q. Li and M. van der Schaar, "Providing adaptive QoS to layered video over wireless local area networks through real-time retry limit adaptation," *IEEE Transactions on Multimedia*, vol. 6, no. 2, 2004.

[49] F. Ling, W. Li, and H. Sun, "Bitplane coding of DCT coefficients for image and video compression," *Proc. SPIE*, vol. 3653, pp. 500–508, 1998.

[50] C. Luna, L. Kondi, and A. Katsaggelos, "Maximizing user utility in video streaming applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 2, pp. 141–148, 2003.

[51] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. LeGall, *MPEG Video Compression Standard*. International Thompson Publishing, 1996.

[52] K. Nahrstedt, H. H. Chu, and S. Narayan, "Qos-aware resource management for distributed multimedia applications," *Journal of High Speed Networks*, vol. 7, pp. 229–57, 1998.

[53] A. Ortega and M. Khansari, "Rate control for video coding over variable bit rate channels with applications to wireless transmission," in *Proceedings of the 2nd IEEE International Conference on Image Processing (ICIP)*, vol. 3. IEEE Computer Society Washington, DC, USA, Oct. 1995.

[54] L. Ozarow, "Source-Coding Problem with Two Channels and Three Receivers," *Bell Syst. Tech. Journal*, vol. 59, no. 10, pp. 1909–1921, 1980.

[55] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient peer-to-peer streaming," in *Proceedings of the 11th IEEE International Conference on Network Protocols*. IEEE Computer Society, 2003, p. 16.

[56] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr, "Chainsaw: Eliminating trees from overlay multicast," *Proc. IPTPS*, 2005.

[57] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "A measurement study of the bittorrent peer-to-peer file-sharing system," Delft University of Technology, Tech. Rep. PDS-2004-007, Apr. 2004.

[58] J. A. Pouwelse, J. R. Taal, R. L. Lagendijk, D. H. J. Epema, and H. J. Sips, "Real-time video delivery using peer-to-peer bartering networks and multiple description coding," in *Proceedings International Conference on Systems, Man and Cybernetics*, The Hague, The Netherlands, 2004, pp. 4599–4605.

[59] R. Puri and K. Ramchandran, "Multiple description source coding through forward error correction codes," in *Proc. Asilomar Conference on Signals, Systems and Computers*, Asilomar, CA, Oct. 1999.

[60] K. Ramchandran, A. Ortega, and M. Vetterli, "Bit allocation for dependent quantization with applications to multiresolution and MPEG video coders," *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 533–545, Sept. 1994.

[61] G. Reyes, A. R. Reibman, and S. F. Chang, "A corruption model for motion compensated video subjected to bit errors," in *Proceedings Packet Video Workshop*, 1999.

[62] K. Rijkse, "H.263: Video coding for low-bit-rate communication," *IEEE Communications Magazine*, vol. 34, pp. 42–45, 1996.

[63] A. Said and W. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 243–250, June 1996.

[64] H. Schwarz, D. Marpe, T. Schierl, and T. Wiegand, "Combined scalability support for the scalable extension of H.264/AVC," *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pp. 446–449, 2005.

[65] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*. Urbana: The University of Illinois Press, 1964.

[66] J. Shapiro, D. Center, and N. Princeton, "Embedded image coding using zerotrees of wavelet coefficients," *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, vol. 41, no. 12, pp. 3445–3462, 1993.

[67] S. Sheu, K. A. Hua, and W. Tavanapong, "Chaining: A generalized batching technique for video-on-demand systems," in *Proc. of the International Conference on Multimedia Computing and Systems*, Ottawa, Canada, June 1997.

[68] J. Shin, J. Kim, and C. Kuo, "Quality-of-service mapping mechanism for packet video indifferentiated services network," *IEEE Transactions on Multimedia*, vol. 3, no. 2, pp. 219–231, 2001.

[69] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *Signal Processing Magazine, IEEE*, vol. 18, no. 5, pp. 36–58, 2001.

[70] K. Stuhlmüller, N. Färber, M. Link, and B. Girod, "Analysis of video transmission over lossy channels," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 1012–1032, 2000.

[71] K. Stuhlmüller, N. Färber, and B. Girod, "Analysis of video transmission over lossy channels," *IEEE Journal on Selected Area in Communications*, vol. 18, no. 6, pp. 1012–1032, June 2000.

[72] G. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, 1998.

[73] J. R. Taal, K. Langendoen, A. van der Schaaf, H. W. van Dijk, and R. L. Lagendijk, "Adaptive end-to-end optimization of mobile video streaming using QoS negotiation," in *ISCAS, special session on Multimedia over Wireless Networks*, vol. I, Scottsdale, AZ, May 2002, pp. 53–56.

[74] J. R. Taal, J. A. Pouwelse, and R. L. Lagendijk, "Scalable multiple description coding for video distribution in p2p networks," in *Proc. Picture Coding Symposium*, San Francisco, CA, 2004.

[75] J. R. Taal, Z. Chen, Y. He, and R. L. Lagendijk, "Error resilient video compression using behavior models," *EURASIP Journal on Applied Signal Processing*, no. 2, pp. 290–303, Feb. 2004.

[76] J. R. Taal and R. L. Lagendijk, "Fair rate allocation of scalable multiple description video for many clients," in *Proc. of SPIE, Visual Communications and Image Processing*, vol. 5960, July 2005, pp. 2172–2183.

[77] ——, "Asymmetric multiple description coding using layered coding and lateral error correction," in *Proc. twenty-seventh Symposium on Information Theory in the Benelux,*. Noordwijk: Werkgenootschap Informatie- en Communicatietheorie, June 2006, pp. 39–44.

[78] A. Tanenbaum, *Computer Networks*. Prentice Hall PTR, 2002.

[79] D. A. Tran, K. Hua, and T. Do, "A peer-to-peer architecture for media streaming," *IEEE Journal on Selected Areas in Communications, Special Issue on Advances in Service Overlay Networks*, vol. 22, no. 1, pp. 121–133, jan 2004.

[80] E. Tuncel and K. Rose, "Additive successive refinement," *Information Theory, IEEE Transactions on*, vol. 49, no. 8, pp. 1983–1991, 2003.

[81] T. Ue, S. Sampei, N. Morinaga, and K. Hamaguchi, "Symbol rate and modulation level-controlled adaptive modulation/TDMA/TDD system for high-bit-rate wireless data transmission," *IEEE Transactions on Vehicular Technology*, vol. 47, no. 4, pp. 1134–1147, Nov. 1998.

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

[83] V. Vaishampayan, N. Sloane, and S. Servetto, "Multiple-description vector quantization with lattice codebooks: design and analysis," *Information Theory, IEEE Transactions on*, vol. 47, no. 5, pp. 1718–1734, 2001.

[84] V. A. Vaishampayan and J. Domaszewicz, "Design of entropy constrained multiple–description scalar quantizers," *IEEE Trans. on Information Theory*, vol. 40, no. 1, pp. 245–250, Jan. 1993.

[85] C. van den Branden Lambrecht and O. Verscheure, "Perceptual quality measure using a spatiotemporal model of the human visual system," *Proc. SPIE*, vol. 2668, pp. 450–461, 1996.

[86] A. van der Schaaf and R. L. Lagendijk, "Independence of source and channel coding for progressive image and video data in mobile communications," *Visual Communications and Image Processing (VCIP2000)*, vol. 4067, pp. 187 – 197, June 2000.

[87] A. van der Schaaf, K. Langendoen, and R. L. Lagendijk, "Design of an adaptive interface between video compression and transmission protocols for mobile communications," in *11th Packet Video Workshop (PV-2001)*, Kyongju, Korea, Apr. 2001. [Online]. Available: http://www.ubicom.tudelft.nl/docs/PV2001submit.pdf

[88] M. van Der Schaar, "Cross-layer wireless multimedia transmission: challenges, principles, and new paradigms," *IEEE Wireless Communications Magazine*, vol. 12, no. 4, pp. 50–58, 2005.

[89] A. van der Vegt, "Auto Rate Fallback Algorithm for the IEEE 802.11a Standard," Utrecht University, Tech. Rep., 2002.

[90] H. van Dijk, K. Langendoen, and H. Sips, "ARC: a bottom-up approach to negotiated QoS," in *3rd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2000)*, Monterey, CA, Dec. 2000, pp. 128–137.

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

[92] A. A. Webster, C. T. Jones, M. H. Pinson, S. D. Voran, and S. Wolf, "An objective video quality assessment system based on human perception," in *Human Vision, Visual Processing, and Digital Display IV.*, San Jose, CA, February 1993, pp. 15–26.

[93] P. H. Westerink, J. H. Weber, D. E. Boekee, and J. W. Limpers, "Adaptive channel error protection of subband encoded images," *IEEE Transactions on Communications*, vol. 41, no. 3, pp. 454–459, 1993.

[94] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 688–703, 2003.

[95] T. Wiegand, G. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 560–576, 2003.

[96] Q. Zhang, W. Zhu, and Y. Zhang, "End-to-end QoS for video delivery over wireless internet," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 123–134, 2005.

Summary

Although mostly hidden, video compression plays a large role in many of our every day activities such as watching TV, web-browsing and mobile telephony. Every device with a matrix screen will be used to display moving pictures.

In fact, the essence of compression does not change for TV or digital video or for mobile communications. Only the level of compression changes, and the format and capabilities of the display. The required level of compression might even change when the connection is prone to noise, failure and varying numbers of users. In that case, the video transmission system should be able to adapt to the changing conditions, such that the user will still receive the best quality given the circumstances. This adaptability is in fact the *leitmotif* throughout this thesis.

The thesis is split up in two parts. The first part discusses a framework for real-time video streaming over an infrastructure that is prone to disruptions. Especially wireless networks and peer-to-peer networks suffer from disruptions as noise, peer-failure, packet loss and bandwidth variations due to varying numbers of users. The framework has a multi-level structure with an underlying network, a network adaptation layer, the video compression layer and an application. Every layer in this structure has a specific task. Besides exchanging the video data, the layers also exchange status information that express the current status of for instance the network, the quality of the video or the average processing and transmission delay. By exchanging this information, the individual layers are able to adapt to changes, for instance by applying more compression.

This layered structure serves two purposes. First of all, it could be a basis for an implementation where the layers cooperate to guarantee ti the best video quality under the given circumstances. Secondly, the framework gives better understanding of which parameters play the most important role when adapting the video compressor and other layers.

Then the thesis discusses three different techniques to design the video compressor that fits in the presented framework. The first technique employs a parametrized video compressor. By using behavior models, the behavior of the video compressor can be predicted based on knowledge of the circumstances

as given by the other layers. Consequently, the optimal settings for the video compressor can be found. The accuracy of these models is important, since a balance has to be found between the accuracy and the complexity and predictive value of the models.

The second technique is based on a layered video compressor. Multiple video layers are generated such that the video quality can be increases when multiple layers are received by the user. It is necessary that the lower video-layers are also received, otherwise the decoding will fail. Layered compression also plays a role in a new technique for multiple description coding (MDC), that is presented in the second part.

The third technique is based on MDC where also different descriptions are generated. The difference is that the resulting quality depend on the number of descriptions. It is not required that lower-level descriptions are received in order to successfully decode the video. Since MDC is used in two papers in the second part, a more thorough discussion on symmetric and asymmetric MDC is presented.

In the second part, five different articles are presented. The first article discusses an experiment in which the layered framework with cooperating video and network layers is employed in a dynamic and error-prone network. Following this, an article is presented that uses a different type of video compressor, and discusses the accompanying behavior models and optimization of the settings. The subsequent article discusses the adaptive video compression system that is used in a wireless 802.11 network, where also the network layer is able to adapt to the circumstances. By cooperation between the video compressor and the network layer, the system is able to respond fast to changing conditions.

The last two articles discuss the usage of MDC for transmitting real-time video over peer-to-peer networks where the descriptions can be sent independently. One article handles the optimization of the compressor settings for a large user group, such that all users receive acceptable quality. The other article discusses a new method for MDC that is able to adapt to unbalanced channels. Whenever different channels have unequal capacity and reliability, asymmetric MDC offer a better solution than layered coding, symmetric MDC and single description coding.

Samenvatting

Video compressie speel een grote, alhoewel grotendeels verborgen, rol in veel van onze dagelijkse activiteiten, zoals TV-kijken, Web-browsen en het gebruik van de mobiele telefoon. Ieder apparaat met een matrix-scherm zal worden benut om bewegend beeld weer te geven, waardoor ook video compressie toegepast moet worden.

In feite verschilt het principe van compressie niet voor TV of video en voor mobiele telefoons. Het verschil zit hem in de mate van compressie en het formaat van het weergavescherm. De benodigde mate van compressie kan zelfs fluctueren als de verbinding onderhevig is aan ruis, uitval en wisselende aantallen gebruikers. In dat geval zal het videoverzendsysteem zich moeten aanpassen aan de veranderde omstandigheden, dusdanig dat de kijker de best mogelijke kwaliteit te zien krijgt, die onder die omstandigheden mogelijk zijn. Het is deze aanpasbaarheid die als een rode draad door dit proefschrift loopt.

Dit proefschrift is opgedeeld in twee delen. Allereerst wordt een raamwerk besproken voor live video verzending over een infrastructuur die onderhevig is aan verstoringen. Met name draadloze verbindingen en zogenaamde peer-to-peer verbindingen lijden aan verstoringen zoals uitval, ruis en belastingsvariatie. Vervolgens worden drie technieken besproken voor het ontwerp van de video compressor binnen dit raamwerk. In het tweede deel worden diverse artikelen gepresenteerd die allemaal in bepaalde mate uitgaan van dit raamwerk en diverse aspecten belichten van het ontwerp van de video compressor en de optimalisatie van de instellingen ervan.

Het raamwerk biedt een ontwerpstructuur voor een videoverzendsyteem dat zich kan aanpassen aan veranderende netwerkomstandigheden. Het is in eerste instantie gericht op ware-tijd verzending van video, zodat het tijdverschil tussen beeld-opname en weergave aan de kijker gelimiteerd is. Het raamwerk volgt een gelaagde structuur met een onderliggend netwerk, een netwerk-adaptatie laag een video compressie laag een en applicatie. Iedere laag heeft in die structuur zijn eigen taak. Naast de beeldgegevens die worden overgedragen tussen die lagen, worden ook gegevens uitgewisseld die uitdrukking geven aan de huidige omstandigheden zoals de kwaliteit van het netwerk, de kwaliteit van de video en de gemiddelde vertraging die optreed door ver-

werking en verzending. Doordat de lagen deze gegevens uitwisselen zijn ze in staat in te spelen op veranderingen in de situatie, bijvoorbeeld door meer compressie toe te passen.

Deze gelaagde structuur dient twee doelen. Ten eerste kan het als basis dienen voor een implementatie waarin de verschillende lagen werkelijk samenwerken om de video verzending zo goed mogelijk te verzorgen onder wisselende omstandigheden. Ten tweede biedt het raamwerk inzicht in welke parameters de belangrijkste rol spelen bij het instellen van de video compressor en andere lagen.

Vervolgens behandelt het proefschrift drie verschillende technieken om een video compressor te ontwerpen, die is in te passen in het raamwerk. De eerste techniek gebruikt een video compressor waaraan verschillende parameters ingesteld kunnen worden. Met behulp van gedragsmodellen kan het gedrag van deze compressor voorspeld worden op basis van de omstandigheden zoals die door de ander raamwerk-lagen worden doorgegeven. Vervolgens kan de optimale instelling gevonden voor de video compressor. De accuraatheid van de gedragsmodellen speelt hier een belangrijke rol, hoewel een afweging gemaakt zal moeten worden tussen deze accuraatheid en de complexiteit en voorspellende kracht van de modellen.

De tweede techniek is gebaseerd op een gelaagde video compressor. Hier worden verschillende videolagen gegenereerd zodat de kwaliteit verhoogd kan worden zodra meer lagen ontvangen worden door de kijker. Het is dan wel noodzakelijk dat de onderliggende videolagen ook steeds ontvangen worden. Deze methode biedt de mogelijkheid om het aantal te versturen lagen te laten afhangen van de beschikbare capaciteit op het netwerk. Gelaagde compressie speelt ook een belangrijke rol in een nieuwe techniek voor meervoudige beschrijvingscodering die wordt gepresenteerd in het tweede deel van het proefschrift.

De derde techniek is gebaseerd op de meervoudige beschrijvingscodering (MBC). Hier worden ook verschillende beschrijvingen gegenereerd. Met dit verschil, dat de kwaliteit zoals de kijker die te zien krijgt afhangt van de hoeveelheid beschrijvingen die hij ontvangt. Er is dan geen noodzaak om ook onderliggende video beschrijvingen te ontvangen. Aangezien MBC een belangrijke rol speelt in twee artikelen in het tweede deel van het proefschrift, wordt dieper ingegaan op theorie van symmetrische en asymmetrische MBC.

In het tweede deel wordt een vijftal artikelen gepresenteerd. Het eerste artikel bespreekt een experiment waarin het gelaagde raamwerk met samenwerkende lagen wordt toegepast op een veranderend netwerk. Vervolgens wordt een artikel gepresenteerd dat een ander type video compressor gebruikt en behandelt de bijbehorende gedragsmodellen en optimalisatie van de compressor instellingen. Vervolgens word een artikel gepresenteerd dat de adaptieve video compressor gebruikt in een draadloze 802.11 netwerk, waarbij ook de netwerklaag zich kan aanpassen aan de omstandigheden. Door de samenwerking

tussen video compressor en netwerk adaptatie laag, kan zeer snel gereageerd worden op de veranderende omstandigheden.

De laatste twee artikelen behandelen het gebruik van meervoudige beschrijvingscodering voor het verzenden van ware-tijd video over peer-to-peer netwerken, waarbij de beschrijvingen onafhankelijk over het netwerk verzonden worden. Het ene artikel bespreekt de optimalisatie van de compressor instellingen voor een grote groep kijkers zodat alle gebruikers een acceptabele kwaliteit ontvangen. Het andere artikel bespreekt een methode voor meervoudige beschrijvingscodering dat zich aanpast aan ongebalanceerde kanalen. Zodra verschillende kanalen een verschillende capaciteit en betrouwbaarheid hebben, is de asymmetrische MBC een betere oplossing dan gelaagde codering, symmetrische MBC en traditionele enkele beschrijvingscodering.

Curriculum Vitae

Jacco Taal was born in Hoek van Holland (Rotterdam), The Netherlands on September 2, 1976. He obtained a HAVO diploma in 's-Gravenzande in 1993. After successfully finishing the Propaedeuse at Technische Hogeschool Rijswijk in 1994, he was accepted in the same year at Delft University of Technology for the Electrical Engineering study.

During his studies he ran a small IT company. In 1998 he worked for one whole year in a Dutch company as a programmer to solve Millennium Problems in their software. He graduated in 2001 in the UBICOM project at the Information and Communication Theory Group, on 'Error Resilient Video Streaming over Wireless Transmission Channels'.

After Working in UBICOM as a researcher, he commenced the Ph.D. track in 2003 at the Information and Communication Theory Group. During this track he was involved in three different interdisciplinary research programs, namely UBICOM, CACTUS and ISHARE. In these projects, cooperation with researchers of different background (Computer Science, Physics, Industrial Design), played an important role.

Besides this research he was involved in guiding master graduation students, supervising lab work in the course on Multimedia Compression.

Since December 2007, he has been working as business developer in the Tribler Valorization Project, in Delft. In this new function he has to combine technical supervision of development and managing the external contacts with potential clients of Tribler.

He is a member of Advanced School for Computing and Imaging. He plays various sports such as korfbal, squash, diving and skiing. In his korfbal club, he has been active in the board, organizing events and bookkeeping. Other interests include photography, reading, architecture, astronomy and traveling.