QoS framework for video streaming in home networks

QoS framework for video streaming in home networks

Dmitri Jarnikov

QoS framework for video streaming in home networks

Printed by: Eindhoven University Press, The Netherlands.
Cover design: Inga Douhaya.

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Jarnikov, Dmitri

QoS framework for video streaming in home networks / by Dmitri Jarnikov. -
Eindhoven : Technische Universiteit Eindhoven, 2007.
Proefschrift. - ISBN 978-90-386-1090-0
NUR 980
Subject headings: video technology / image compression / data communication / computational complexity / consumer electronics / real-time computers
CR Subject Classification (1998): E.4, C.2, G.3

The work in this thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics).

QoS framework for video streaming in home networks

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de Rector Magnificus, prof.dr.ir. C.J. van Duijn, voor een commissie aangewezen door het College voor Promoties in het openbaar te verdedigen op maandag 27 augustus 2007 om 16.00 uur

door

Dmitri Jarnikov

geboren te Minsk, Wit-Rusland

Dit proefschrift is goedgekeurd door de promotor:

prof.dr.ir. P.H.N. de With

Copromotoren:
dr. J.J. Lukkien
en
dr. P.D.V. van der Stok

Contents

List of Tables

List of Figures

1

Introduction

1.1 Research area and problem statement

There is currently a major technological revolution taking place with regard to the availability of digital video on the markets for personal computers (PC), consumer electronics (CE) and mobile devices. Consumers are moving from analog TV reception and analog storage to digital storage and, more importantly, to electronic management of video. Direct Broadcast Satellites (DBS) transmit digital signals; cable service providers are installing infrastructures with digital capability and there is a substantial growth in the availability of digital video channels [1]. Tiny video clips and movie trailers can be watched on the Internet, while video-on-demand enables entire movies to be shown whenever and wherever the user requires. Personal computers and handheld devices are joining TV sets as facilities for showing videos. As more and more of the devices in consumers' houses have the facility to acquire, process and display video, the interconnectivity of the devices has set an immense challenge.

Households have a fully developed infrastructure for analog video. This infrastructure, however, cannot be extended in a cost-effective way to serve the emerging devices that work with digital video. For example, in order to watch a video from different devices, a physical connection (cable) is required between the devices and a TV set. This involves a huge amount of cabling in the house and may require the user to have specific expertise if devices need an intermediary to connect (as in

the case of a handheld or mobile device connected via a PC to a TV set). There are currently a variety of research and development projects in progress that focus on creating an interoperable digital video infrastructure inside the house. The first step towards a solution is to create and deploy an in-home network that can be used for video distribution.

1.1.1 In-home networks

An in-home network [1] provides a convenient means for connecting together different consumer electronics devices – from DVD players and TV sets to PCs and PDAs. The introduction of new pervasive computing devices, such as PDAs, handheld computers and smart phones that allow users access to multimedia information, means that home networks have to be partly wireless.

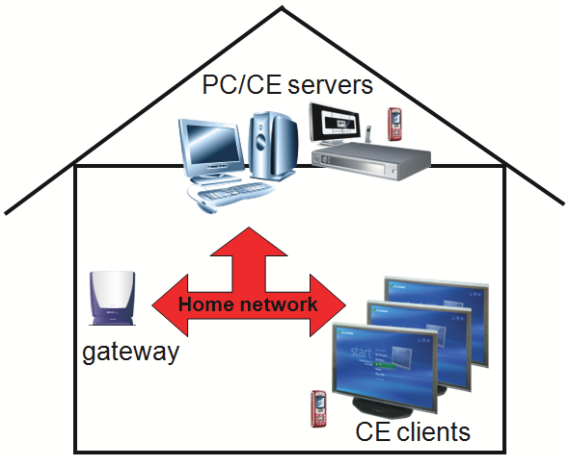

Figure 1.1. Home network.

Figure 1.1 shows an overview of a home network. The network is made up of a mixture of interconnected PC and CE devices. The devices are subdivided into servers and clients according to their capacity for handling video: clients process and display the video, whilst servers distribute the video and can also process and sometimes even display. The video is distributed directly or via a gateway, which is also responsible for connecting the home network to the Internet. The gateway is seen by most CE manufacturers as a powerful PC-based solution.

[1] For simplicity, in this and the following chapters, we use *home network* as a shorthand for *in-home network*.

1.1.2 Problem statement

In general, there are two types of video delivery: video streaming and video delivery by download. Video streaming is the real-time transmission of video to a client device that enables simultaneous delivery and playback of the video. In effect, video streaming splits the video into parts, transmits these parts in succession and enables the receiver to decode and play back the video as the parts are received, without having to wait for the entire video to be delivered. In video streaming there is usually a short delay between the start of the delivery and the beginning of playback at the client. Video streaming also has low storage requirements since only a small portion of the video is stored at the client at any point in time. This is in contrast to file download, where the entire video has to be delivered and stored on the client device before playback can begin. Unlike conventional applications, video streaming generally requires a continuous bandwidth guarantee as well as stringent bounds on delays and jitters [2].

Wireless streaming environments present many challenges for the system designer. Wireless streaming systems are limited by wireless bandwidth and client resources. Video makes the most stringent requirements on the network – a successful video streaming system has to combine limited latency with high capacity and low bit error rates. Wireless network bandwidth is scarce because of its shared nature and the limitations in the availability of wireless spectrum [2]. Even though wireless media can cope with occasional transmission errors, their performance can be devastated by bursts thereof [3]. Client resources are often limited in practical terms by power constraints and by display, communicational and computational capabilities. Different devices have different processor/memory potential, so not every device will necessarily be capable of processing all video data that is sent by the server.

A successful wireless video streaming system must be able to stream video to heterogeneous devices over erroneous communication links with a fluctuating bandwidth. To achieve a high level of acceptability of wireless multimedia (in particular wireless video), several key requirements need to be satisfied to ensure a reliable and efficient transmission:

- easy adaptability to wireless bandwidth fluctuations caused by co-channel interference, multipath fading, mobility, handoff, competing traffic, etc.;
- robustness to partial data losses caused by the packetization of video frames and high packet error rate; and
- support for heterogeneous clients with regard to their access bandwidths, computing capabilities, buffer availabilities, display resolutions and power limitations.

The CE manufactures and service providers are among the major stakeholders

in the deployment of a wireless video streaming solution on the market. Their main motivation is to make profit by introducing a new system and, at the same time, to minimize the cost of manufacture. A solution must, therefore:

- reuse the existing devices and software systems as much as possible,
- use legacy devices,
- comply with existing standards.

An additional requirement that originates from the real-time properties of the developed system is low start-up latency. Start-up latency is defined as the maximum time from the start of the streaming to the time video playback starts at the display.

1.2 Domain analysis

This section gives an overview of the approaches, methods and techniques that exist in the formulated research area. An introduction into the domains that are discussed in the thesis can be found in Appendices A and B.

1.2.1 Video adaptation for network

A network-aware application, i.e. an application that can deal with changes in the network environment [4], should adjust its behavior in response to network performance variations. Since transmission rate is dictated by channel conditions, problems arise if the available transmission rate is lower than the video bit rate. Video content adaptation techniques enable the video to be adapted [2] to suit the varying network conditions. These techniques are *simulstore*, *stream switching*, *frame dropping*, *transcoding*, *scalable video coding*, and *multiple descriptive coding*. The first five techniques aim at adaptation of the video stream to the bandwidth limitations, the *multiple descriptive coding* is a technique that improves robustness of the transmission. Further categorization of the techniques is given in Table 1.1.

Adapting to the bandwidth. Static.

The *simulstore* solution is based on simultaneous storage on the sender of different streams with different spatial resolutions, temporal resolutions and signal-to-noise ratio levels. When a sender starts streaming it chooses an appropriate stream according to present network conditions. The advantages are optimum coding efficiency for each individual stream, easy selection of the appropriate stream at the sender side and low complexity at the sender and client [5]. A major disadvantage is that the stream is chosen at the start of a transmission and the sender cannot

[2] The term *adapted* equally relates to dropping parts of the data during the transmission as well as to decreasing the amount of data that is prepared to be streamed.

Aim	**Technique**	**Type**
adapting to the bandwidth	*simulstore*	static
	stream switching	static/ dynamic
	scalable video coding	static/ dynamic
	frame dropping	dynamic
	transcoding	dynamic
improving robustness	*multiple descriptive coding*	

Table 1.1. Video content adaptation techniques. The two basic categories are: adaptation to the bandwidth limitations and robustness improvement. The adaptation to the bandwidth is sub-categorized into static/dynamic or mixed types. 'Static' means that the technique pre-processes video data before the streaming starts. 'Dynamic' means that an adaptation of the takes place during the streaming as a reaction of bandwidth changes.

change it if network conditions are changed. Thus, the choice of bit rate for a stream is always based on a worst-case analysis, which leads to poor bandwidth utilization.

Adapting to the bandwidth. Static/dynamic.

Stream switching is an extension of the *simulstore* approach which allows another stream to be selected if the transmission conditions have changed. Stream switching offers an additional advantage to the simulstore solution – it allows adaptation to a varying transmission capacity. Furthermore, also a transition to another media format is possible by simply selecting another stream [5]. However, the reaction to a change in network conditions is slow. The change delay depends on the distance to the next switch point [3] and the number of frames that are currently in the sender buffer. On average the delay ranges from $1 - 2$ frames to $10 - 15$ frames, which is unacceptable for a wireless environment because of the frequent bandwidth fluctuations. Also, the long time bandwidth variation may lead to poor bandwidth usage, as it is impossible to have streams with all possible requested bit rates.

A large part of the research activities carried out into video streaming are concentrated on the use of *scalable video coding* that describes the encoding of video frames into multiple layers, including a base layer (BL) of relatively low-quality

[3] Special switching frames are used for switching to another stream

video and several enhancement layers (ELs) that contain increasingly more video data to enhance the base layer and thus give rise to video of increasingly higher quality [6]. The principles of *scalable video coding* are described in Appendix A together with the standard coding techniques. Examples of proprietary scalable video coding schemes can be found in [7, 8, 9, 10].

An adaptive priority-based selective repeat transmission scheme for transmitting scalable video over a single-hop lossy wireless link is proposed in [11]. The proposed on-line priority-based scheme prioritizes all layers of various frames of a group of pictures and transmits them over a lossy wireless link. Although the scheme achieves good results with respect to video quality, it requires a substantial control effort in the transport layer.

A real-time video transmission scheme, which is based on scalable noncausal predictive codec with vector quantization and conditional replenishment [7] and is capable of providing spatial and temporal scalabilities, is described in [12]. The approach eliminates error propagation by combining bandwidth adaptability with error concealment. The scheme produces a relatively constant visual quality in real-time streaming over wireless networks.

The construction of a model that takes into account the frame dependencies of scalable video streams and analysis of the performance of different packet-dropping mechanisms is described in [13]. The findings show that scalable video combined with the priority dropping mechanism gives rise to a higher throughput, lower delay and lower delay jitter for multimedia transmission.

In [14], authors propose a new scalable video transmission scheme which does not require major changes in network protocols. In the proposed scheme frames are dropped dynamically either by the sender or by the network, depending on the level of network congestion. The scheme is based on encapsulation of video frames with priority information which is used to allow the network to drop frames during congestion.

The system proposed in [15] provides unequal error protection for the layers of a scalable video. Results show that the picture quality of a streamed video degrades gracefully as the packet loss probability of a connection increases.

Additional research that have been reported in the literature for adaptive streaming of scalable video described in [16, 17, 18].

Adapting to the bandwidth. Dynamic.

Frame dropping is one of the major techniques for rate adaptation to bandwidth variations in video streaming applications. Efficiency and simplicity are the major reasons that the method is widely used. Frame dropping, however, is undesirable when it happens at high frequency and especially when two or more frames are dropped consecutively. In the wireless environment, however, the bandwidth may

fluctuate with a large amplitude, which forces a number of frames being dropped in a short period of time. This causes motion judder since the dropped frames usually are replaced by replaying previous frames, which, in turn, is very annoying to viewers.

Transcoding is the transformation of media from one format to another (e.g. from MPEG-2 to MPEG-4) or the transformation of media within the same media format (e.g. change of frame rate, bit rate or image resolution) [19, 20, 21, 22, 23]. Transcoding is extremely efficient for handling long-term bandwidth variations. However, for short drops in bandwidth the reaction delay is in the order of a few frames (a number of frames could still be in a sender buffer plus a transcoder needs at least one frame to adapt rate control). Also, adaptation of the media in the network is not possible. Some examples of the transcoding solutions for video streaming are described in [24, 25, 26, 27].

Improving robustness.

Multiple Description Coding (MDC) 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 [28, 29, 30]. In general some amount of redundancy has to be added in order to increase the error resilience and to enhance the gracefulness resilience. There are methods of symmetric MDC where each description is equally important and equivalent (similar but not the same) and, more promising, asymmetric MDC where descriptions that are not equally important and may be prioritized (so, having description *one* may give better quality than having only description *two*). A general framework based on the method of asymmetric *multiple description coding* (AMDC) is presented in [28]. The AMDC encoder fits the network conditions and in many cases outperforms single description coding, symmetric MDC, layered coding and in some cases layered coding with unequal error protections [28].

A combination of *scalable video coding* and *multiple descriptive coding* is possible with descriptions are formed using MDC with forward error correction building on a scalable video coder. The advantage of the approach, as described in [31], is that video streaming becomes robust against bandwidth variations or failing network nodes, which in both cases cause random chunks of data to be unavailable to a video decoder.

1.2.2 Multimedia processing on terminals

Multimedia processing on a receiving device (terminal) represents another field of research relating to video streaming. Sometimes a device is not capable of processing the whole video that is sent by the sender, so one of two approaches has to be applied: either the amount of video data that is being sent to the device is scaled

down or the complexity of the video processing is reduced. The first approach simply involves video content adaptation, which is described in the previous section. The substantial difference is that the content adaptation, which is performed at the sender side, is the result of feedback from the receiving device to the sender. The second approach addresses terminal resource management.

Terminal Resource management

Scalable algorithms [32] have become popular enablers of dynamic resource management, where the worst-case analysis is substituted with estimates for average load. In general, a scalable algorithm is an algorithm that allows a trade-off between resource usage and the output quality of the algorithm. A scalable video algorithm (SVA) can be manipulated via its internal settings to produce a video stream of variable quality. The basic idea behind SVA is shown in Figure 1.2. The

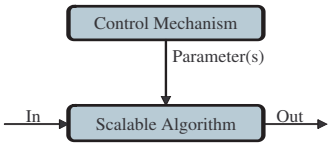

Figure 1.2. Scheme of a scalable algorithm.

control mechanism influences the behavior of the algorithm by means of a set of parameters, taking into account the present state of the system and, in particular, the resource availability.

A method for regulating the varying computation load of a scalable MPEG decoder is described in [33]. The decoding quality of a frame is scaled in accordance with the estimated complexity of the frame processing and the resources required for the processing. The method optimizes the output quality of individual frames.

The work described in [34] optimizes the overall quality of a video whilst changing the quality of the video processing in order to fit in with the resource limitations of the terminal. The approach balances different QoS parameters: picture quality, deadline misses and quality changes.

1.2.3 Adaptable multimedia streaming

There are a number of solutions [35, 36] for using feedback control to adapt dynamically to the amount of resources available over the whole video delivery and processing chain, including sending video over the network, receiving it from the network, decoding, and displaying the decoded video. The framework described in [35] adapts video content on the basis of the feedback information from the receiver with respect to the observed network conditions and the current resource availability on the device. The adaptation of the video is based on frame-rate reduction [36].

1.3 Focus of this thesis

To outline briefly the application domain in which the work described in this thesis is to be executed, we show a simplified example of a home network. Figure 1.3

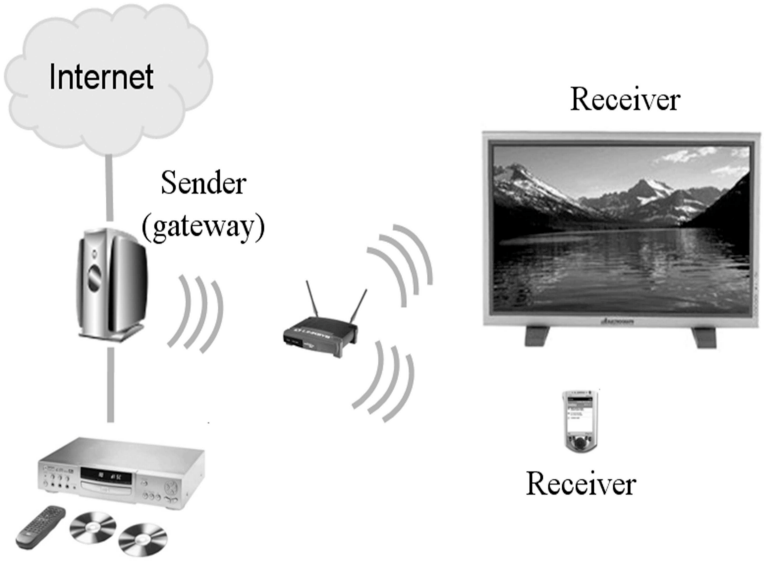

Figure 1.3. Home network view used in this thesis.

shows an example of a video streaming system that consists of a set of receivers (TV and PDA) wirelessly connected to a sender (gateway). The gateway is connected to the outside world via an Internet connection. This connection is used to receive video content either in real time (e.g. video-on-demand or live broadcasting) or by downloading a video and storing it for use at a later stage. In addition, the gateway obtains video content from the DVD player, which serves as an in-home video content supplier. The gateway distributes content via the wireless network to TV and/or PDA. Since the available bandwidth may fluctuate over time, for example due to signal interference or because the channel is being used by another application, the gateway should adapt the video signal continuously to ensure the best possible video quality on terminals.

We recognize long- and short-term bandwidth variations on a wireless link. A long-term bandwidth change can be caused, for example, by one or more new applications that use the same link or by changing the distance between a sender (or an access point) and a receiver. A short-term bandwidth change usually occurs as the result of interference.

In this thesis we use a simple sender-receiver model. A sender is a powerful device that provides media access to video data for all receivers in the net. A sender

can also pass on the content received in real time (applications for broadcasting sporting events, for example). A receiver is a CE device. It is very important that the costs are low for these devices, which means low processing power and minimum memory use.

1.3.1 Purpose

The main purpose of this thesis is to design a framework that enables video streaming to resource-constrained terminals and performs a bit-rate adaptation that can be used to improve the quality of video transmission by tackling short- and long-term bandwidth fluctuations. The framework is required to handle network and terminal resource constraints.

Network issues

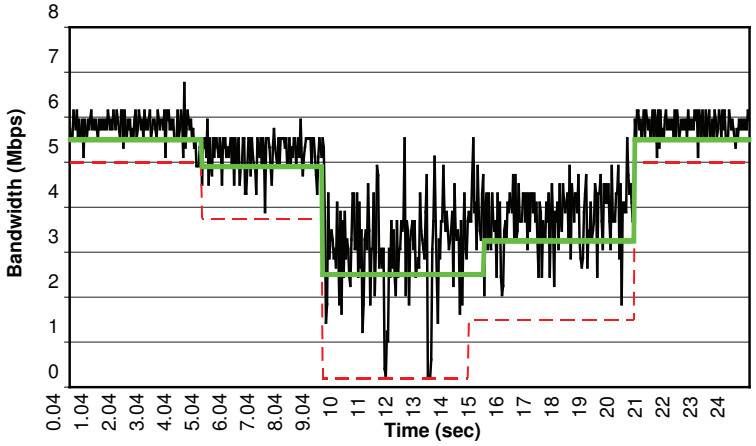

Figure 1.4. Bandwidth fluctuations on a wireless channel.

Figure 1.4 shows an example of the bandwidth fluctuations as perceived by a TCP stream sent at maximum packet rate. For the first 5 seconds the TCP stream is the only user of the link. Then, for the next 5 seconds an additional wireless stream shares the link, for the next 5 seconds a third stream is added, then for the next 5 seconds the second stream stops and, finally, after 20 seconds, the third stops as well. Every 40 ms the amount of bits that have arrived is measured to obtain the effective bit rate for the TCP stream.

The solid line depicts a video quality level close to the measured value [4]. The price paid for this is an occasional loss of data. In general, occasional losses are

[4]For the sake of simplicity, the average bit rate is used in the figure. In reality, the bit rate of individual frames is slightly different from the average, as explained in Appendix A.

acceptable in a streaming application as these losses do not affect the perceived video quality very much.

To exploit the available bandwidth to the full, the video bit-rate curve should retrace the measured curve shown in Figure 1.4. However, this is impossible in practice due to such factors as the variable bit rate of the video, the use of fixed-sized video layers, etc. Even when the video bit rate follows the available bandwidth, the end user is faced with the unpleasant effect of frequent quality changes [37].

We could change the bit rate of the video on the basis of feedback. This would trigger the source to change to another bit-rate value when appropriate. In Figure 1.4, we base this triggering on the changes in the two average bandwidth values denoted by dashed and solid lines. However, knowing when the bit-rate change should occur is not enough, we need to know by how much the bit rate should change. Should we aim for a worst-case (dashed line) or a more 'optimistic' guaranteed level (solid line)?

Using the more pessimistic dashed line, the video gets through with maximum probability. However, the drawback is the low effectiveness of the bandwidth usage. We see that due to the fluctuations in the intervals $[1, 5)$ and $[20, 25)$, the worst-case is 1 Mbit/s below the measured bit rate, while in interval $[9, 14)$ the bandwidth fluctuation becomes so high that the worst-case scenario brings us no video at all.

A video that follows the solid line utilizes the available bandwidth effectively and keeps the amount of lost data within reasonable limits. The challenge set by the network is how to provide a solution that executes bit-rate adaptation in accordance with an 'optimistic' pattern of behavior.

Terminal issues

Besides the network aspects, the capability of the receivers also comes into play. Different CE devices have different processor/memory potential, thus not every receiver may be capable of processing all video data that is streamed by the sender. A receiver should be allowed to receive and process different amounts of data in accordance with its capability.

Figure 1.5 shows an example of the processing resource consumption for decoding a video. For the sake of simplicity we assume that a video can be processed at three different quality levels. The quality level of the processing is determined by the video bit rate; higher bit rates require higher processing levels and operation at lower quality levels forces the decoder to process less video information. The decoding of a video at the first quality level only consumes an average of 23% of the resources available on the terminal, processing at the second level requires 38%, while processing at the third level consumes about 75% of the resources. Table 1.2 shows the relationship between quality level and video bit rate.

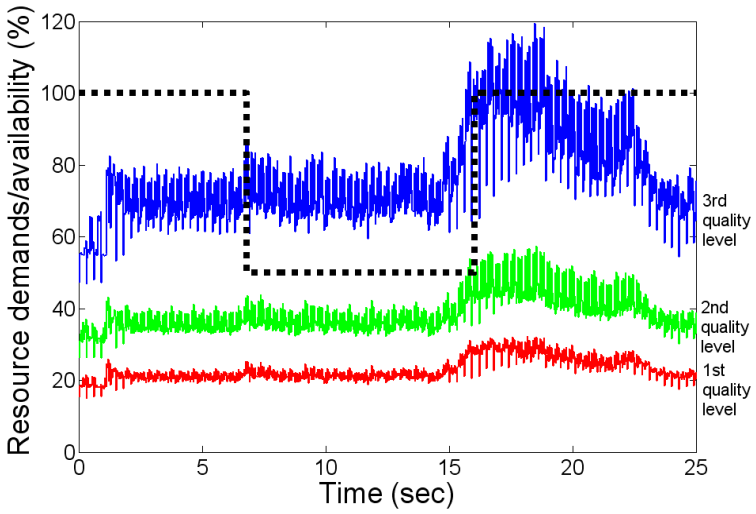

Figure 1.5. Resource availability vs. resource demands for video decoding.

Quality level	Bit-rate
1	2 Mbps
2	3 Mbps
3	6 Mbps

Table 1.2. Relationship between quality level of the decoding and bit rate of the video.

As shown in Figure 1.5, during the first 7 seconds the video-processing application is allowed to use almost 100% of the available resources. From 6 to 16 seconds another application uses the resources, leaving the video processing with only 50% resource availability. In this case, processing at the third level is not possible, so the decoder changes to the second quality level. After 16 seconds, the video-processing application is again allowed to use 100% of the resources, however, high resource demands due to an increase in the complexity of the video in the 16 to 20 seconds time interval force continuous switching between the third and second quality level.

Whenever the resource demands are at the limits of the resource availability, the decoding process is often forced to make changes to the quality level, which leads to frequent quality changes in the output video. As has already been mentioned, the end user does not appreciate these quality changes.

In the example shown in Figure 1.5 it would be best for the decoder to operate at the second quality level during the time period from 16 to about 19 seconds. The

quality provided by the decoder is then average, but stable.

The key challenge set by the terminal is to develop a decoding algorithm that provides a good video quality whilst keeping the number of quality changes to a minimum.

System level issues

The decisions made by the sender with respect to the bit-rate adaptation and the decisions made by the decoding process with respect to the quality level for decoding need to be synchronized when a terminal is connected to a network. Figures 1.4 and 1.5 show that the intended behavior of the sender and terminal do not agree during the time period from 16 to 21 seconds. The terminal tries to process the video at the third quality level, while the bit-rate adaptation shrinks the bit rate to about 3 Mbps, thus leaving the terminal with an insufficient amount of data.

It is essential to create an overall control mechanism that synchronizes the local decisions made by the sender and the terminal if the designed framework is to be deployed successfully.

1.3.2 Assumptions and limitations

Several assumptions have been made with regard to the environment in which the framework should operate.

MPEG-2 is the only standard for the video distribution inside the home network. As a result, all the client devices are capable of processing a MPEG-2 video. The gateway gets the video from the Internet or from a storage device inside the home in non-scalable MPEG-2 (a typical source for video is DVD). The gateway has sufficient resources to transcode the video into a scalable format and to stream the video over the network.

The connection between sender and receiver is either wireless or a combination of wireless and wired links. The wireless connection is formed with an access point. Since our solution focuses on home networks, the usual number of network devices in the network is one (the access point) or, in exceptional cases, two. If traffic shaping is performed on the network, we assume no collaboration between the framework and the traffic shaping mechanism.

The service discovery mechanism is outside the scope of this thesis. We assume that the sender and receivers know each other, i.e. they can connect directly with their IP addresses.

1.3.3 Contributions of this thesis

The main contributions presented in this thesis have been published in a number of research papers [38, 39, 40, 41, 42, 43, 44, 45]. The contributions fall within three areas – scalable video coding, video streaming, and terminal resource management.

These areas are linked by the context of the research, QoS framework for video streaming to resource-constrained terminals.

In this thesis we present a framework for achieving high-quality video transmission over wireless networks based on scalable video coding. An important advantage of the proposed scalable video coding scheme is that it only requires a non-scalable legacy video decoder to process each layer. The approach can therefore be used for video streaming to CE devices.

Slow bandwidth fluctuations are present in all networks when streams share the bandwidth. The proposed framework removes video information to reduce the bit rate of the video in a controlled fashion. At the sender side, a transcoder adapts the bit rate to the slow fluctuations while the fast fluctuations, which are specific for wireless networks, are dealt with by throwing away layers of the scalable video. The absence of dependencies between frames in enhancement layers makes the system resilient to the loss of arbitrary frames from an enhancement layer. The proposed methodology for creating enhancement layers based on frames with independent fail characteristics is not MPEG-2 specific and can be applied to other coding standards.

Important quantities are the number of layers and the size of the layers, which define layer configuration. A change in the network conditions forces a change in the number or size of one or all of the layers. We demonstrate that knowledge of the network conditions helps in choosing an optimized layer configuration for the scalable video coding.

Scalable video is chosen to make trade-offs between user-perceived quality and terminal resources. We show that a controller can be used for a terminal to optimize perceived quality with respect to the available processor power and the amount of input data. The developed controller does not depend on the type of scalability technique. The controller uses the strategy that is created by means of an Markov Decision Process (MDP). Additional research addresses parameter setting optimization.

1.4 Outline of the thesis

Chapter 1 provides an overview and background of the research covering introduction to the research area, formulation of the research questions and overview of the issues that are addressed in this thesis.

The top-down approach is used in the description of the system (see Figure 1.6). The conceptual overview of the system is first formulated in Chapter 2, specifying but not detailing any subsystems/components. Each subsystem is then refined in yet greater detail in Chapters 3 to 5.

Chapter 2 focuses on the description of the framework and its components.

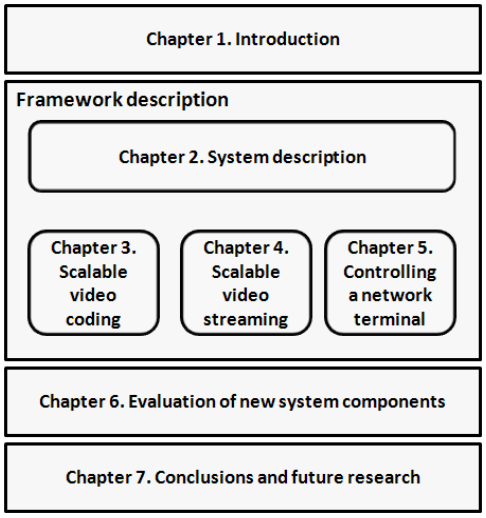

Figure 1.6. Structure of the thesis – a top-down approach.

The chapter describes responsibilities of the components and communication between them. This chapter discusses how video data can be efficiently distributed to a receiving device. The key technology is scalable video coding. The chapter shows how a framework assists in adapting the digital code to the changing transmission conditions to optimize the quality rendered at the receiver. The conceptual description of the framework has been presented in 2005 at the IFIP International Conference on Embedded and Ubiquitous Computing (EUC-05) under the title "A Framework for Video Streaming to Resource-Constrained Terminals" [39].

Chapter 3 provides an overview of the developed scalable video coding scheme. The proposed video coding scheme requires standard non-scalable MPEG-2 video decoders, and is suitable for the needs of the both, video streaming and video processing, parts of the framework. The new scheme produces video without dependencies of the base layer on an enhancement layer and dependencies between frames in an enhancement layer. This allows an easy drop of an arbitrary number of frames from an enhancement layer and decreases the required bandwidth, which helps to accommodate wireless link fluctuations. The paper that describes the proposed scalable video coding and presents results of an evaluation, where the new scalable video scheme is compared to a non-scalable solution has been presented in 2005 at the Ninth IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA 2005) under the title "Wireless Streaming based on a Scalability Scheme using Legacy MPEG-2 Decoders" [44].

Chapter 4 covers the issue of controlling the streaming of scalable video over wireless links and describes how scalable video transmission can adapt to the

widely and frequently fluctuating bandwidth. The streaming uses TCP, which provides the following advantage: all the intelligence of the system is concentrated at the sender side; no network protocol adaptation is needed. A dynamic adaptation of the scalable video to the available network resources that is based on a hierarchical approach to handle at the highest level the slow bandwidth changes, and at a lower level the fast bandwidth changes has been presented in 2007 in Journal of Systems and Software under the title “Hierarchical resource allocation for robust in-home video streaming” [45].

Chapter 5 mainly focuses on the terminal resource management research and shows that scalable video techniques together with a controlling mechanism for a device that receives and decodes video data provide good user experience while adapting to the limitations of the device. The challenges are resource limitations of the device (processor) and network (bandwidth). The developed controller that optimizes user-perceived quality by smoothening the quality fluctuations and avoiding deadline misses has been presented in 2004 at the IEEE International Conference on Multimedia and Expo (ICME 2004) under the title “Predictive Control of Video Quality under Fluctuating Bandwidth Conditions” [43]. A method to decrease an amount of pre-calculated strategies for the controlling mechanism by exploring the dependency between layers configuration and network conditions has been presented in 2005 at the IEEE 14th International Conference on Computer Communications and Networks (ICCCN2005) under the title “Adaptable video streaming over wireless networks” [38].

Chapter 6 covers the evaluation of the proposed solution. The tests include evaluation of the scalable coding technique, streaming techniques, and terminal resource management. This work shows that scalable video techniques together with a controlling mechanism for a device that receives and decodes video data provide a solution to some challenges posed by home networks. The challenges are resource limitations of devices (processor, memory, etc.) and network (bandwidth). Additionally, a wireless network usually has bandwidth fluctuations that consecutively lead to rapid throughput fluctuations. These rapid changes severely reduce the quality of viewing as perceived by an end user. We developed a controller that optimizes user-perceived quality when looking at available input data and available processing power. The quality is optimized by smoothening the quality fluctuations and avoiding deadline misses.

Chapter 7 provides a summary of the various concepts and methods introduced in this research. It also covers applicability, limitation, as well as future recommended expansion on this research.

2

System description

This chapter describes a framework that enables real-time wireless video streaming between various CE devices in a home environment. The chapter defines the structure in which the different elements of the framework are described. As the basis for the description of the framework we use a distributed system that consists of a sender and receiver connected to a network.

The chapter describes the minimum set of framework components required for the video streaming and video processing on a resource-constrained terminal that satisfies the requirements coming from the problem statement section of the previous chapter. A component has a certain input, output, behavior, and may have responsibility to take decisions that influence the configuration of the whole system. The behavior of the component is the most important characteristic; it is described informally or in precise algorithmic notation.

2.1 Adaptation to the resource limitations

The system consists of three basic elements: sender, receiver and network. Two of the elements, receiver and network may and, most of the time, do have resource limitations. Both receiver and network resources, are often used by other applications, which makes control of resource availability impossible. In addition, network bandwidth may fluctuate due to physical properties of the medium, which

makes impossible not only control, but also prediction of changes in the resource availability.

A system that streams video without taking into account availability of resources, experiences data losses during transmission or during decoding of video material every time there is resource shortage. The rendering of video material is very sensitive to data losses because usually the losses create artifacts (mostly, due to time-dependency in video decoding) that are extremely annoying to the user. To make a sender-receiver react to changes in resource availability, a policy is proposed to describe how, when and what parts of the system should adapt to the changing conditions.

The change in network resources (bandwidth) suggests to have the sender perform video adaptations to minimize transmission losses and maximize quality. These adaptations must be highly flexible due to the unpredictability of the transmission losses. The terminal resources such as processor cycles, define the quality of the video processing. Changes of terminal resources are handled locally in the receiver.

Figure 2.1 shows the basic scheme of the policy for adaptation to resource changes in the sender-receiver system. The sender continuously observes the network and the receiver observes its terminal resource constraints. As soon as the network conditions change the sender reconfigures the transmitted video code (i.e. the number and sizes of the layers of scalable video code). Changes in the video code are observed by the receiver, which adapts its processing settings. The time granularity of the actions shown in Figure 2.1 is given in Table 2.1. The sender observes the network every time a part of video data is transmitted. A typical video stream comprises 25 or 30 frames per second, so every 30 – 40 ms a new frame is available for transmission. That gives a time granularity in milliseconds for network observations. Not every change in network conditions should lead to the reconfiguration of the video coding. Depending on the projected sensitivity of the system and the particularities of a network environment, the time granularity of video-coding reconfiguration could be in seconds or, even, minutes. The time granularity of the resource-constraints observations at the receiver is in seconds or minutes, because the changes of terminal resources are in majority the result of starting/finishing of an application. Consequently, the adaptation of processing settings at the receiver has a time granularity in seconds/minutes.

The video configuration information is absolutely essential for the decoding process. Incorrect configuration information leads to an incorrect decoding process and artifacts. The purpose of the whole framework is the suppression of artifacts. To guarantee the successful transmission of the change in video code configuration, this information is incorporated in the video data stream.

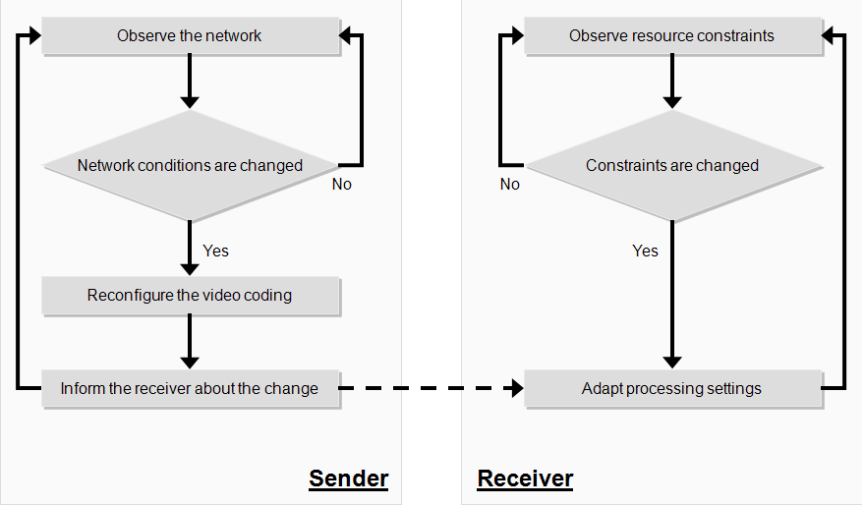

Figure 2.1. The decision-making algorithm for the sender and receiver. Solid lines show the transition from one algorithm block to another as the result of an event. Dashed lines show events that initiate changes on an another device.

Device	Action	Projected time granularity
Sender	Observe the network	milliseconds
Sender	Reconfigure the video coding and inform the terminal about the change	seconds / minutes
Receiver	Observe resource constrains	seconds / minutes
Receiver	Adapt processing settings	seconds / minutes

Table 2.1. Overview of the projected time-granularity for the decision-making algorithm.

2.2 System design

The design of our framework is shown in Figure 2.2 [1]. A receiver, which is connected wirelessly to the sender, processes incoming video data in accordance with local resource limitations. The sender adapts the input video data in accordance with the network conditions.

The video data comes to the sender from a storage inside the house (for ex-

[1]For simplicity, in the following figures, we omit *Lossless network* that provides an input to the transcoder as well as *Display* that always serves as an output for the decoder .

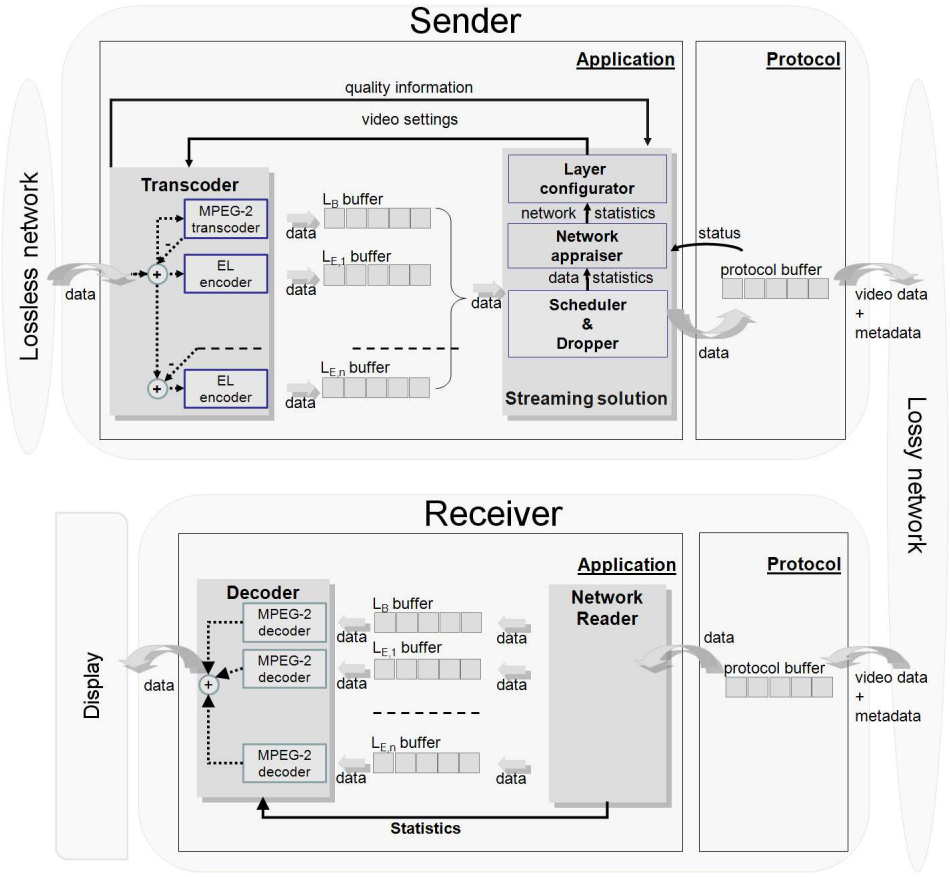

Figure 2.2. Video-distribution system that is based on multi-layered scalable video. The sender application comprises of the scalable video transcoder and the streaming solution. The receiver application consists of scalable-video decoder and the network reader.

ample, DVD player) or from online source outside the house (for example, video-on-demand). The developed system accepts video only in MPEG-2 format. If necessary, the sender transforms the incoming video into MPEG-2 format. This transformation is outside the scope of the thesis.

The sender produces a set of video streams, where every stream carries one layer of a scalable video. Every stream can be decoded by a non-scalable MPEG-2 video decoder. The decoded layers can be merged by a summation module to obtain video of a high quality.

The video streams are sent to the receiver via an unreliable communication channel (e.g. wireless network). The sender takes care that the streams are packetized in accordance with specifications of the communication protocol. The packetization at the sender and de-packetization of data at the receiver are outside the scope of the thesis.

The receiver acquires data streams from the network, decodes every stream, merges the decoded layers and displays the result on the screen.

The video data travels through the sender via the following path:

1. The transcoder reads non-scalable MPEG-2 video stream from the reliable channel, processes it and writes to the output buffers the video streams that contain layers of the scalable video. The transcoder creates one output buffer for every layer.
2. The streaming solution takes data from the transcoder output buffers and fetches it into protocol buffer, according to the layer priorities.
3. The transmission protocol takes responsibility to send data over network to the receiver.

At the receiver side the video data goes through the following path:

1. The incoming data packets are stored in the protocol input buffer.
2. The network reader is responsible for taking the data from the transmission protocol, sorting the data into layers of the scalable video and storing the layers into decoder input buffers. The decoder has one input buffer per layer.
3. The decoding process reads video streams from the input buffers, processes individual frames, merges the frames and puts decoded frames into the decoder output buffer. The details of the output rendering are device specific and they are not addressed in this thesis.

2.2.1 Sender

The sender takes broadcast or stored content and transcodes it into a scalable video. This video is sent over a wireless network to a receiver. The observed network behavior characteristics are used to make a decision about configuration of the scal-

able video, i.e. for choosing the number and bit rates of layers. The components of the sender shown in Figure 2.2 are discussed in details.

Scalable video transcoder

The transcoder converts non-scalable video into multilayered SNR scalable video in accordance with the current layer configuration.

The input video data is supplied to the transcoder via a reliable channel, which means we can assume that there are no losses or delays in the incoming stream. The input video is compliant with the MPEG-2 standard [46]. Along with the stream, the transcoder takes a set of values that defines the internal parameters of the MPEG-2 compression. The parameters are defined at the beginning of the session and do not change until the transcoder is re-initialized. The transcoder also takes the layer configuration expressed as the number and bit rates of layers as an input parameter. The transcoder checks the layer configuration settings after every processed frame, which means that the output layers may change at run time.

The transcoder operates in one of two modes – the *full processing* mode or the *fast processing* mode. The transcoder in the full processing mode decodes the input stream and encodes it into scalable video streams with new encoding settings. The fast processing mode enables the input stream to be only partly decoded and re-encoded with new bit rate.

The transcoder incorporates information about the currently chosen configuration of layers and estimations of the probabilities of the successful transmission of the layers as user data within the base layer stream. User data is defined by the MPEG-2 standard [46] to allow data not related to the decoding process be placed within the stream. The writing of configuration settings into data streams allows natural propagation of changes through the system, as all parties involved can be informed of the complete change history. A base layer stream is the preferred target because its delivery and processing is guaranteed.

The SNR scalable video [47] coding approach has been designed specifically with inter- and intra-layer dependencies in mind (see Figure 2.3). If a frame from a base layer depends on a frame from an enhancement layer, error propagation may take place when an enhancement layer frame is lost. The used technique removes all dependencies of base layer frames on enhancement layer frames. Furthermore, the ability to drop an arbitrary number of frames from an enhancement layer if network conditions are bad or if a receiver does not have enough processing power, requires that frames in an enhancement layer should have no relation to each other.

The output of the transcoder is a set of streams, where each stream contains one layer of the scalable video coding. Moreover, for every frame in the stream the transcoder produces a PSNR value that represents the quality of the picture with respect to the original.

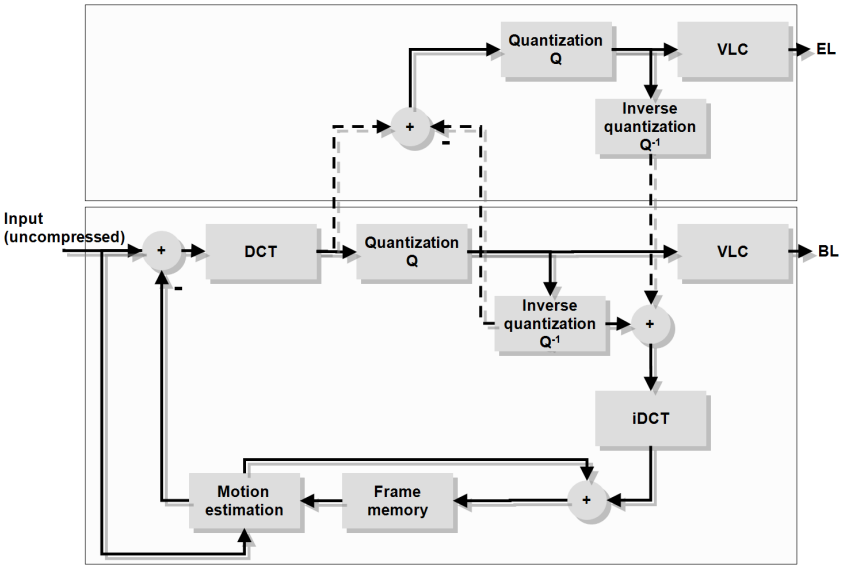

Figure 2.3. Standard MPEG-2 SNR scalable encoder.

Streaming solution

The streaming solution is responsible for:

- the prioritization of the video data and dropping of outdated packets before submitting them to the network protocol,
- performing an estimation of the current bandwidth of the network,
- choosing an optimal layer configuration for the observed network conditions.

The streaming solution takes video streams provided by the transcoder as an input. The solution has *a-priori* knowledge of the layers and can distinguish between the base layer and different enhancement layers. Other input is the limitations of the lifetime of a packet in the buffer (maximum time that a packet of may spend in the transcoder output buffer before being sent).

Since BL information is absolutely necessary to enable scalable video usage, this layer has the highest priority. The priority of EL decreases as the layer number increases. When a frame from enhancement layer number x ($L_{E,x}$) is transmitted and a frame from base layer (L_B) arrives, a sender will send the L_B frame after transmission of the current packet belonging to $L_{E,x}$ (if any). When a frame from $L_{E,x}$ arrives, it preempts a frame from $L_{E,y}$ where $y > x$ (see Figure 2.4).

When a channel degrades, the buffers of the sender become full. This affects above all the low-priority streams and spreads to the higher-priority streams.

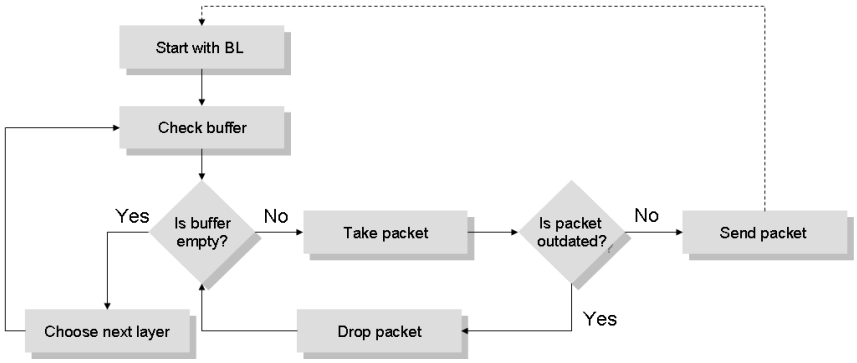

Figure 2.4. Reading video data from transcoder output buffer.

The video streaming has very strict time requirements, so if a frame transmission is significantly delayed, it is not possible for the frame to be used on the decoder side. Consequently, transmission of the outdated frame wastes the bandwidth and should be avoided. The scheduler checks every packet to see if it is outdated and if it finds an outdated packet, it drops the packet before it is submitted to the network protocol.

The scheduler stores the packets into a protocol buffer. The information regarding the fullness of the protocol buffer is communicated to the network appraiser as an indication of the network status.

The streaming solution performs an estimation of the current bandwidth of the network. The estimations are used for choosing an optimal layer configuration.

The solution observes the fullness of the protocol buffer and makes an estimation of the current bandwidth based on the changes in the amount of data in the buffer. The frequency of scheduler observations is an input parameter for the network appraiser. Knowledge of the amount of information that goes through the buffer during a period of time gives a good indication of the network throughput. Moreover, changes in the bandwidth availability are easily detectable, as the decrease in the number of data packets in the buffer means that the current bit rate of the stream(s) is lower than the network throughput, while the increase in the buffer fullness suggests that the bit rate is higher than the available bandwidth.

The dropping of a part of the video data is a mechanism for handling short-term bandwidth fluctuations. If the available bandwidth decreases for a long period of time, adjustments to the data streams at the level of video coding are necessary. The streaming solution chooses the number and bit rates of layers to be produced based on the information acquired about network conditions.

Observations of the available bandwidth are used to estimate the tendency for changes in the network bandwidth, B^R , that is currently available. The value of

B^R together with the history of changes constitutes the observable network conditions.

For the observable network conditions the streaming solution estimates the loss probability per layer for the set of layer configurations and the average quality that can be delivered by the configuration (by looking at probabilities of the successful transmission of the layers and knowing an objective quality of the video from the rate-distortion curve that is based on the quality measurements given by the transcoder). The layer configuration that delivers the highest quality under the given network conditions is then used in the system.

The algorithm for choosing a layer configuration consists of the four steps that are repeated continuously. The general description of the steps is given in Table 2.2. A detailed discussion of the layer configuration algorithm is provided in Chapter 4. The output of the algorithm of layer configuration search is the number of layers and the bit rates of layers that should be used by the transcoder for the next frame.

Step	**Description**	**Calculated parameter(s)**
1	Updating the rate-distortion dependency for the transcoder. The real-time recalculation of the dependency is critical to enable the component to estimate the average picture quality that is delivered by video stream(s) of the given bit rate.	Dependency between *frame size* and *frame quality (PSNR)*
2	Evaluating the network status by observing the changes in the bandwidth. The changes are monitored by observing variations of buffer fullness of the transmission protocol	*Available bandwidth* (momentary, history of changes)
3	Defining the set of possible layer configurations from which the best suitable candidate will be selected. Estimating probabilities of the successful transmission of the examined layer configurations under the network conditions and predicting the average video quality that can be delivered by the layer configurations. quality.	Paired set of *layers configuration – average quality*
4	Choosing the best layer configuration and informing the transcoder about the new layer configuration.	*Number of layers* and *bit rates of layers*

Table 2.2. Four steps for choosing a layer configuration.

2.2.2 Receiver

The receiver is responsible for obtaining the required number of layers from the ones available, for decoding and for displaying video data. The components of the receiver shown in Figure 2.2 are discussed in detail.

Network reader

The main task of the network reader is to receive the video data from the network, sort it into different layers and feed the layers to the decoder. The data streams from the sender are the only input of the network reader. In the case of scalable video coding, transmission delays may lead to a loss of synchronization between the layers, forcing a frame from one layer to arrive significantly later than a corresponding frame from another layer. Frames from base and enhancement layers should be merged during the decoding process. If a frame layer arrives too late to be merged in time with the frames from other layers, this frame is discarded before it is offered to the decoder.

The component supplies the decoder with synchronized data and informs the decoder about the number of layers that are present for every frame.

Decoder

The decoder reads the video streams and processes them so that at the output there is an uncompressed video frame, which can be shown on the device's display. The decoding process is implemented as a scalable video algorithm, where the number of processed layers is a parameter that changes the output quality as well as the resource consumption of the component.

The input of the decoder is the number of layers to be decoded for a particular frame. If, for any reason, the number of layers, asked to be decoded, is higher than the number of layers available, the decoder processes all available layers. Moreover, if there is no input relating to the number of layers to be decoded, the decoder processes as many layers as are available. These situations should be avoided as much as possible, because they may result in the loss of synchronization between layers and, consequently, alterations to the picture shown if some frames are missing during the transmission. To illustrate this, imagine that at a certain period of time there are frames 1 and 2 in BL buffer and only frame 2 in EL buffer. If the decoder is asked to process two layers for the frame 1 , it processes frame 1 of BL and frame 2 of EL.

The decoder needs a given amount of resources on the terminals. These resources are not always available, either because the resources are insufficient from the start, or because other applications use part of the available resources. The amount of available resources changes with time. The decoder, therefore, should be able to change it's resource consumption. The latter is possible if the decoder

is implemented as a scalable video algorithm. A decoder implemented as an SVA needs a controlling mechanism to ensure that resource consumption of the decoder is scaled to the limitations of the device. The controlling mechanism calculates the trade-off between the resource consumption of the decoder and the quality of the resulting video. The controlling mechanism is discussed in detail in Chapter 5.

The decision of the controlling mechanism depends on the amount of video data available for the next frame, the amount of terminal resources that are available, the amount of resources that are used by the decoder and an estimate of how much data will be available for the frame that will come after the next one. Since the amount of available video data is dependent on the network, a controller strategy is linked directly to the network conditions.

It is important to mention that it is only possible to use the controller in a system where a budget scheduler schedules the resource usage (such as a processor) of different processes. A process (or task) should not only be scheduled, i.e. told when to run, but should also be forced to stay within allocated amount of resources. The latter enables resource-availability guarantees to be made for all of the tasks in the system.

It is also assumed that there is a resource manager that decides on the distribution of the system resources over the applications that are running. The resource management is not required if the video-processing application is the only application on the device. In this case, the budget available to the application is around 100%, the number of layers and overall bit rate that can be handled by the device do not change over time and are defined by the device processing capabilities (hardware/software configuration of the device). As soon as there is another application that requires a substantial amount of resources, the distribution of the resources may change, lowering the capabilities of the video processing, so the overall bit rate that can be handled and the number of layers that can be processed change as well. The mechanism by which the resource manager reacts to the changing conditions is beyond the scope of this thesis.

2.3 Outline of the system description

The following three chapters address in turn the domains of the research area, i.e. scalable video coding, video streaming and terminal resource management.

Scalable video coding research presents an improved SNR scalable video coding scheme for a wireless video streaming system. The proposed video coding scheme requires standard non-scalable MPEG-2 video decoders. The new scheme produces video in which the base layer is not dependent on an enhancement layer and there are no dependencies between frames in an enhancement layer.

Video streaming research addresses the issue of controlling the streaming of

scalable video over wireless links and describes how scalable video transmission can adapt to the widely and frequently fluctuating bandwidth while maintaining a smooth video display. Video streaming based on scalable video coding makes it easy to drop an arbitrary number of frames from an enhancement layer and decreases the required bandwidth, which helps to accommodate wireless link fluctuations.

Terminal resource management research shows that scalable video techniques together with a controlling mechanism for a device that receives and decodes video data provide a solution to the challenges posed by home networks. The challenges include the resource limitations of devices (processor, memory, etc.,) and network (bandwidth). The controller optimizes user-perceived quality when looking at the available input data and available processing power. The quality is optimized by smoothing out the fluctuations in quality and avoiding deadline misses. The strategy used by the controller is created off line by means of a Markov Decision Process.

3

Scalable video coding

In this chapter we describe a scalable video coding technique that transforms non-scalable MPEG-2 video coming to the sender into a set of streams containing layers of scalable video as described in Chapter 2. This chapter portrays the research that preceded the development of the coders, including analysis of various scalability coding alternatives, their performance and general suitability for the developed framework. The chapter also provides an overview of the transcoding techniques that are used.

3.1 Requirements to the developed coding

Using as a basis the general definition of the requirements for the video streaming system that are mentioned in Section 1.1.2, we define the following requirements for the scalability technique to be developed:

- There should be no coding dependency of base layer on a enhancement layer or of $L_{E,x}$ on $L_{E,y}$ where $x < y$. This is necessary in order to ensure that the absence of a part of or the whole of EL does not create any disturbance in BL or lower EL. It is also necessary for correct functioning of the sender-side prioritization of the video layers and terminal resource management. The prioritization scheme that satisfies the requirements has the following structure ($\Phi(f)$ is priority of frame f): $\Phi(FB^n) > \Phi(FE_i^j)$ and $\Phi(FE_i^j) >$

$\Phi(FE^k_{i+1})$ for any natural $n, j, k < n_F$, where FB^n is frame number n in L_B , FE^j_i is frame number j in $L_{E,i}$, FE^k_{i+1} is frame number k in $L_{E,i+1}$ and n_F is the number of frames in the video sequence. Based on this rule, only the layer number, and not the frame number, plays a role in the priority assignment. In this case, the inter-layer frame dependencies have no influence – FB^n may depend on FB^m and FE^j_i may depend on FE^k_i (for any natural $n, m, j, k < n_F$).

Having a coding dependency of lower layers on the higher layers significantly complicates the priority mechanism. Let us consider the example of MPEG-2 SNR scalable video coding, where some frames of $L_{E,1}$ are used in the motion compensation of L_B. A frame in $L_{E,1}$ should be merged to the L_B frame with the same frame number. Figure 3.1 shows dependencies between frames in the L_B and $L_{E,1}$. For the sake of simplicity, the dependency of FE^x_1 on FB^x is not shown.

The figure demonstrates an example where frame FE^1_1 from the enhancement layer is needed for the correct decoding of $FB^{2..9}$ frames from the base layer. The loss of frame FE^1_1 leads to errors in the frames of the L_B. Moreover, because frames $FB^{2..9}$ are not as they were expected to be, the enhancement could not be applied to these frames. That renders frames $FE^{2..9}_1$ useless. As a result, the transmission and processing of frame FE^1_1 is more important than the processing of some frames from the base layer. In general, it is safe to say that $\Phi(FE^1_1) > \Phi(FB^{2,3,5,6,8,9})$. The differentiation in priorities of FE^1_1 and $FB^{4,7}$ is, however, not that clear. The loss of either FB^4 or FB^7 leads to errors in frames $FB^{2,3,5..9}$ and $FB^{5,6,8,9}$ respectively. With no much difference in the amount of the frames affected, it is hard to say what loss, FE^1_1 or FB^4 , has bigger effect on the quality of the decoded video. The answer depends on the bit rates of the layers, amount of motion in the video material, etc. Thus, the inter-layer dependencies complicate algorithms that rely on clear definition of the frame priorities, making the sender-side prioritization and terminal resource management virtually impossible.

- There should be minimum dependency between frames in EL. The ideal target is for EL frames to be independent of each other to make it easy for an arbitrary number of frames to be dropped from an enhancement layer, which helps to accommodate wireless link bandwidth fluctuations or changes in resource availability on a terminal.
- The developed scalable coding scheme should produce streams that can be decoded by standard non-scalable decoders. If this is the case, a minimum of modifications will be required at the receiver side promoting the reuse of

already existing software/hardware decoders.

- The scalable video coding technique should provide the required flexibility in the choice of the number of layers. The ideal target is an arbitrary number of layers. This is necessary to satisfy the broad spectrum of the receiving devices.

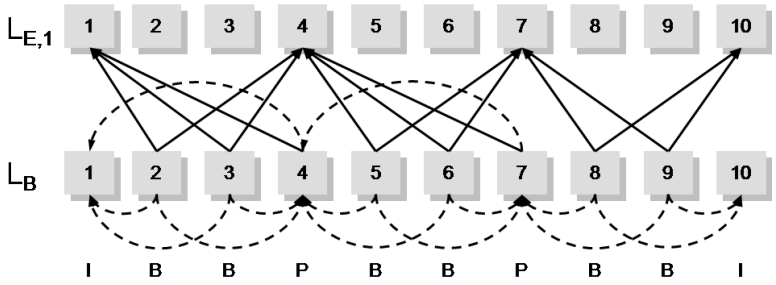

Figure 3.1. Dependency of L_B frames on $L_{E,1}$ frames in MPEG-2 SNR (an arrow from A to B means A depends on B). The dependencies of $L_{E,1}$ on L_B (every FE_1^i depends on FB^i) are not shown for simplicity.

In addition to the functional requirements, the scalable video coding technique developed must also achieve good results in terms of user-perceived quality.

3.2 Preliminary research

Using various approaches based on spatial, temporal and SNR scalability methods, the choice of the basic scalability technique to be used in the system is motivated by investigating which technique can satisfy the aforementioned requirements. The basic description of scalability techniques is provided in Appendix A.3.

The choice and further development of a scalability technique depends on the way in which the decoding process is organized. Possible approaches to the organization of the decoding process for scalable video are depicted in Figure 3.2. Type-I assumes a video where BL and EL are encoded as standard non-scalable video streams. The streams are decoded by different decoders and then merged together. The merging operation may, if necessary, include scaling operation. Type-II takes scalable video that complies with a standard scalable technique (such as MPEG-2 SNR). The output of the decoder is a joint decoded video. Type-III operates with video streams that can be merged together before actually being decoded.

Type-I decoding requires that a device is capable of processing multiple layers at the same time. This can be implemented either via parallel processing or by exploiting a time-division technique. In both cases, a terminal should have significantly more computational resources than other types of scalable video decoding.

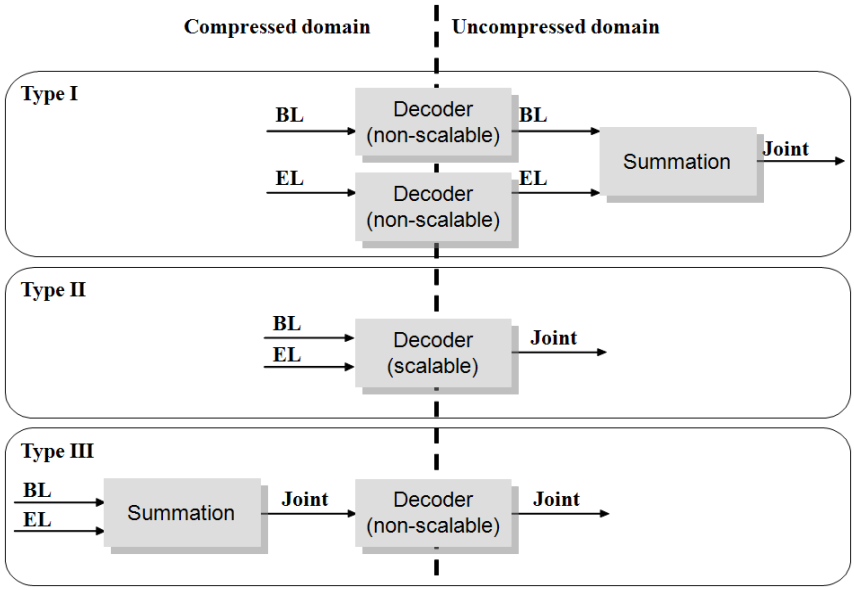

Figure 3.2. Three types of organizing decoding process for scalable video coding.

Type-II, which corresponds to the classical ideas of scalable video coding [46], usually requires the least amount of resources. The disadvantage of the approach is the standardization, and in order to be deployed successfully the decoding should be compliant with a standard. The present state of the market is such that there are no off-the-shelf systems equipped with a scalable decoder and, moreover, there is no agreement within the research community about the particular scalability standard that should be used. Furthermore, scalable decoders are more algorithmically complex.

Conceptually speaking, a Type-III decoding process has the greatest chance of being implemented in CE devices [1]. The advantages of this type include: reuse of a standard non-scalable decoder, low resource demands and easy modifications on the decoding side. The encoding techniques based on temporal scalability and/or non-standard data partitioning are the primary candidates for the Type-III. These techniques, however, encounter significant problems with regard to the graceful quality degradation if a part of the enhancements or – even worse – all of the enhancement layer(s) is not transmitted.

[1] *De-facto* market situation is very different. Various implementations of Type-I decoding process in combination with transcoding are present.

3.2.1 Temporal scalability

The temporal scalability method produces video where EL has no influence on BL, because it is composed of B frames that have no influence on I and P frames that are stored in BL. For the same reason, frames in the EL have no dependencies on each other. Moreover, the layers can be merged in the obvious way into a single stream, by putting B frames back between I and P frames. The result is a usual non-scalable video stream that can be decoded by a non-scalable decoder (Type-III, in Figure 3.2).

The main drawback of this solution is subjective quality degradation. The quality degradation is the direct result of the decrease in the frame rate of the video if EL layer is not available, hence the frame rate is one of the most important characteristics that affect user perception [37].

Another problem associated with the temporal scalability method is the low flexibility in the choice of the number of layers and their bit rates. The number of B frames in a stream determines the number and average bit rate of the enhancement layers. In the general case [2], the number of EL that is possible under temporal scalability equals the number of B frames between two successive I or P frames. For example, a typical MPEG-2 stream has two B frames between two consecutive I or P frames. As a result the number of ELs, n_E , can be one (all B frames go to the single EL) or two (every second B frame goes to the second EL).

Furthermore, the total bit rate of the ELs is also defined by the number of B frames in the stream – the more B frames are present in a stream, the bigger fraction of the overall bit-rate may go to ELs. For a typical MPEG-2 stream the maximal bit rate of EL equals $\frac{1}{2}$ of the bit rate of the BL in the case of having two B frames [3].

Lack of flexibility in choosing number and bit rate of layers combined with huge quality degradation rules temporal scalability out.

3.2.2 Spatial scalability

In the spatial scalability approach there is also no dependency of BL on EL or between ELs. For most of the implementations that are available nowadays, both BL and EL of the spatial scalable video coding can be processed by a non-scalable decoder (Type-I in Figure 3.2, where the summation/merge module also provides scaling functionality).

The drawback of this approach is the interframe dependencies in EL. Although it is possible to create an EL that contains mutually independent frames, such an

[2] Meaning that the B frames stored in EL are picked with a certain regularity, such as every second or every third. Configurations with aperiodic picking of B frames are technically possible but not used in practice.

[3] This is based on the observation that B frames produce $\frac{1}{3}$ of the size of the overall stream.

EL would be of a considerably lower quality [48]. In addition, spatial scalability requires substantial resources for scaling operations during the decoding. These facts make the use of spatial scalability unattractive.

3.2.3 SNR scalability

An SNR scalability is considered to be the best choice for the system developed. SNR scalability is introduced in the MPEG-2 standard [46]. In the approach defined by the standard, motion compensation for a frame in the base layer requires a reference frame created by BL and EL together. Error propagation may therefore take place when an enhancement layer frame is lost.

Since the introduction of the MPEG-2 standard, many improved SNR scalable video coding schemes have been proposed (for example, [49, 50, 51]). The main point requiring improvement in the standardized technique is the dependency of BL frames on EL frames. This weakness is overcome by a modified MPEG-2 SNR video encoder described in [51]. The scheme of the encoder is shown in Figure 3.3. Two rectangles in the figure separate functional modules that are required for encoding base layer frames (bottom square) from modules needed for enhancement layer frames (top square). As shown in Figure 3.3, no enhancement layer data participates in the encoding of the base layer. Moreover, there is no back-loop of any kind in the section that is responsible for EL encoding, so the proposed scheme produces no interframe dependencies in EL.

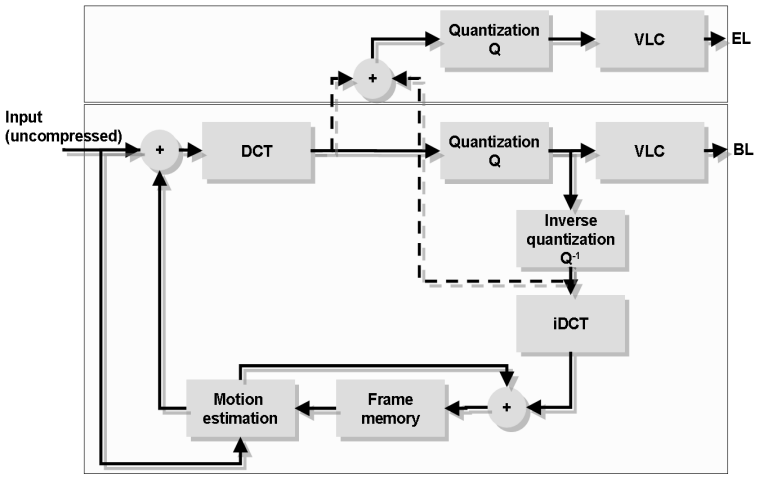

Figure 3.3. Non-compliant MPEG-2 encoder based on SNR scalability.

The drawback of the solution based on SNR scalability is that it requires a special scalable video decoder to handle the processing of the layers. Below we propose a new encoding technique that is derived from the encoder presented in

Figure 3.3. The solution creates layers that are fully compliant with non-scalable syntax.

3.3 MPEG-2 SNR scalable coding

The scalable video coding technique developed in this thesis makes the enhancement layers compliant with non-scalable MPEG-2. To create these layers we wrap the enhancement layer data in the syntax of a non-scalable MPEG-2 stream. A video data stream encoded in accordance with the proposed method can be decoded on a terminal equipped with a scalable decoder that corresponds to Type-I in Figure 3.2.

3.3.1 Enhancement-layer frame types

As mentioned above, the EL syntax should comply with MPEG-2 non-scalable coding. According to the MPEG-2 standard, a stream may contain frames of different types: I , P and B frames. Dropping arbitrary frames from EL requires that frames in EL should have little or no dependency on each other.

The *default* option for creating an EL without having interdependent frames is to create an enhancement layer that consists of I frames. Since I frames have no dependencies on each other, the loss of a frame does not affect subsequent frames.

Options other than the aforementioned one have dependencies inside the group-of-pictures (GOP) structure and, therefore, allow error propagation. We consider three different types of GOPs:

- IP_n , where an I frame is followed by n P frames. For example, with $n = 4$ the frame sequence is *IPPPPIPPPPI*.... When $n = 0$ the stream contains only I frames. GOP size for this approach is calculated as $S_{GOP} = n + 1$.
- IB_n , where I frames are followed by n B frames. For example, with $n = 4$ the frame sequence is *IBBBBIBBBBI*... [4]. When $n = 0$ the stream contains only I frames. GOP size for this approach is calculated as $S_{GOP} = n + 1$. This GOP type, in fact, combines two subtypes – GOP with unidirectional prediction and with bidirectional prediction. Unidirectional prediction covers the case when all B frames are predicted from a single preceding or succeeding I frame. In bidirectional prediction every B frame is predicted from two, one preceding and one succeeding, I frames.
- and $I(B_xP)_yB_x$, which is the standard structure for an MPEG-2 GOP. For example, with $x = 2$ and $y = 3$ a GOP looks like *IBBPBBPBBPBB*. Appar-

[4] Although the given GOP structure does not violate MPEG-2 standard, some practical difficulties exist. Most off-the-shelf MPEG-2 decoders fail memory initialization when encounter a video sequence with GOPs that have no P frames. This is solved by adding an empty P frame to the end of the GOP.

ently, the first two types are the particular mutations of the third type. When $x = 0$ the stream has format IP_y , with $y = 0$ the stream format is IB_x. When $x = 0$ and $y = 0$ the stream contains only I frames. GOP size for this approach is calculated as $S_{GOP} = (y + 1) \cdot (x + 1)$.

To decide between these approaches we investigate the consequences of a frame being lost on the rest of the stream. Whenever a frame is lost during transmission it cannot be used by the decoder. This creates a chance that some of the subsequent frames that rely on the lost frame could not be processed by the decoder. For example, losing an I frame causes all the frames in the GOP to be useless due to the fact that all dependencies in a GOP originate from the I frame.

Figure 3.4 shows that probability that a frame cannot be used by a decoder if 1 out of 10, 1 out of 100 or 1 out of 1000 frames is lost during transmission. Parameters of the corresponding GOP structure are shown on the horizontal scale. When $n = 0$, the GOP consists of only I frames ($x = 0, y = 0$ for $I(B_xP)_yB_x$ GOP). The vertical scale shows frame loss probability at a decoder, i.e. the probability that a frame is unusable for the decoder. If a frame is lost during transmission, it is considered to be lost for a decoder as well as any other frame that uses the lost frame as a reference. Thus, with loss probability during transmission equal to $\frac{1}{1000}$, the real loss at a decoder is much higher.

The probability that a frame cannot be used by a decoder if some frames are lost during transmission can evaluated by two methods – theoretical calculation based on formulas and empirical observations based on simulations. The theoretical approach is used when the precise knowledge of the probability values are needed, whereas the empirical approach gives a rough idea about the influence of the GOP parameters on the probability values.

The probability, P_D that a frame cannot be used by a decoder if some frames are lost during transmission can calculated as follows:

- IP_n: the probability that only one frame from a GOP cannot be used by a decoder is

$$P_D = \frac{1}{n+1} \cdot (P_d(I) + \sum_{i=0}^{n} P_d(P_i)), \tag{3.1}$$

  where $P_d(I)$ is the probability that I frame cannot be used by the decoder, $P_d(P_i)$ denotes the probability that i^{th} P frame cannot be used by the decoder, P_N is the probability to loose a frame during the transmission. I frame does not depend on any other frame in the GOP, so $P_d(I) = P_N$. The first P frame in the GOP, P_1, cannot be decoded in the following cases:

  1. The frame is lost during transmission (the probability of this to happen is P_N),

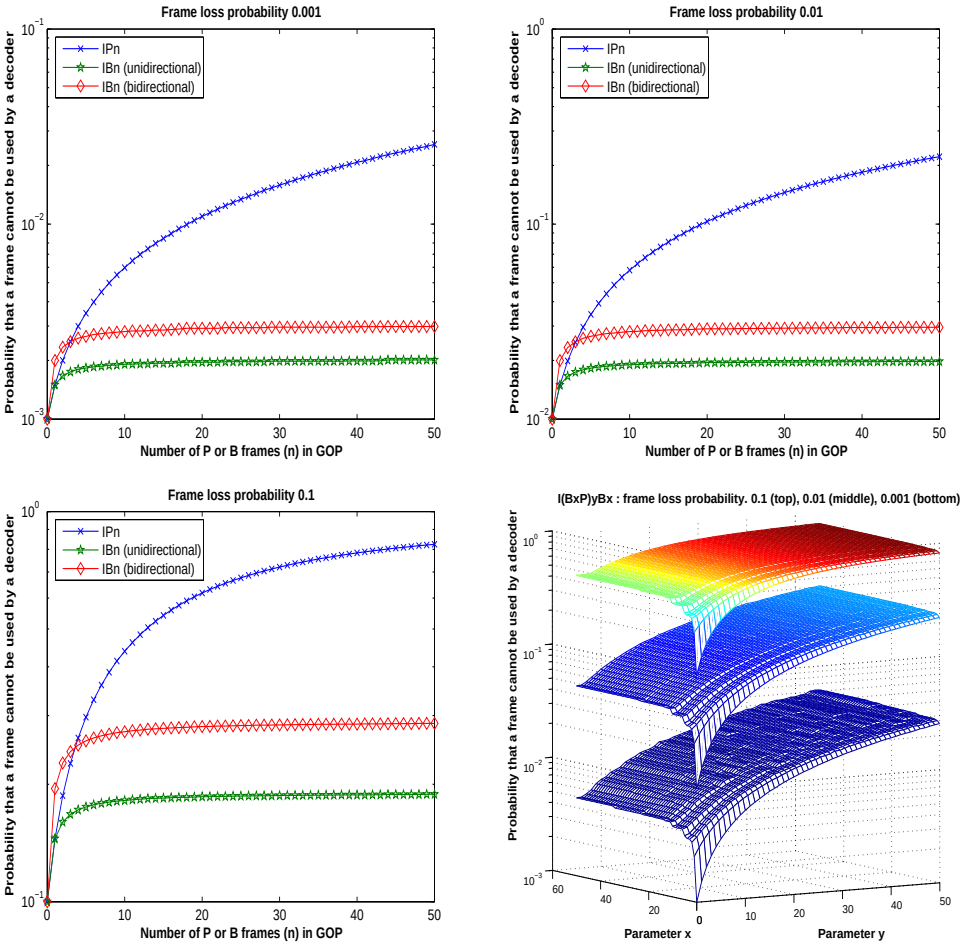

Figure 3.4. Probability that a frame cannot be used by the decoder as a function of a transmission frame loss probability and configuration of the GOP.

2. I frame, on which P_1 depends, cannot be decoded.

Thus,

$$\begin{aligned} P_d(P_1) &= P_N \cdot (1 - P_d(I)) + P_N \cdot P_d(I) + (1 - P_N) \cdot P_d(I) \\ &= P_N + P_N \cdot (1 - P_N). \end{aligned} \tag{3.2}$$

Consequently, for frame P_2, which depends on P_1, we have

$$\begin{aligned} P_d(P_2) &= P_N \cdot (1 - P_d(P_1)) + P_N \cdot P_d(P_1) + (1 - P_N) \cdot P_d(P_1) \\ &= P_N + P_N \cdot (1 - P_N) + P_N \cdot (1 - P_N)^2. \end{aligned} \tag{3.3}$$

Using Equations (3.1),(3.2),(3.3), the resulting formulae to calculate the probability that a frame cannot be used by a decoder

$$\begin{aligned} P_D &= \frac{1}{n+1} \sum_{i=0}^{n} \sum_{j=0}^{i} P_N \cdot (1 - P_N)^j \\ &= \frac{P_N}{n+1} \sum_{i=0}^{n} (n - i + 1) \cdot (1 - P_N)^i \ , \end{aligned} \tag{3.4}$$

where n is the number of P frames in the GOP.

- IB_n: frames inside a GOP are independent on each other with the exception that all B frames are dependant on one or two I frames. The probability that a frame cannot be used by a decoder, if all B frames in a GOP are predicted from a single I frame (backward or forward uni-directional predictions), is calculated follows. The probability that an I frame cannot be used by the decoder is, again, $P_d(I) = P_N$. The probability that a B frame cannot be used by the decoder is

$$\begin{aligned} P_d(B_x) &= P_N \cdot (1 - P_d(I)) + P_N \cdot P_d(I) + (1 - P_N) \cdot P_d(I) \\ &= P_N + P_N \cdot (1 - P_N). \end{aligned} \tag{3.5}$$

Equation (3.5) is valid for all $x \leq n$. Thus, for a video stream with IB_n GOP where B frame are only backward or only forward predicted

$$P_D = \frac{P_N}{n+1}(1 + 2 \cdot n - n \cdot P_N), \tag{3.6}$$

where n is the number of B frames in the GOP.
The bi-directional prediction introduces more dependencies between frames in the GOP. A simulation was used to calculate probabilities for the GOP where B frame are predicted from two I frames (bi-directional prediction). The results are shown in Figure 3.4.

- $I(B_xP)_yB_x$: A simulation was used to calculate probabilities for that GOP structure. The results are shown in Figure 3.4.

Unidirectional IB_n GOP shows lower losses than bidirectional IB_N GOP. This is a result of less interdependencies with unidirectional prediction, where the whole GOP may be become useless for a decoder only if it's first frame (I frame) is lost. With bidirectional prediction, loss of either an I frame from the current GOP of from the succeeding GOP discredits the whole GOP. The results for mixed IB_n GOPs, where different B frames are formed based on uni- or bi- directional prediction, should lie between the two curves.

IP_n , bidirectional IB_n and $I(B_xP)_yB_x$ are much more vulnerable to frame losses than the unidirectional IB_n GOP. The latter performs better even for small sizes of GOP, whilst for larger GOPs the difference increases dramatically. Thus, we consider the IB_n structure with unidirectional prediction as a potential candidate for enhancement layers.

3.3.2 Encoding

The previous section shows two possible stream structures for EL – having only I frames or having all frames of type B bounded by single I frame and single P frame. Performing an encoding that is based on SNR principles as shown in Figure 3.3 and that conforms with the decoding scheme of Type-I (Figure 3.2) could be done in two ways: by cascade of non-scalable encoders or by a specific SNR encoder.

Cascade of non-scalable encoders

Figure 3.5 shows a scheme of the cascade encoding process that corresponds to the encoding principles shown on the scheme in Figure 3.3 (the details are revealed in Appendix C). The original signal is encoded by an MPEG-2-compliant non-scalable encoder into a BL stream (L_B). The resulting stream is decoded and a subtraction module calculates the pixel value difference between the original video and the video in L_B. The difference is encoded in $L_{E,1}$ by the same (or another) encoder.

The major advantage of this method is that there is no modification to legacy encoders. The only extra functionality is the subtraction that can be implemented as a separate software module. This method also enables the creation of ELs with a variety of GOP structures.

The major disadvantage of this method is its computational complexity. Encoding of a scalable video that consists of BL and n ELs requires $n + 1$ instances of encoders and n instances of decoders. Another disadvantage of this approach is the additional inaccuracy during encoding introduced by extra forward and inverse discrete cosine transforms (DCT) that are performed on data.

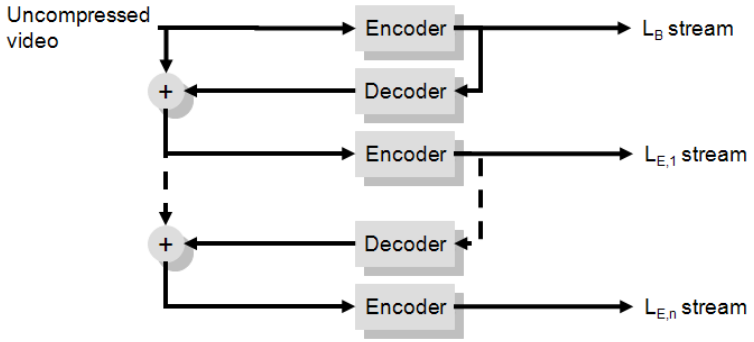

Figure 3.5. Principal scheme of SNR-enabled cascade encoding.

Specific SNR encoder

SNR encoder created in this thesis work has two operational modes regarding the structure of EL: I-frame mode and B-frame mode.

In the first mode, frames of an enhancement layer are encoded as I frames. The contents of a frame is composed of residuals from the encoding of the corresponding BL or previous EL frame. The residuals are quantized and stored in the EL stream in accordance with the scheme in Figure 3.3. As shown in Figure 3.6, value f_x of the original frame is represented in the BL frame by value b_x. If the number of bits available for the BL frame is not enough for lossless encoding of the original frame, a non-zero difference represented by r_x should be encoded in the EL.

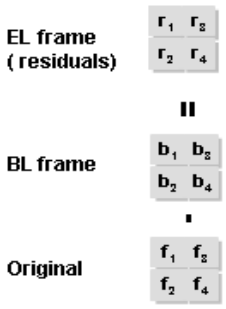

Figure 3.6. Calculation of EL frame content for I-frame mode.

In the second mode, B frames are used to create EL. Theoretically, the encoder in the second mode should use fewer bits to encode the same enhancements [5]. The

[5]Chapter 6 presents comparisons of two modes, where the hypothesis is verified.

main reason for this is that a B-type macroblock with only zero values can be skipped. This is impossible with I frames, where all empty macroblocks are still encoded in the stream.

To benefit fully from B frames, the information contained in this frame needs to be encoded as if it is predicted. If a macroblock of a B frame is not predicted, it is encoded as an I-type macroblock. Thus, B frames will be encoded in the same way as I frames, which means there is no difference with the first mode. We looked at how to make the information that we store in a B frame into predicted information.

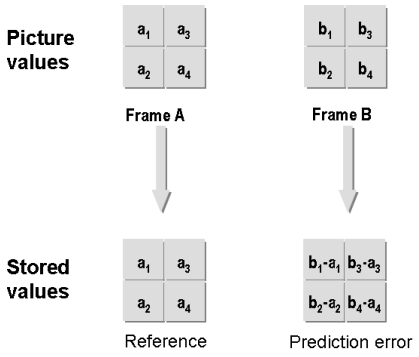

Figure 3.7. Prediction of frame B from frame A.

One result of motion prediction on a frame could be seen as splitting the frame into two parts – one part has predicted values, while the other part contains prediction errors. Figure 3.7 shows an example of predicting a block of values for frame B from the previous frame A. Values a_x of frame A are used as predicted values for frame B. The difference between the original values b_x of frame B and the predicted values are called prediction errors. The prediction error ($b_x - a_x$) is stored in the stream together with motion vectors. The motion vectors tell a decoder what part of a reference frame should be used to restore predicted values (i.e. where to find a_x). A decoder adds predicted values to prediction errors to restore the full frame.

Motion estimation and compensation are performed in the spatial domain (i.e. values involved in prediction represent the real pixels of a picture). However, the values we are working on are in the *DCT domain*, and this means that predicted values as well as prediction errors can be distributed over different macroblocks and, moreover, contribute to an arbitrary number of DCT coefficients.

An obvious way to simplify this is to assume that every macroblock is fully predicted from a macroblock that has the same spatial position in the reference frame (the motion vectors are zero). The summation of values from a reference

frame macroblock and a macroblock that contains predictions errors can be done either in the spatial or in the DCT domain.

We organize the EL frames such that the values in the EL frame are prediction errors of the motion compensation from a reference frame. As shown in Figure 3.8, value f_x of the original frame is represented in the BL frame by value b_x. If the number of bits available for the BL frame is not enough for lossless encoding of the original frame, a non-zero difference represented by r_x should be encoded in the EL. We put r_x into the EL, pretending this is a prediction error.

During decoding a decoder adds r_x to the predicted value p_x , thus giving e_x. To obtain f_x , e_x should be added to b_x. Thus, e_x must be equal to r_x , which implies that p_x is zero. As a result, a reference frame must be empty (zero DCT coefficients), as it is necessary to preserve the values stored in a B frame.

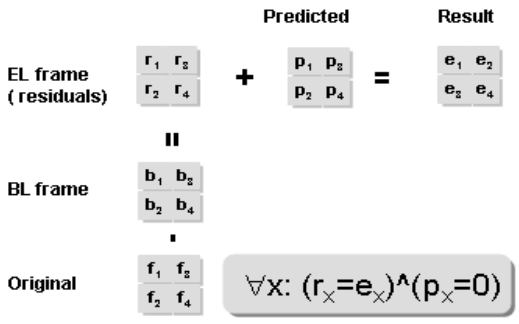

Figure 3.8. Calculation of EL frame content for B-frame mode.

We make I (the first frame of a GOP) empty and encode every macroblock of a B frame as a forward predicted macroblock with zero vectors. A decoder adds zero values from the reference frame to our 'prediction error' values, which results in exactly the same values. The content of a macroblock thus stays the same as in the previous mode, but the macroblock type is now B. This allows more efficient encoding because empty macroblocks can be skipped.

3.4 Transcoding

The previous section discussed SNR scalable encoders. An encoder requires an uncompressed video as an input. In Chapter 1, however, a non-scalable MPEG-2 video is listed as the primary input format. Transforming a non-scalable video into a scalable format is a job of a transcoder.

The most naive implementation of transcoding is re-encoding. The re-encoding means that the input video is decoded by a standard MPEG-2 decoder and the

uncompressed video data is fed into a scalable encoder. A standard decoder can be coupled with either cascade of non-scalable encoders or specific SNR encoder. The re-encoding combines an easy implementation with a high quality of the produced video. The only disadvantage of the approach is high resource consumption.

A more advanced technique is a SNR scalable transcoder that extends the specific SNR encoder with transcoding capabilities. Creating an SNR scalable video from a non-scalable input consists of two steps: lowering bit-rate of the input video to create a BL and storing the difference between the original video and BL into one or more EL. The first step is the primary responsibility of a typical single-layer transcoder, whereas the second step describes an enhancement that can be built on top of the transcoder to enable EL encoding capabilities. In this thesis, two algorithms for making a single-layer transcoder are considered: open-loop transcoding and closed-loop transcoding.

3.4.1 SNR open-loop transcoder

An open-loop transcoder for the BL has a three-step architecture: 1) parsing of the incoming bitstream, 2) performance of the transcoding operation, and 3) compilation of the outgoing bitstream. The first and the last step in the transcoding chain correspond to entropy decoding and encoding. The operation in between performs the actual bit-rate reduction on the video sequence. This type of transcoder does not involve motion estimation, motion compensation or DCT. In the open-loop transcoder the core of the process, the transcoding operation, is re-quantization, which is an inverse quantization followed by a forward quantization with a coarser quantization parameter.

The difference between the original and the re-quantized values is stored in the enhancement layer. This transcoder architecture is depicted in Figure 3.9.

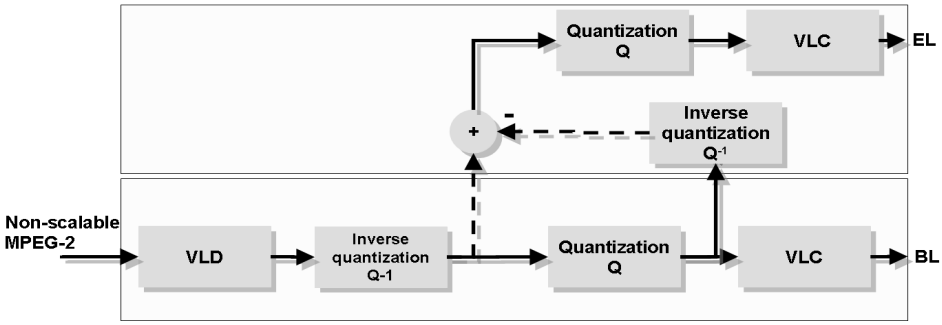

Figure 3.9. Open loop transcoder that produces SNR scalable video.

A major drawback of the open-loop transcoder are drift errors. Drift errors occur in BL stream because the motion-compensated frames in a GOP rely on the

reference frame, which gets modified during transcoding. As an example, imaging that frame is encoded as an I frame. The original value of pixel x is $f_I(x)$. The transcoder decreases the amount of data in the frame by applying coarser quantization, which introduce an error $e_I(x)$. The resulting value of pixel x is $\tilde{f}_I(x) = f_I(x) - e_I(x)$. Now, the following P frame uses this I frame as a reference, which means that the value of pixel x , $f_P(x)$ is not encoded directly. Instead, the difference $d_P(x) = f_P(x) - \tilde{f}_I(x)$ is used. The transcoder modifies the P frame introducing the error $e_P(x)$. The resulting value of pixel x is $\tilde{d}_P(x)$. $\tilde{f}_P(x)$ is calculated as follows:

$$\tilde{f}_P(x) = \tilde{d}_P(x) + \tilde{f}_I(x) = d_P(x) + f_I(x) + e_I(x) + e_P(x). \qquad (3.7)$$

So, the modification of a pixel value in frame P is a result of two errors.

The errors in predicted frames cannot be compensated by the enhancement layer in scalable video coding. The enhancement layer stores the difference between the original values and re-quantized values of the current frames only. It means that $e_I(x)$ can be compensated for frame I, but not for frame P, where only $e_P(x)$ could be compensated by EL. Consequently, the drift errors remain present in the scalable video made by an open-loop transcoder.

3.4.2 SNR closed-loop transcoder

In video transcoding, it is desirable that a video layer can be decoded correctly without any drift errors, hence the BL video layer should have its own drift-error correction motion-compensation loop. This transcoding architecture is shown in Figure 3.10.

Drift-error correction is provided as follows. The input coefficients to the enhancement layer are subtracted from those obtained via the inverse quantizer. This difference signal, which represents the information lost in the transcoding process, after conversion from the frequency domain to the pixel domain by an inverse discrete cosine transform (DCT), is then accumulated in a motion-compensation loop which receives motion vectors extracted from the incoming signal. This accumulated drift is then converted to the frequency domain and added as a correction to the next frame.

3.5 Decoding

A general scheme of a decoder that is capable of handling the proposed video streams is shown in Figure 3.11. Bitstreams of BL and EL are decoded separately and inverse quantized; after inverse DCT transformation and motion compensation both layers are combined. In the I-frame mode, the motion-compensation chain is not involved in processing an enhancement layer due to the absence of predicted frames. This allows a simplified implementation of the decoder. The decoder that

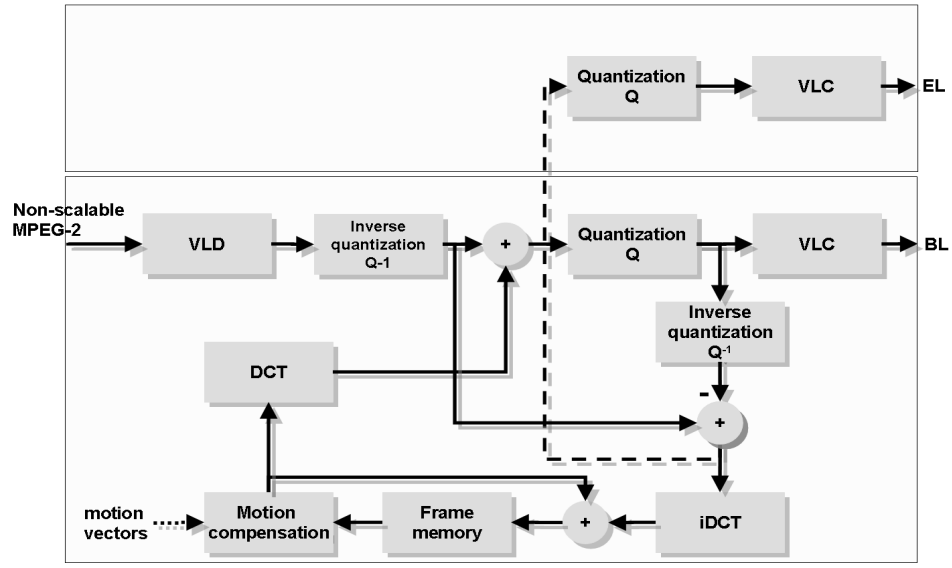

Figure 3.10. Closed-loop transcoder that produces SNR scalable video.

handles video streams created in the B-frame mode is also capable in decoding streams made in the I-frame mode. In the later case, the motion-compensation chain is unused.

The functionality outlined in the boxes of Figure 3.11 corresponds to a general implementation of MPEG-2 decoders with an external superposition of streams coming from the decoders (represented by the plus sign to the right of the boxes in Figure 3.11). The main condition for the superposition is the presence of layers: an enhancement layer can be used only if all previous enhancement layers are available. The summation module makes a summation of video frames according to a simple equation

$$R(x,y) = [\sum_{j=0}^{N_L} I_j(x,y)] - (N_L - 1) \cdot C, \tag{3.8}$$

where

- $R(x,y)$ is the resulting pixel in the output frame,
- $I_j(x,y)$ is the pixel at position x ,y in a frame of video stream number j (L_B is 0, $L_{E,1}$ is 1 , etc.),
- C is a constant, in our case equal to 128.

The choice of the constant in this case is determined by the specifics of MPEG encoding/decoding, which can be explained as follows. Pixel values of an uncom-

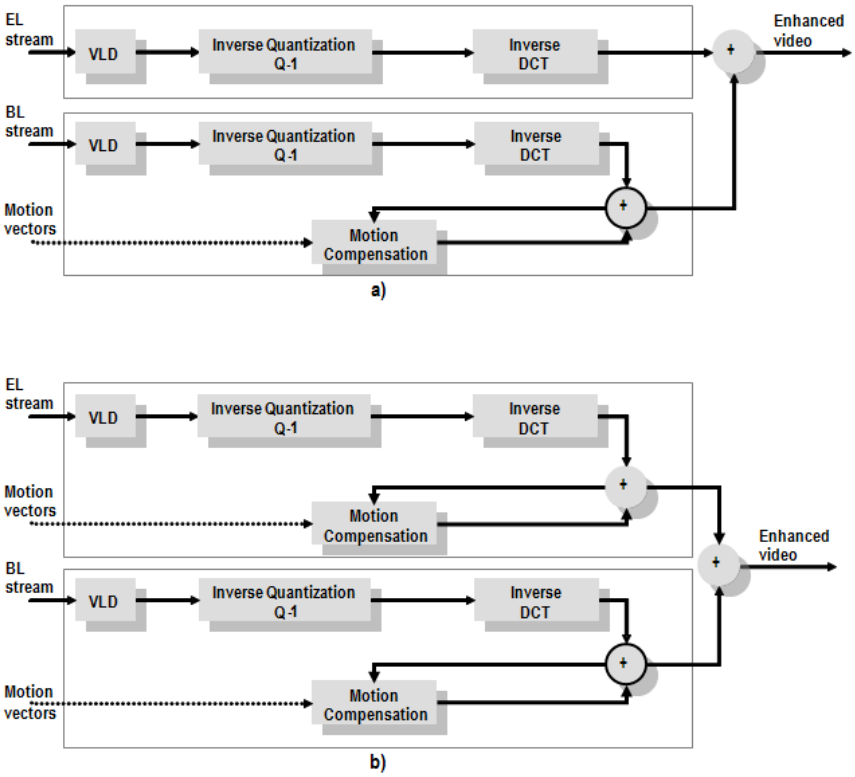

Figure 3.11. Principal schemes of a decoder: a) I-frame mode compatible, b) B-frame mode compatible.

pressed image have a range between 0 and 255 (8 bits). DCT performed on 8-bit pixel values produces 14-bit signed coefficient values. Subtraction of 128 from pixel values shifts the range to $[-128, 127]$, which, after DCT is performed, produces coefficient values within the range $[-1024, 1023]$. For encoding efficiency the shifting operation became an essential part of the MPEG-2 standard.

The modification of a pixel value through encoder and decoder is as follows:

1. original value f is shifted by 128, changing to $f_s = f - 128$,
2. after DCT the value is changed to $p = F^{DCT}(f_s)$,
3. after quantization the value is $p^{'} = F^{Q}(p)$,
4. the decoder performs inverse quantization to get $p^{''} = F^{iQ}(p^{'})$,
5. inverse DCT reproduces the original value and shifts it by 128 $f^{'} = F^{iDCT}(p^{''}) + 128$.

When the EL is created, it contains $r = p - p^{''}$. After quantization, an encoder produces $r^{'} = F^{Q}(r)$. The decoder, after all aforementioned steps outputs $R = F^{iDCT}(r) + 128$.

The summation function merges L_B and $L_{E,1}$

$$\begin{aligned} S &= F^{iDCT}(p^{''}) + 128 + F^{iDCT}(r) + 128 \\ &= F^{iDCT}(p^{''} + r) + 256 \\ &= f + 128. \end{aligned}$$

Apparently, during the summation the resulting value is increased by 128 as many times as many layers are decoded. The summation module therefore subtracts 128 times number of enhancement layers.

3.6 Conclusions

We investigated various scalability techniques and show that scalable video coding that is based on SNR principles is the most suitable for the developed system. We developed SNR scalable video-encoding technique that produces multiple layers. The layers can be processed by a standard non-scalable MPEG-2 decoder.

During the research and development, multiple alternative solutions were created:

- **Encoding technique**: two modes for creation of an EL are proposed. In I-frame mode an EL contains only I frames. In B-frame mode an EL consists

of GOPs that have IB_n structure. In Chapter 6 the two modes are compared to each other and the choice of parameter n is discussed.

- **Encoder implementation**: we proposed SNR-specific encoder as an alternative to an implementation based on cascade of non-scalable encoders. The comparison of the two approaches is presented in Chapter 6. In addition, the best layer configuration for given network conditions is a question, hence in Chapter 6 we study the influence of the layers split on the delivered picture quality.
- **Transcoder implementation**: in this chapter we discussed three approaches for making a transcoder – re-encoding, open-loop SNR transcoder and closed-loop SNR transcoder. The elaborated research and development of the transcoders are not a part of this thesis. Only open-loop SNR transcoder and re-encoding based transcoder were used within the scope of the thesis. A comparison of the quality of these two approaches is given in Chapter 6. An additional information regarding SNR transcoder can be found in [52].

The following expectations exist prior to the evaluation: a sender in a home network should have a transcoder that is based on the re-encoding approach. The re-encoding consists of decoding a MPEG-2 stream into raw video data by a standard non-scalable MPEG-2 decoder. Raw video data is, than, processed by a SNR-specific encoder that creates a scalable video.

As discussed in Chapter 1, the sender in a home network is seen by most CE manufacturers as a powerful PC-based solution. In this case the re-encoding approach seems to be the best option for making a transcoder. The advantages of the approach are:

- Low implementation complexity. The transcoding is done by a decoder-encoder couple. The encoder should comply with the scalability technique that is proposed in this chapter. The decoding can be done by any of-the-shelf MPEG-2 standard decoder.
- Adaptability to another input format. The decoder and encoder are not connected directly. So, replacing an MPEG-2 decoder by, for example, a MPEG-4 AVC decoder that outputs raw video data would allow the sender to take a MPEG-4 AVC stream as an input.

The choice of encoder implementation is explained as follows. Since we are not bounded by sender resources, the computational complexity of SNR encoder implementation that is based on cascade of non-scalable encoders plays no role. Moreover, the simplicity of implementation gives a significant advantage to the cascade-based approach. The SNR-specific encoder, on the other side, should produce a video of higher quality (under the same bit rate),because the cascaded-based

solution has more functions where a video data could be disrupted – cascaded iDCT/DCT processing and quantization operations reduce the PSNR values by about 0.1 – 0.5 dB (this is confirmed in Chapter 6). Facing a trade-off between video quality or implementation complexity we choose SNR-specific encoder.

4

Scalable video streaming

This chapter presents a streaming solution for real-time transmission of a scalable video over wireless/wired networks. The video data produced by the transcoder is sent over a heterogenous network that consists of at least one wireless link.

The basic requirements and assumptions regarding the streaming environment are explained in Chapter 1:

- A home network may contain a few nodes (e.g. an access point, a router).
- Links in the network may be wired or wireless, but the thesis' main target is a fully wireless network.
- Any data loss in between the video data producer (transcoder at the sender side) and consumer (decoder at the receiver side) should be transparent for and controlled by the streaming solution on the application level. Ideally, the data is sent without any losses through the transport layer.
- The data should be transported in a timely fashion, e.g. a sufficient amount of information should be available to the decoder, to keep the frame rate of the output video constant.
- The buffer size at the receiver side should be minimized and should not exceed values required to buffer one second of video data. The requirement on the buffer size originates from the request to use CE devices as receivers.

Due to the cost minimization, CE devices do not have sufficient memory resources to buffer large portions of an incoming video stream.

The major research question answered in the chapter is: *is it possible to create a video streaming solution that minimizes transmission data losses and improves protocol throughput by observing protocol behavior at the sender side*? The presented streaming solution observes changes in protocol buffer fullness to detect changes of network conditions. The information about the current network conditions is used to calculate an optimal amount of video data that goes to the protocol (per time interval) to avoid losses in the transport layer and to keep utilization of the bandwidth as high as possible.

4.1 Preliminary research

Figure 4.1 shows the basic idea of video streaming, where a streaming solution appears as a black box between the transcoder (at the application level) and network protocol. This section answers questions regarding the choice of protocol and details of the streaming solution.

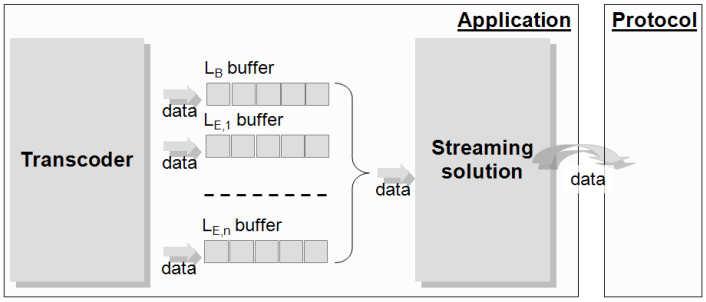

Figure 4.1. Principal positioning of the streaming solution in the system.

4.1.1 Protocol choice

Two protocols are considered for the purpose of video streaming – RTP and TCP. In the following subsection we discuss the pros and cons of these.

The discussion assumes that for wireless links a MAC retransmission mechanism [53], which allows automatic re-sending of a non-delivered packet for a (predefined) number of times, is used. This mechanism ensures a guaranteed transmission of data, but results in delays during transmission (which can be harmful for time-sensitive applications [54]). MAC retransmissions have been shown to negatively affect TCP performance [55, 56, 57] for multi-hop wireless networks. At the same time, experiments with home networks (few wireless clients connected to a single Access Point) show that MAC layer retransmissions improve UDP and

TCP performance [58, 59].

RTP

RTP is a natural choice for real-time video streaming. Poor reliability of the protocol is easily compensated by a MAC retransmission mechanism. A major drawback of using RTP for video streaming is the difficulty of controlling the data losses at the sender that occur due to congestion in the routers. Furthermore, if the connection between a sender and a receiver includes another node (for example, an access point or a router), data losses or delays on a network link between the node and a receiver are not easily observed at the sender.

Figure 4.2. Sender and receiver connected via an intermediate node.

Figure 4.2 illustrates the case where a sender and a receiver are connected via an intermediate node. The connection between the sender and the intermediate node is marked as Link A, while the connection between the intermediate node and the receiver is Link B. Let us consider two possible scenarios:

1. There are no data losses or significant delays on Link B. This occurs when Link A is a wireless channel and the intermediate node is an access point that is connected to the receiver by a dedicated ethernet cable. In the situation described, video data packets arrive at the intermediate node, are put in the node buffer and sent to the destination. Since there is always sufficient bandwidth and there is no competing application, the transmission over Link B suffers no losses or delays.
2. There are losses and delays on Link B. This occurs if the intermediate node is an access point that has a wireless connection to the receiver as Link B. If data is lost during the transmission on Link B, the resulting data stream is disrupted, or recovered upon retransmission on the MAC level. The latter case results in delays of data packets in the node buffer. With time, the buffer could overflow, resulting in a packet drop at the intermediate node. The loss influences BL packets or EL packets with equal probability. Moreover, the loss cannot be directly detected at the sender side, thus preventing error recovery at the sender.

The same reasoning can be applied to a connection with multiple intermediate nodes. Since the sender has control over the first link only, errors on later links or nodes are not directly detected at the sender. There are two possible solutions that allow the control of losses and delays for video data that is streamed by RTP:

1. create intermediate nodes that are aware of scalable video,

2. organize feedback from a receiver to the sender.

To bring scalable video awareness to nodes, software modifications will have to be made to nodes that were not designed for the purpose of scalable video streaming. Controlling losses and delays using RTP with a feedback channel from a receiver is much like using a TCP acknowledgement mechanism. For the moment, RTP has no retransmission capabilities.

The data loss by RTP is hidden from the sender and, as the result, cannot be detected and/or avoided by a streaming solution. Thus, the protocol cannot be used in the developed system.

TCP

TCP is the second option to be considered as a protocol for video streaming. TCP eliminates uncontrolled losses of data due to its 100% reliability. Despite the fact that TCP is a reliable protocol, it is rarely used for real-time video streaming. The reason for this is that TCP retransmits the data that is lost or corrupted during the transmission, which then results in transmission delays that are not tolerated in a real-time environment. Moreover, in case of packet loss, TCP reduces the transmission rate leading to even longer delays.

An advantage of TCP is that any losses or delays during transmission are visible at the sender side no matter on which link (A or B) they occur. Moreover, all actions required for recovery from the above-mentioned problems are incorporated in the protocol, which means there is no need for additional measures at application level. As a payoff, the reliability of TCP comes at the expense of higher bandwidth usage due to data retransmissions and acknowledgement transmission. If at some point in time the application tries to send more data than TCP can deliver (e.g. video bit rate larger than network bandwidth), the losses of data can still occur due to application/protocol buffer overflows. The reasons for an overflow are the following. When an application cannot write data to a protocol buffer because the buffer is full, the data can be discarded or maintained in the application buffer. The discarded data is irreversibly lost, so the majority of applications prefer to keep the data in the application buffer until the protocol buffer is emptied. If the protocol buffer is not emptied for a long time, the application buffer gets full. This is not a problem for applications that can delay the production of data. Video applications (especially those dealing with real-time input) must continue the production, so old data in the application buffer needs to be discarded to create space for new data.

Despite the two problems of TCP-based streaming, the absence of timely delivery and the possibility to loose data in application/protocol buffers, TCP guarantees absolute reliability of the delivery, which makes it the most suitable protocol for the developed streaming solution.

4.1.2 Single stream vs. Multiple streams

Single stream versus multiple streams delivery is another point of attention in developing a video streaming. The scalable video coding produces multiple layers of video data, with all layers composing a single video stream. The majority of modern solutions for scalable video streaming uses one connection per stream. The advantages of the approach are easy management of the layers during the transmission, the possibility of using unequal error protection for different layers, and the possibility to organize multi-path transmission. The disadvantage of the approach is the difficulty to support prioritization of layers during the transmission.

A single stream approach uses only one connection and a single data stream to transmit multiple layers. The video frames from the layers are written one by one to the stream in accordance with the layer's priorities. The L_B frames precede $L_{E,1}$ frames, $L_{E,1}$ frames precede $L_{E,2}$ frames, etc. Since TCP does not change the order of data arrival, it is guaranteed that the order of the layers at the receiver are exactly the same as they were at the sender.

4.2 Prioritization

Scalable video produced by a transcoder at the sender side can be sent over a network according to the importance of the frames, i.e. BL first and then ELs. BL frames are absolutely necessary to reproduce a video, while ELs only improve the quality of the video delivered by BL, from which their importance is derived. The relevant prioritization is handled at the application level in the following way. The layers of a scalable video are stored in separate buffers. Every layer consists of video frames split into a set of data packets. A scheduler is responsible for moving data packets from the application buffers into a protocol buffer, from where the packets are sent to their destination (see Figure 4.3). The scheduler reads

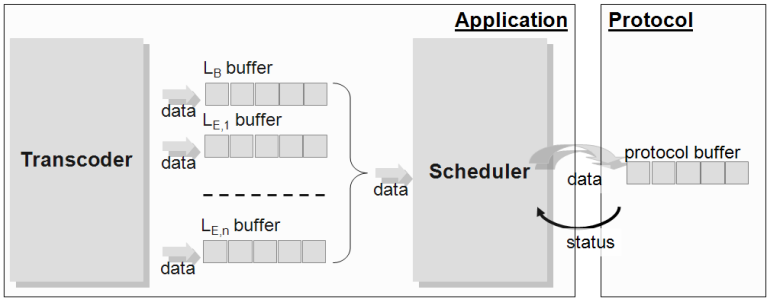

Figure 4.3. Streaming solution with prioritization scheduler.

packets from the BL buffer and, if there is no BL packet, from the EL buffers. Consequently, the BL part of a frame is always sent first and the rest of the frame

composed of the ELs layers is transmitted afterwards.

The scheduler verifies that there is sufficient space in the protocol buffer for data packets coming from an application buffer. If there is no space available, the scheduler does not take packets from the application buffers. Consequently, the protocol buffer fullness spreads into the application buffers. This mechanism is absolutely necessary because it enables buffer management at the application level which, to a large extent, decouples buffer management from the chosen communication protocol.

Once video packets are stored in the protocol buffer, the responsibility for the transmission of the data lies entirely with the protocol itself. The data is sent in a single TCP stream, so the order of the layers for a given frame does not change during the transmission.

4.3 Controlled losses – EL drop

In the previous sections we discussed the fact that the delivery of video data is guaranteed at the protocol level. If the network bandwidth drops below the video bit rate, the application buffers become full. The larger the buffers, the longer the periods of insufficient bandwidth can be hidden from the end user. The cost of the large buffers is increased latency (the time needed for a video frame to be transferred from the sender to the receiver).

Latency of more than 2 seconds, which corresponds to the buffering of 50 – 60 frames, is not acceptable in real-time video applications. Given the prioritization scheme described in the previous section, the fuller an application buffer, the longer the packets from the buffer wait for transmission, which leads to synchronization problems because the difference between the arrival of BL frames and the arrival of the EL frames increases. If the EL frame is not available to a decoder at the time the corresponding BL frame is being processed, transmission of that EL frame is pointless because it will be dropped at the receiver side. Moreover, if a BL frame is transmitted to a receiver with a significant delay, this frame is discarded at the receiver side because it is too old. To avoid this, we assign a timestamp to all packets in the application buffers so that the scheduler can check how long a packet has spent in the application buffer. If this time is longer than a given threshold, which depends on the amount of buffering in the system, the packet is dropped by the scheduler. Furthermore, if an application buffer is full when the application is attempting to write data to the buffer, the application removes any outdated packets.

As discussed in Chapter 2, preemption of frame transmission leads to an unwanted resource wastage. When during transmission of frame FE_1^i from enhancement layer $L_{E,1}$ frame FB^j from base layer L_B arrives to the application buffer, a

sender will send FB^j after transmission of the packets belonging to FE_1^i that are already placed in the protocol buffer. During the transmission of FB^j, FE_1^i data, left in the application buffer, may become outdated. If the scheduler would drop the outdated data, the resources which have been spent on transmission of FE_1^i are wasted. That may lead to a regretful situation, when the sender continuously uses resources to transmit only parts of frames from the EL. A simplified example of such situation is shown in Figure 4.4.

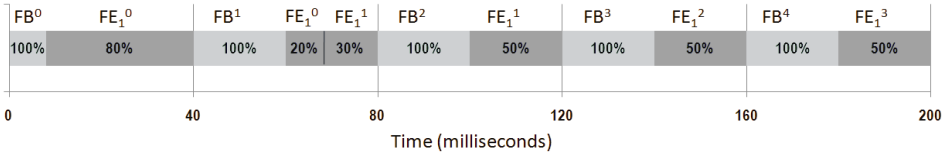

Figure 4.4. Timeline for transmitting a video data from the application buffers to the protocol buffer.

Figure 4.4 presents the timeline for transferring video data from the application buffers to the transmission-protocol buffer. It is assumed that the transcoder produces a scalable video consisting of two layers, L_B and $L_{E,1}$ with a frame period of 40 ms. The bit rates of the layers are 1 and 2 Mbps respectively. We set the maximum time that a frame can spend in the application buffer is two frame periods (80 ms). The network throughput is 2 Mbps. The protocol buffer size is 5 KB (which is adequate for a slow link with a very small round trip time). For the sake of simplicity, let us assume that:

- every frame in a layer has the same size,
- frames FB^i and FE_1^i are put in the application buffer concurrently,
- and data is transmitted in chunks of 2 KB.

The horizontal scale in Figure 4.4 represents time in milliseconds from the moment frames FB^0 and FE_1^0 (the first frames in L_B and $L_{E,1}$ respectively). In the figure, the operation of reading a part of a video frame from the application buffer and writing it into the protocol buffer is shown as a block that occupies time. Each block is marked with its frame number(on top). The percentage inside a block shows how much data of the frame was transferred in the given time slot. To improve visual clarity of the figure, the transfer of FB^0 is shown as if it occupies the first 8 ms of the first frame period. In reality, the whole frame is written to the protocol buffer instantly. In 8 ms the first 2 KB chunk of FB^0 is transmitted to a receiver, making space in the protocol buffer for data from FE_1^0. This is a startup behavior.

As shown in Figure 4.4, during the first frame period all of FB^0 and 80% of FE_1^0 is written to the protocol buffer. New frames, FB^1 and FE_1^1 are produced at

40 ms. FB^1 preempts FE_1^0. After the scheduler transfers FB^1 data to the protocol buffer, it continues with the rest (20%) of FE_1^0 and a part (30%) of FE_1^1. Frames FB^2 and FE_1^2 are produced at 80 ms, and FB^2 preempts FE_1^1. Only 50% of FE_1^1 is written into the protocol buffer in the third frame period, since at 120 ms FE_1^1 is preempted by FB^3. When FB^3 is transferred to the protocol buffer, the scheduler drops the rest of FE_1^1 because it is older than 80 ms. As a result, only 80% of FE_1^1 will be transmitted to the receiver, making FE_1^1 useless for the decoder. Consequently, the scheduler writes 50% of FE_1^2 to the protocol buffer in the fourth frame period, and switches to FB^4 in the fifth. After FB^4 is transferred to the protocol buffer, the scheduler drops the rest of FE_1^2 to deal with the first half of FE_1^3, whereas the second half of the frames will be dropped one frame period later. As a result, despite having a high utilization of the network bandwidth, the receiver never gets a full EL frame (except for FE_1^0). Thus, dropping an outdated part of a frame is not the best approach.

An alternative is to not remove the outdated packets, which is a work-preserving approach where instead of dropping the part of FE_1^i, transmission is continued. The extended life of FE_1^i increases the probability that frames from EL are dropped. Note that the work-preserving approach can result in multiple frames to be dropped as the result of a successful transmission of a single frame, whereas an approach that always drops a part of an outdated frame can result in no EL frames at all.

Let us discuss the difference between keeping/dropping outdated data in the application buffers. Figures 4.5 - 4.8 show an example behavior of buffers at the sender (application buffers) and at the receiver (decoder buffers). The settings, simplifications and assumptions are taken from the previous example (see page 57). The decoder at the receiver buffers 5 frames before the start of video processing (known as 'initial buffer fullness').

Figures 4.5 and 4.6 show the filling of the application buffers and the decoder buffers for the *no skipping* approach in which outdated packets are *not* removed. Since the bit rate of the L_B is lower than the available bandwidth, L_B frames are all transmitted successfully over the network within half a frame interval after their arrival at the protocol buffer. The second half of the frame interval is used to transmit $L_{E,1}$ frame packets. As implied by Figure 4.5, when a new L_B frame is available in the application buffer, it preempts the $L_{E,1}$ frame that is being transferred to the protocol buffer (for example, at 40 ms a new BL frame is available in the buffer, and it preempts the previous EL frame). As shown in Figure 4.6, at time 312 ms, the decoder starts processing frame number 3 while the transmission of the FE_1^3 frame is still in progress. As a result, the decoder only processes FB^3 , which gives rise to a video of low quality. The same happens with frame number 4. Later on, at 352 ms, the required FE_1^4 is also not available because a large part of the

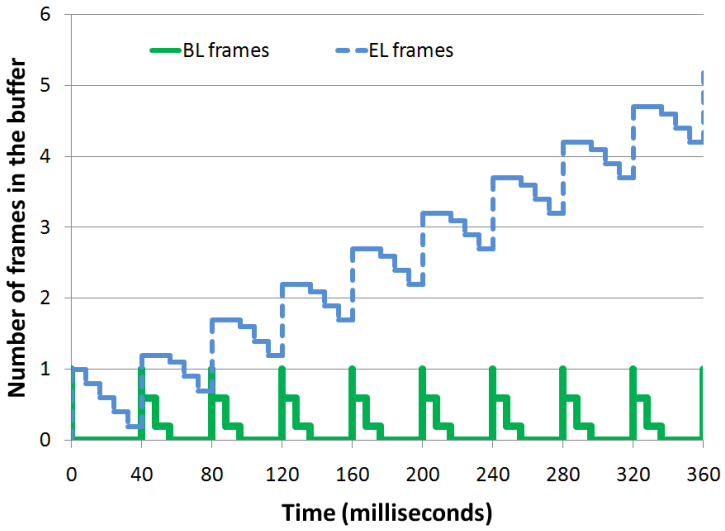

Figure 4.5. Application buffers at the sender. The *no skipping* approach. The dotted line shows number of frames in the EL buffer (layer $L_{E,1}$), the solid line shows number of frames in the BL buffer (layer L_B).

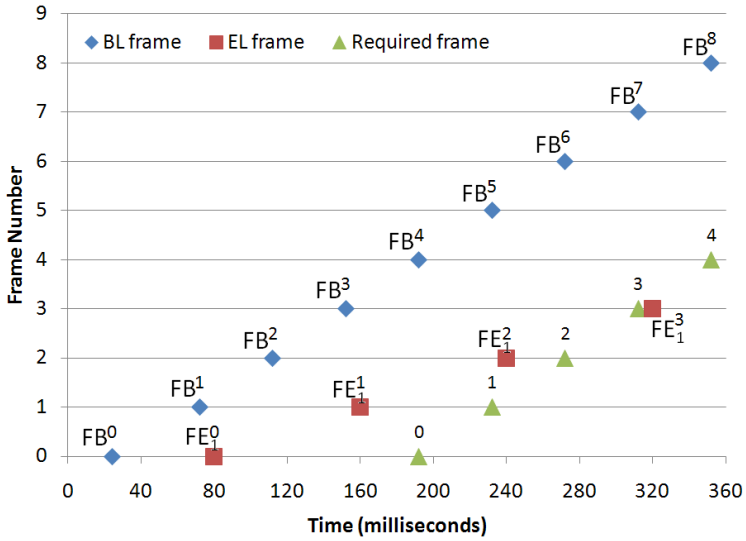

Figure 4.6. Frames of L_B and $L_{E,1}$ in the decoder buffers at the receiver. The *no skipping* approach.

previous frame interval was spent on the transmission of FE_1^3 despite the fact that FE_1^3 will never be used by the decoder. As a result, the bandwidth is wasted on outdated frames and from frame number 3 onwards the decoder processes only L_B frames.

The *skipping* approach is shown in Figure 4.7 and 4.8. Under the same conditions, the scheduler does not even start with the transmission of FE_1^3 because at the time of transmission of FE_1^2, the frame FE_1^3 has spent more than 80 ms in the buffer [1]. The result of FE_1^3 being skipped is that FE_1^4 is transmitted on time.

Removal of outdated packets is an easy and efficient solution for adapting the bit rate to suit the available bandwidth to scalable video. The solution is easy to apply to EL buffers, because frames in EL have no dependencies on each other, so any frame can be skipped. A situation may exist when dropping of all frames in EL cannot accommodate bandwidth variations, so during a deep drop in the available bandwidth the BL buffer may overflow. In contrast with EL frames, frames in BL have interframe dependencies and therefore require a more sophisticated skipping mechanism. We use the I-Frame Delay (IFD) method [60] for BL buffer management. IFD represents a temporal scaling technique. When the network bandwidth drops below the bit rate of BL, IFD decreases the bit rate of BL by dropping video frames in accordance with the importance of the frames for the rest of the video stream. A reasonably low amount of dropped frames might not be noticeable to the end user, yet the bit-rate reduction achieved is high enough to enable recovery from bandwidth fluctuations.

Figure 4.9 shows positioning of IFD at the sender side. IFD uses the frame type to guide the frame skipping process, thus implementing a Push-out Buffer technique as follows: when the application buffer is full, IFD first pushes the B frames out of the buffer and then, if still necessary (i.e. the bandwidth has dropped significantly for a longer period), it pushes out the P and I frames. By only dropping all B frames from a video stream we can make the resulting video stream fit into a bandwidth that is half the bit rate of the original stream, still preserving interframe dependencies. The price that has to be paid for this is a reduced frame rate – in the case of an $I(B_2P)_yB_2$ GOP structure, if all B frames are dropped this would generally lead to $\frac{1}{3}$ of the original frame rate.

4.4 Controlling data flow – transcoding

So far we discussed only reactive measures to handle bandwidth fluctuations. The dropping techniques decrease the amount of data that is given to TCP. As mentioned above, a long period of bandwidth decrease may lead to the complete loss (due to the dropping) of EL and significant loss of BL frames. To avoid that, a

[1] As mentioned above, the system buffers only 2 frames, which is 80 ms.

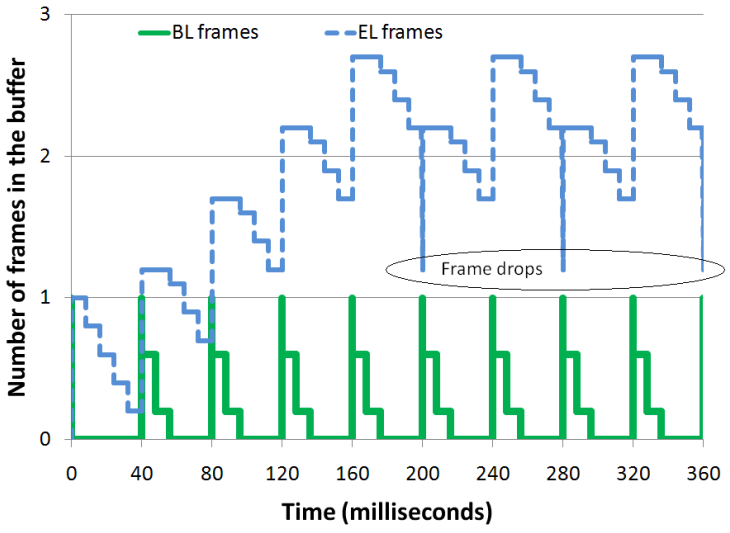

Figure 4.7. Application buffers at the sender. The *skipping* approach. The dotted line shows number of frames in the EL buffer (layer $L_{E,1}$), the solid line shows number of frames in the BL buffer (layer L_B).

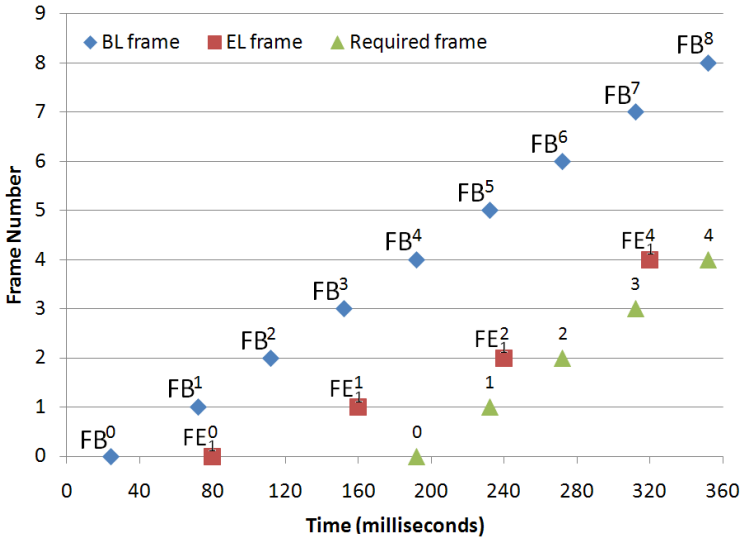

Figure 4.8. Frames of L_B and $L_{E,1}$ in the decoder buffers at the receiver. The *skipping* approach.

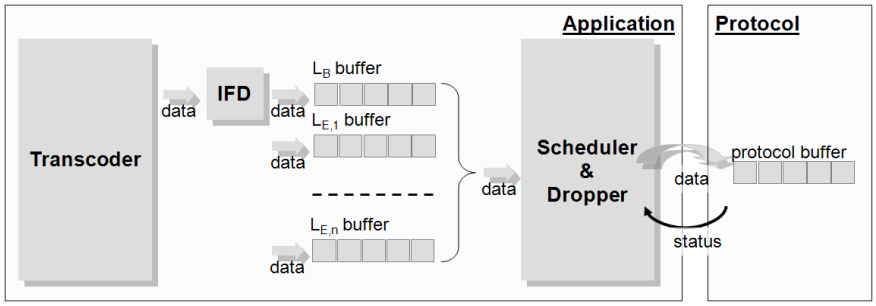

Figure 4.9. Streaming solution with prioritization scheduler, dropper of outdated frames and IFD algorithm.

proactive technique should be used that reconfigures the transcoder when a drop in the bandwidth is detected, so the transcoder produces smaller video layers or a lower number of layers. Making a smaller layer means reducing the amount of bits that is allocated to individual frames, thus making their transmission easier and avoiding buffers getting full.

Figure 4.10 shows the complete scheme of the streaming solution that is capable to anticipate the network behavior and react to it by reducing the amount of produced video data. The scheme has two additional core components: a network appraiser and a layer configurator.

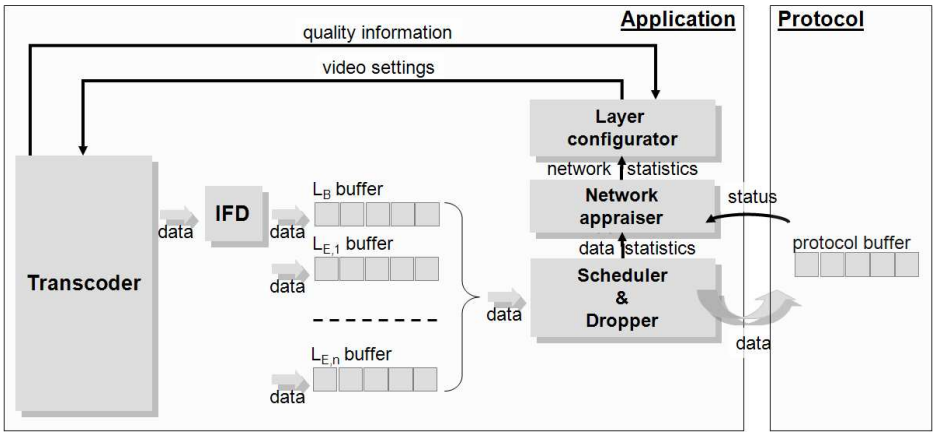

Figure 4.10. Streaming solution with the network appraiser, layer configurator and scheduler&dropper components.

4.4.1 Network appraiser

The main task of the network appraiser is to calculate the current bandwidth and bandwidth variations. As shown in the previous section, there should be no data losses at the protocol level during transmission. All losses are 'transformed' into delays of data. In this case, buffer fullness is a good indication of network conditions. The basic principle is that if packets are sent at a rate larger than the available bandwidth, the queuing delays increase, so the time that the packets stay in the buffer increases. If a buffer is becoming empty or is already empty, the current network conditions allow more data to be sent than is currently being sent. An increase in buffer fullness suggests that an application is sending more data than is possible with the current network state.

The network appraiser uses a simple formula to calculate the momentary available bandwidth (B_t), which is an estimation of the effective available bandwidth, B_t^R , at the time t:

$$B_t = \frac{(S_{t-\delta t} - S_t) + W_{(t-\delta t;t]}}{\delta t} \ . \tag{4.1}$$

Parameters S_t and $S_{t-\delta t}$ denote the protocol buffer fullness (size of data in the protocol buffer) at the moments t and $t - \delta t$ respectively. Variable $W_{(t-\delta t;t]}$ is the amount of data written to the buffer in the time period $(t - \delta t; t]$, which can be calculated as $W_{(t-\delta t;t]} = R \cdot \delta t$, where R is the bit rate of the data stream. In general definition, R is the bit rate of the data that comes from the application buffer to the protocol buffer. Assuming that the scheduler is not dropping frames, R is the bit rate coming from the transcoder.

Under certain conditions, B_t represents the minimal value of the available bandwidth and not the estimation of the effective available bandwidth. That can be explained as follows. Let us assume that $B_t = B_t^R$. From Equation (4.1),

$$S_t = R \cdot \delta t + S_{t-\delta t} - B_t^R \cdot \delta t. \tag{4.2}$$

If the effective available bandwidth, B_t^R , is less than $R + \frac{S_{t-\delta t}}{\delta t}$, the value of S_t is positive or zero. In the case when $B_t^R \cdot \delta t > R \cdot \delta t + S_{t-\delta t}$, the buffer fullness has a negative value which means that the protocol sent more data than was available in the buffer. In reality, after transmitting $R \cdot \delta t + S_{t-\delta t}$ of data the protocol stops until new data arrives at the protocol buffer. The absence of data at the buffer means that $S_t = 0$. So, from Equation (4.1), $B_t \cdot \delta t = R \cdot \delta t + S_{t-\delta t}$, which means that $B_t < B_t^R$. To compensate for the unaccounted bandwidth, an additional value B^e is added to the measured bandwidth if $S_t = 0$. The value of B^e is chosen experimentally as discussed in Chapter 6.

The overall formula to calculate the momentary available bandwidth is

$$B_t = \begin{cases} \dfrac{(S_{t-\delta t} - S_t) + W_{(t-\delta t;t]}}{\delta t} & if\ S_t > 0 \\ \dfrac{S_{t-\delta t} + W_{(t-\delta t;t]}}{\delta t} + B^e & if\ S_t = 0 \end{cases} \tag{4.3}$$

The network appraiser continuously calculates the available bandwidth keeping the history of values over a period of time called *observation window*. A zero-phase digital filtering [61] is used on the calculated values to filter out 'noise', i.e. short-term fluctuations (see Figure 4.11).

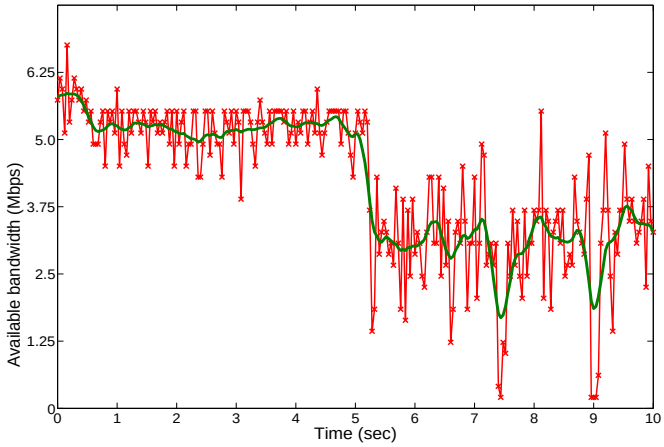

Figure 4.11. An example of filtering of a calculated available bandwidth.

The available bandwidth changes continuously during the transmission. We assume that during a short time interval (less than 1 second) the bandwidth fluctuates around a given average. So, the set of calculated values of the average bandwidth is approximated by a normal distribution, where mean μ_B is the average bandwidth and σ_B is the standard deviation. These two parameters are the output of the network appraiser.

4.4.2 Layer configurator

The layer configurator continuously receives information about the available bandwidth from the network appraiser and, with that knowledge, examines the space for possible video stream configurations (number and bit rate of layers), searching for a configuration that delivers the highest possible quality under the given network conditions. Figure 4.12 presents a detailed view of the layer configurator and its environment. The layer configurator consists of three parts: loss estimator, quality

estimator and decision maker.

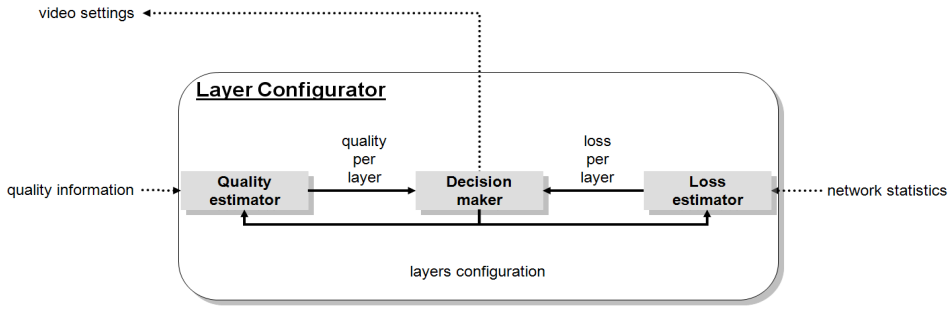

Figure 4.12. Principal scheme of the layer configurator (info flow). Based on the quality information from the transcoder and network statistics from the network appraiser, the layer configurator provides the transcoder with video settings, namely, number and bit rate of layers.

Loss Estimator

The loss estimator calculates the probability of data loss (due to dropping outdated frames from the application buffer) for each layer of a layer configuration under given network conditions. The description of the network conditions is based on the current value and the variations in the available bandwidth as delivered to the loss estimator by the network appraiser.

The loss probability is calculated by matching the estimated performance of the link (in terms of bandwidth changes) to the video traffic model that describes changes in video frame sizes for the given layer configuration. We base the traffic model on the statistical model [62], where the distribution of video frame sizes is approximated by a normal distribution [63, 64].

We denote R^j as the average bit rate of layer j ($j = 0$ for L_B , $j = k$ for $L_{E,k}$). We assume that R^j is a random, normally distributed variable with mean μ_j and standard deviation σ_j. The mean value corresponds to the bit rate of the layer, while the standard deviation depends on the encoding algorithm and should be supplied by the video data producer (for example, for an MPEG-2 TM5 encoder with a GOP (12,3) the standard deviation is close to $\frac{1}{4}$ of the mean value).

As a result, the probability ξ_N that N layers can be transmitted successfully at the current time is

$$\xi_N = P(\sum_{j=0}^{N_L} R^j \leq B_t), \tag{4.4}$$

or

$$\xi_N = P(B_t - \sum_{j=0}^{N_L} R^j \geq 0), \tag{4.5}$$

where P is a probability.

Taking into account that B_t and R^j are normally distributed values, the difference between them is also a normally distributed value $D^N = B_t - \sum_{j=0}^{N_L} R^j$ with mean

$$\mu_{D^N} = \mu_B - \sum_{j=0}^{N_L} \mu_j \tag{4.6}$$

and standard deviation

$$\sigma_{D^N} = \sqrt{\sum_{j=0}^{N_L)} \sigma_j^2 + \sigma_B^2}. \tag{4.7}$$

The probability P that the random variable D^N takes a value greater than 0 can be calculated using a cumulative probability function F_{D^N} for D^N ,

$$P(D^N > 0) = 1 - F_{D^N}(0). \tag{4.8}$$

Cumulative probability function $F_{D^N}(0)$ is calculated as

$$F_{D^N}(0) = \frac{1}{2} \cdot (1 + erf(\frac{-\mu_{D^N}}{\sigma_{D^N} \cdot \sqrt{2}})). \tag{4.9}$$

From Equations (4.5), (4.8), and (4.9), the probability that N layers can be transmitted successfully is calculated as

$$\begin{aligned} \xi_N &= 1 - F_{D^N}(0) \\ &= 1 - \frac{1}{2} \cdot (1 + erf(\frac{-\mu_{D^N}}{\sigma_{D^N} \cdot \sqrt{2}})) \\ &= \frac{1}{2} \cdot (1 - erf(\frac{-\mu_{D^N}}{\sigma_{D^N} \cdot \sqrt{2}})). \end{aligned} \tag{4.10}$$

Furthermore, based on ξ_N we may calculate the probability E_K that only a certain number of layers can be transmitted (for example, only L_B or only L_B and $L_{E,1}$). $K = -1$ means no layers, $K = 0$ means only L_B , $K = 1$ means L_B plus $L_{E,1}$, etc. The probability that only K layers are transmitted is the probability of the successful transmission of K layers minus the probability that $K+1$ layers can

be successfully transmitted

$$\begin{aligned} E_K &= \xi_K - \xi_{K+1} \\ &= (1 - F_{D^K}(0)) - (1 - F_{D^{K+1}}(0)) \\ &= F_{D^{K+1}}(0) - F_{D^K}(0). \end{aligned} \tag{4.11}$$

For $K = -1$, $F_{D^K}(0) = 0$, since $\xi_{-1} = 1$ as follows from Equation (4.4). If the total number of layers is M , $F_{D^{K+1}}(0) = 1$ if $K \geq M$ because the probability to successfully transmit more layers than are available is zero ($\xi_{K+1} = 0$).

The loss estimator returns probabilities for the successful transmission of a certain number of layers, E_K , for every given layer configuration as shown in Table 4.1. For example, the probability that only L_B frames will be transmitted under configuration 2 is 3% , whereas the probability that only L_B and $L_{E,1}$ frames will be successfully transmitted under the same configuration is 93%.

	Bit rate [Mbps]			**Probability**			
Config.	L_B rate	$L_{E,1}$ rate	$L_{E,2}$ rate	none	only L_B	only L_B & $L_{E,1}$	all layers
1	1	2	2	0.00	0.00	0.50	0.50
2	2	2	2	0.00	0.03	0.93	0.04
...	...	...	...	...	...	...	...
n	3	2	1	0.00	0.50	0.45	0.05

Table 4.1. An example of the results of the loss estimator.

Quality estimator

The quality estimator establishes a relation between a layer configuration and the objective quality that is delivered for the configuration. The relation depends on the coding technique that is used and the estimator therefore uses bit-rate/quality dependency (we refer to the dependency as 'quality mapping') derived from the transcoder.

An example of a quality mapping for the base layer is shown in Figure 4.13. The horizontal scale represents input bit rate, the vertical scale represents the quality of the output stream expressed in PSNR. For the sake of simplicity we limit maximal quality because in cases where there is no difference between the input and the output streams, the PSNR value is equal to infinity. Different lines within the figure correspond to a particular bit rate of the output stream. As shown in Figure 4.13, the quality is at maximum when the output bit rate is higher than the input bit rate. As soon as the output bit rate drops below the value of the input bit rate, the quality goes down. Quality as a function of input and output bit rates also depends on the encoding parameters of the original stream and on the content of the stream. The transcoder should, therefore, have a pre-defined quality mapping

for an “average” case. This mapping is used every time when the transcoder starts processing a new stream and is updated by the transcoder at run time.

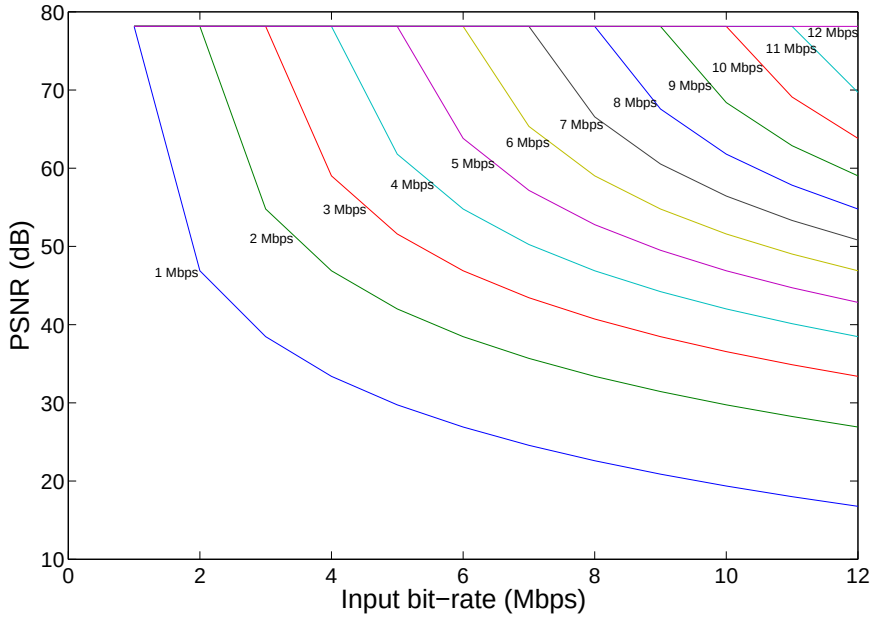

Figure 4.13. An example of transcoder quality mapping for L_B.

The quality of the video produced by the transcoder depends on output and input bit rates of a video. If the input bit rate of a video is fixed, the quality delivered by L_B is a function of the L_B bit rate.

The quality is measured by the transcoder itself and is expressed in PSNR. As follows from Equation (A.2), the PSNR is based on the MSE value, which is a distortion introduced to the video by the transcoder. We denote MSE of L_B with bit rate R_B as $d(R_B)$. Modelling of rate and distortion characteristics has been frequently used within bit-rate control schemes [65, 66, 67]. From a number of approaches described in the literature the exponential model is among the most popular due to an easy computation and good fit characteristics. The exponential model describes the relation between bit rate of the video data R_B and distortion of the data $d(R_B)$ as

$$d(R_B) = \alpha + \beta \cdot \log(\frac{1}{R_B}). \tag{4.12}$$

Equation (4.12) describes distortion as a function of output bit rate of the transcoder. The coefficients α and β are different for various input bit rates of a video and, moreover, for every type of video content. We assume, however, that

for a single video the input bit rate and the major characteristics of the video (pixel resolution, color resolution, etc.) stay the same.

Every time the transcoder starts processing new video material, the quality estimator resets α and β coefficients in the rate-distortion function. With every new frame, the transcoder communicates the MSE of the frame to the quality estimator, which calculates α and β using a least squares error method (the details can be found in Appendix D). Using Equations (4.12) and (A.2) the quality estimator is able to predict the average $PSNR$ value of a video stream containing BL with bit rate R_B.

The quality mapping for an enhancement layer shows what quality improvement is achieved by an $L_{E,i}$ of a given bit rate. The mapping is more complex than for the L_B since the quality depends on three variables: input bit rate, output bit rate for L_B and output bit rate for the $L_{E,i}$.

It is important to mention that all frames in the EL have the same coding type (I or B frames, as described in Chapter 3). As a result, two enhancement layers with bit rates $R_{E,1}$ and $R_{E,2}$ give rise to the same quality improvement as a single enhancement layer with bit rate $R_{E,1} + R_{E,2} - R^O$, where R^O is the overhead for having an additional enhancement layer (the value of R^O is evaluated in Chapter 6). It is, therefore, sufficient to have a quality mapping for a single enhancement layer case in order to estimate quality improvements by any number of ELs.

An additional distortion is introduced to a video by transcoding it into BL and one or more ELs instead BL only, namely:

$$\delta d(R_B, R_{ELs}) = \tilde{\alpha}(R_B) + \tilde{\beta}(R_B) \cdot \log(\frac{1}{R_{ELs}}), \tag{4.13}$$

where

$$R_{ELs} = \sum_{i=1}^{N_L} R_{E,i} - (N_L - 1) \cdot R^O. \tag{4.14}$$

The image improvements brought by an EL with a certain bit rate to a base layer depend also on the bit rate of the base layer. As a result, coefficients $\tilde{\alpha}(R_B)$ and $\tilde{\beta}(R_B)$ are calculated for different values of R_B.

Every time the transcoder starts processing new video material, the quality estimator resets $\tilde{\alpha}(R_B)$ and $\tilde{\beta}(R_B)$ coefficients for all R_B. With every new frame that is encoded into the BL and one or more ELs, the transcoder communicates MSE of the frame to the quality estimator. The MSE is calculated based on BL only or on the sum of BL and one or more ELs. The quality estimator uses the input to recalculate $\tilde{\alpha}(R_B)$ and $\tilde{\beta}(R_B)$.

The final output of the quality estimator is

$$PSNR = 20 \cdot \log_{10} \frac{255}{\sqrt{d(R_B)+\delta d(R_B,R_{ELs}))}}$$

$$= 20 \cdot \log_{10} \frac{255}{\sqrt{\alpha+\beta\cdot\log\left(\frac{1}{R_B}\right)+\tilde{\alpha}(R_B)+\tilde{\beta}(R_B)\cdot\log\left(\frac{1}{\sum_{i=1}^{N_L} R_{E,i}-(N_L-1)\cdot R^O}\right)}} \quad (4.15)$$

Table 4.2 shows an example of the quality estimator results. L_B quality is the PSNR value (in dB) for video that is delivered by the BL alone. The $L_{E,1}$ quality and $L_{E,2}$ quality show quality that is delivered by these layers in combination with the preceding layers. In Table 4.2, L_B from configuration 1 delivers video quality of 28 dB with respect to the original video, while the L_B and the $L_{E,1}$ from the same configuration yield video quality of 33 dB.

	Bit rate [Mbps]			Quality [dB]		
Config.	R_B	$R_{E,1}$	$R_{E,2}$	L_B quality	L_B & $L_{E,1}$ quality	L_B & $L_{E,1}$ & $L_{E,2}$ quality
1	1	2	2	28	33	36
2	2	2	2	35	39	41
...	...	...	...	...	...	...
n	3	2	1	40	42	43

Table 4.2. An example of the results of quality estimator.

Decision maker

The task of the layer configurator is to choose an optimum number of layers with optimum bit rates for the given network conditions. This task is done by bringing together results from the loss estimator and quality estimator in a decision-making module. Figure 4.14 presents the decision-making algorithm for three-layered scalable video.

The algorithm inspects various bit rates for the layers using the following parameters:

$Maximal\_Rate$ is the maximum bit rate for the sum of the layers. The rate is chosen as the maximum of the network throughput (e.g. 6 Mbps for 802.11b).

$Minimal\_Rate$ is the minimal bit rate for L_B that is possible for the given video settings. As an example, a Standard Definition video stream should have a minimal bit rate in the region of 0.75 Mbps, which is needed for the video stream syntax alone.

$Step\_Rate$ is the increase step for going through the possible bit rates for the layers. A general practice is to choose an increase that makes improvements to

```
Best_Quality = 0;
Chosen_Configuration = not_chosen;
[μ_B, σ_B] = Network_Appraiser→Update_Bandwidth();
FOR (R_B = Minimal_Rate; R_B < Maximal_Rate; R_B += Step_Rate)
{
    FOR (R_E,1 = 0; R_E,1 + R_B < Maximal_Rate; R_E,1 += Step_Rate)
    {
        FOR (R_E,2 = 0; R_E,2 + R_E,1 + R_B < Maximal_Rate; R_E,2 + =Step_Rate)
        {
            [ξ_0, ξ_1, ξ_2] = Loss_Estimator→Calculate_Probabilities(R_B,R_E,1,R_E,2);
            [Q_0, Q_1, Q_2] = Quality_Estimator→Estimate_PSNR(R_B,R_E,1,R_E,2);
            Quality = Calculate_Configuration_Quality([ξ_0, ξ_1, ξ_2],[Q_0, Q_1, Q_2]);
            IF (Quality > Best_Quality)
            {
                Best_Quality = Quality;
                Chosen_Configuration = [R_B, R_E,1, R_E,2];
            }
        }
    }
}
```

Figure 4.14. Decision-maker algorithm in pseudo-C.

the quality of the video visible. The experimental evaluation of the influence of the $Step\_Rate$ value on the system performance is given in Chapter 6.

Quality Q^i of configuration i is transmitted under the current network condition and is calculated as

$$Q^i = \sum_{j=0}^{N_L} Q_j \cdot E_j, \tag{4.16}$$

where N_L is the number of layers in the configuration ($N = 0$ means only the BL), Q_1 is the quality achieved by the L_B , Q_k if $k \geq 1$ is the quality achieved by the sum of all layers from L_B to $L_{E,k}$, E_j is the probability to transmit only j layers ($j = 0$ is only the L_B). The configuration n where $Q^n = max_{i=0..N_C}(Q^i)$, where N_C is the number of configurations, is considered to be the best for the current network conditions and it is communicated to the transcoder.

4.5 Conclusions

In this chapter, we have presented a streaming solution that enables streaming of a scalable video over a wireless/wired link. The approach is based on TCP streaming and therefore shows no data losses during the transmission.

Priorities can be assigned to layers of scalable video coded material to guarantee that transmission of the BL is not hindered by transmission of the ELs, and the higher EL has no influence on the transmission of the lower, more important, ELs. Due to the single-stream approach, the prioritization scheme is valid over the whole transmission path and does not depend on the network environment.

The lossless transmission results in potentially high transmission delays, when the bandwidth decreases. Two mechanisms are introduced to handle bandwidth variations – *frame dropping* handles short term bandwidth drops allowing an immediate decrease in the amount of data that is offered to the protocol, whereas *transcoding* handles long term bandwidth variations changing the amount of the produced video data. Frame dropping represents a reactive approach towards network-condition changes, while a transcoding-based approach executes a full spectrum of activities including network-bandwidth estimation, loss estimation and quality estimations for the video being transmitted. The major advantage of the approach is that all functionality can be fully implemented at the sender side.

Chapter 6 presents an extensive evaluation of the suggested solutions, focusing on the influence of internal parameters on the behavior of the solution as well as on the applicability of the solution under various network conditions.

The proposed streaming solution is not video-encoding specific and can be used with other (scalable) video coding techniques.

5

Controlling a network terminal

This chapter presents a resource management technique that employs a budget preservation approach for a scalable video-decoding application. The developed technique optimizes user-perceived quality by taking into account the available input data and available processing power. The quality is optimized by smoothing the quality fluctuations and preventing deadline misses. The optimization strategy is created offline by means of a Markov Decision Process. The algorithm of the strategy creation is, for a large part, based on the work of Clemens Wüst [34].

Within the scope of this thesis, the development of the resource-management technique focuses only on processor resources. The management of other system resources (bus, memory, etc.) has not been studied.

5.1 Preliminary research

The video-decoding application on the terminal is responsible for receiving scalable video data, decoding and merging frames of the scalable video into frames of uncompressed video that are ready to be rendered by the terminal. The reception of the video data, as described in Chapter 2, consists of reading data from the network protocol buffer, de-packetizing data, splitting the data stream into the set of layers of the scalable video and putting these layers into the buffers of the scalable

video decoder. This is the responsibility of the network reader, which is depicted in Figure 5.1.

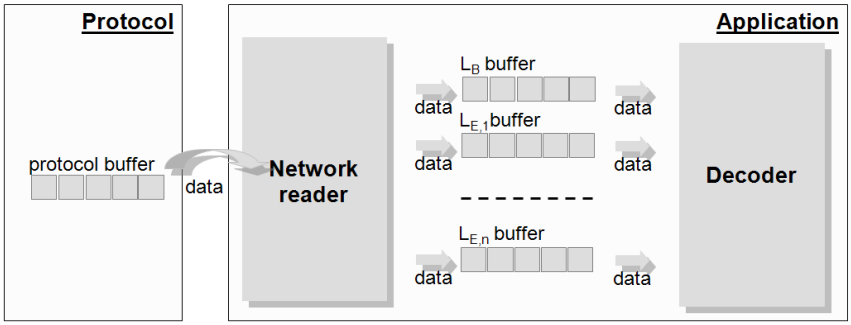

Figure 5.1. Overview of the network reader.

As discussed in Chapter 1 it is not guaranteed that the video-processing application can use all the device resources. Therefore, the developed video-decoding application should be able to perform even if the available resources are not sufficient to decode all incoming video data at the highest quality. Moreover, the application should be prepared to tolerate resource availability changes during runtime [1].

The successful functioning of the video-processing application assumes that a terminal has basic components for the resource management – budget scheduler and quality manager. Figure 5.2 shows an example organization of the resource management system. The quality manager is responsible for the definition of the priorities of various applications running on the terminal and, consequently, for assigning resources to every application. The resource assignment is then communicated to the budget scheduler, which takes care that the resources assigned to the application are used solely by the application and, at the same time, an application does not use more resources than are assigned to it.

It is assumed that the video-processing application runs only when the assigned resources are sufficient for the seamless processing of a BL at the minimal bit rate. In the worst case, that is an amount of resources that allows to decode an *empty* (black picture) frame from a BL every frame period. The frame period is the time between displaying two successive frames. In the general case, if a frame is not processed during the frame period, it is late for a display and, therefore, it is considered useless. The time when a frame processing must be complete is a deadline for the frame. A deadline-miss defines the state of the system when a frame is not decoded before its deadline.

[1] Although practice shows that the changes in resource allocations happen rarely, less than a few per hour.

The developed application consists of two major parts (excluding the network-reader component that always executes at the highest priority and, therefore, cannot be managed by the resource management system) – a controller and a decoder. The decoder is responsible for processing layers of a scalable video and merging them frame by frame for future display. The controller is continuously getting input from the budget scheduler regarding the utilization of the resources and changes the resource consumption of the decoder by modifying the decoder's settings, so the resources are not exceeded. The budget scheduler provides resources to applications in terms of budgets. Budget is the amount of resources that the application can use during a certain time period. The budget assigned to a video-processing application is specified with respect to the frame period. A budget of 50% assigned to a decoder that processes one frame every 40 ms says, in fact, that the decoder has 20 ms of exclusive resource usage for processing a frame. The time representation of the resource budget is referred to as a time budget.

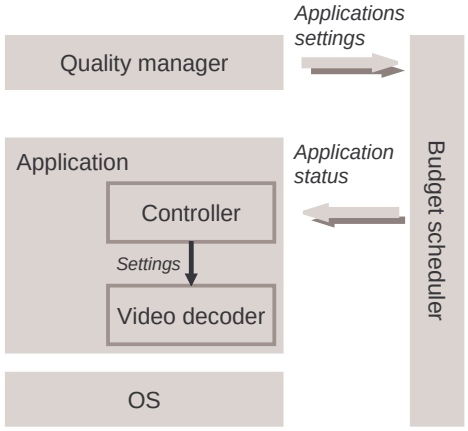

Figure 5.2. Resource management on a terminal.

The decoder can adjust the resource consumption in two ways: change the accuracy of the processing (e.g. lower accuracy of inverse DCT transformation) or change the amount of data that is processed (e.g. using fewer layers of the scalable video).

Changing the accuracy of the processing is only possible when the processing is done in software. Many CE devices on the market, however, use hardware modules to perform the most computationally intensive tasks or even the entire decoding process. Having a hardware module with a fixed functionality disallows any changes in the processing algorithm, making a technique that relies on such changes useless. The approach that exploits reduction of the amount of data that is sent to the decoder suits all range of the devices – with only software, only

hardware, or mixed software/hardware decoders.

The decoder can be manipulated by setting the number of processed layers to adjust its resource consumption. Since the quality of the decoder output depends directly on the number of processed layers, the quality of the output is a function of the resource consumption. An algorithm that trades resources for quality is a scalable video algorithm as described in Chapter 1.

The decoder in Figure 5.3 is organized as a scalable video algorithm, where the number of processed layers is an internal parameter that influences the resource consumption and output quality of the decoding. Processing of L_B requires the least amount of resources, but produces a picture of the lowest quality. When the decoder processes $L_{E,1}$ and merges it to L_B , the resource consumption increases so as the quality of the output picture. The processing of the higher ELs provides increasingly higher quality of the output, while consuming more resources.

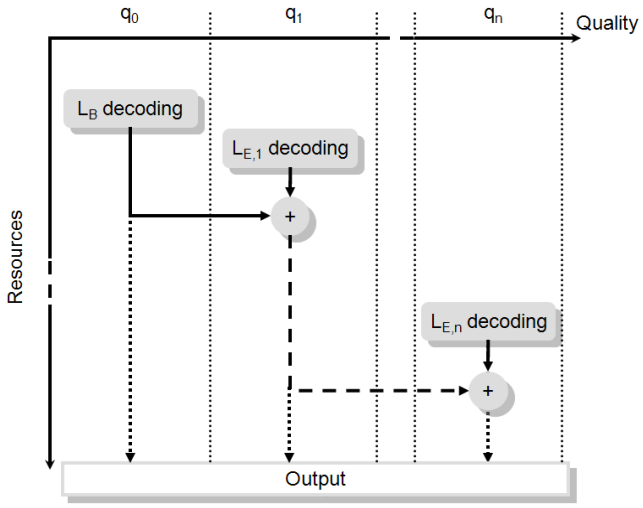

Figure 5.3. Decoder of SNR scalable video.

The relation between the resource consumption and the number of processed layers depends, for the lager part, on the implementation of the decoder. Decoding of a scalable video requires the following components (as shown in Figure 3.11): variable length decoding, de-quantization, inverse DCT and motion compensation. The influence of these components on the overall resource consumption is discussed below. As mentioned above, the only considered resource is a processor.

- **Inverse DCT** is one of the most computationally demanding components. The transformation is performed on a block of data indifferent of the content of the block. Therefore, the resource consumption of the component dur-

ing the processing of the video is linearly dependent on the number of data blocks that are processed. The latter is defined by dimensions of the video picture. As the result, the component uses

$$R_{iDCT} = \alpha_1 \cdot N_L \tag{5.1}$$

resources, where N_L is number of layers of a scalable video and α_1 is a parameter that depends on component implementation and execution platform.

- **Motion compensation** is another high resource-demanding component. The motion compensation consists of simple block reading and occasional filtering operations that are performed on blocks of data. So, the number of operations linearly depends on the number of blocks that should be processed. The latter depends on the frame types. For example, I frames have no predicted frames, so the motion compensation is not performed during processing of these frames. On the other hand, B frames usually contain many predicted frames and therefore, the motion compensation is involved intensively. In the case of scalable video coding, the motion-compensation involvement and, thus, resource consumption for decoding the BL and EL is different. As described in Chapter 3, the EL is created using only I frames or I and B frames. If the I-frame mode is used to encode EL, the resource consumption of the motion compensation for processing EL is zero, so it is only defined by the BL. Otherwise, the motion-compensation involvement is calculated based on B frames, which are the major part in the IB_n GOP of an enhancement layer. It is shown in [68] that bit rate has little influence on the number of motion-compensated blocks. As a result, the resource consumption of motion-compensation component is

$$R_{MC} = \begin{cases} \alpha_2 & if \quad \text{I}-frame \quad mode \\ \alpha_2 + (N_L - 1) \cdot \alpha_3 & if \quad \text{B}-frame \quad mode \end{cases}, \tag{5.2}$$

where α_2 is the amount of resources needed to process BL and α_3 is the amount of resource needed to process an EL. Parameters α_2 and α_3 depend on component implementation and execution platform.

- **De-quantization** is an insignificant component from resource-consumption point of view. Moreover, the de-quantization is often performed with a variable length decoding component. For the current analysis de-quantization is neglected.
- **Variable length decoding** is the only component that has high demands for resources and shows a clear dependency on the bit rate of the processed video. The amount of computation in variable length decoder is linearly dependent on the amount of the video data in the layer. The resource con-

sumption of the component is

$$R_{VLD} = \sum_{i=0}^{N_L-1} \alpha_4 \cdot R^i + \beta, \tag{5.3}$$

where R^i is bit rate of layer i (0 is L_B , 1 is $L_{E,1}$, etc.). If, however, the variable length decoding is implemented as a hardware module, the bit rate of the video stops playing an important role in the resource consumption of the component. So, for hardware implementation,

$$R_{VLD} = \alpha_5 \cdot N_L. \tag{5.4}$$

Summation of the layers is an additional component that is part of scalable video decoding. The summation is performed on uncompressed video and, therefore, its resource consumption is linear to the number of layers

$$R_{sum} = \alpha_6 \cdot N_L. \tag{5.5}$$

An additional resource consumption is caused by the functionality that facilitates the above-mentioned components. There are some activities regarding headers parsing, frame memory management, motion vectors processing, etc. This activities are content dependent and are responsible for around 5% of the overall decoder resource consumption.

From Equations (5.1)- (5.5), the resource consumption of a decoder is defined by the number of processed layers and, in the case of the variable length decoder implemented as a software module, by the bit rates of the layers. Figure 5.4 shows an example of resource consumption of the decoder where the consumption depends on either on the number of processed layers or on the number of the layers and bit rates of the layers.

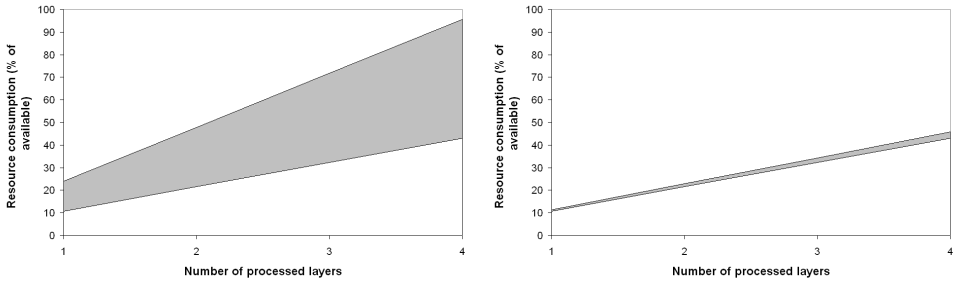

Figure 5.4. An example of resource consumption of a scalable video decoder as a function of number of processed layers: left) with additional dependency on bit rates of the layers (e.g. a software decoder); right) independent of bit rates of the layers (e.g. a hardware decoder).

The discussion above is valid for the majority of off-the-shelf decoders. In the research environment, however, solutions exist that perform computational complexity scalable video processing [69, 70].

5.2 Controlling the decoder

The number of enhancement layers processed in addition to the base layer defines the *decoding quality level*. When it operates at the lowest quality level, the decoder processes only the base layer. If the quality level is increased, the decoder then increases the number of layers to be processed, which results in a rise in the resource consumption. In effect, if quality level i is chosen, this means decoding L_B and $L_{E,j}$ for all j with $1 \leq j \leq i$.

The mapping of the internal quality level to the amount of processed layers is depicted in Table 5.1.

Internal quality level	Number of layers to be processed
q_0	L_B
q_1	$L_B + L_{E,1}$
q_2	$L_B + L_{E,1} + L_{E,2}$
...	...
q_n	$L_B + L_{E,1} + L_{E,2} + L_{E,3} + ... + L_{E,n}$

Table 5.1. The quality levels of the scalable decoder.

The core responsibility of the controller is to change the quality level of the decoder in such way that a frame can be successfully processed within the given budget. As shown in Figure 5.4, decoding at a certain quality level may require a different amount of resources. Therefore, the controller may act in two ways, via a

1. worst-case approach, where the controller chooses the highest quality level with the maximum resource consumption that is lower than the given budget;
2. 'average'-case approach – where the controller chooses a quality level with the average resource consumption that is close to the budget.

The worst-case approach leads to an under-utilization of resources. Figure 5.5 shows an example of a video decoder CPU consumption for processing a three-layer scalable video. The horizontal scale gives the frame number of the video. The vertical scale presents CPU consumption by the decoder. Three graphs in the figure correspond to the processing at three quality levels – the lower curve is quality level 0 , the middle curve is quality level 1 , and the top curve is quality level 2. If the budget given to the video-processing application is 40% of the CPU, the worst-case approach chooses quality level 0. That choice results not only in low quality of the output video, but also in a huge amount of wasted resources – roughly 72% of the assigned budget is unused.

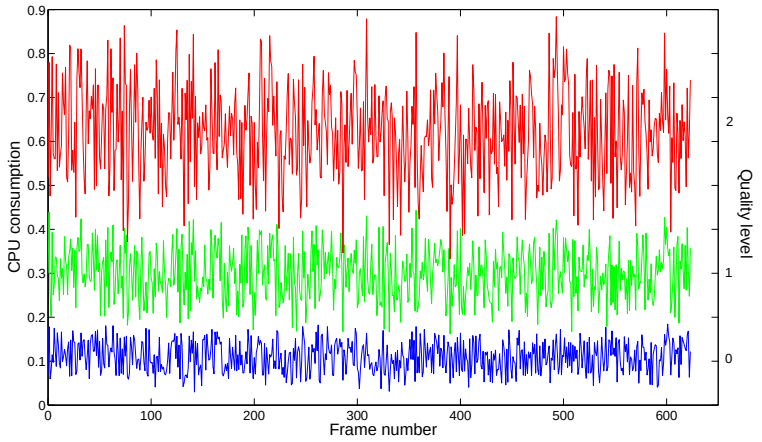

Figure 5.5. CPU resource consumption by the decoder at different quality levels.

In the provided example, the 'average'-case approach chooses quality level 1. The budget utilization in this case is close to 75%. However, in 3.5% of the frames the budget is not sufficient for processing a frame. That situation can be resolved in the following ways:

- Some amount of video frames can be processed in advance, thereby smoothing peak loads in the resource consumption. This method requires that the decoder starts processing the next frame immediately after the current frame processing is finished. Processing in advance is possible when the next frame is available in the input buffer of the decoder and there is free space in the output buffer to put an additional uncompressed frame. If one of these requirements is not met, the decoder is put on hold until the requirements are met again. Since the developed system is targeted at CE devices, the amount of buffering, especially of uncompressed video, is minimal. Thus, the number of frames that can be processed ahead is limited to 2 or 3.
- When the amount of resources is not sufficient for processing a video frame at the chosen quality level, the decoder should process the frame at a lower quality level. The switching between the quality levels is performed at runtime. Moreover, it should be done before the decoder starts the processing of a frame. So, the controller should anticipate 'problematic' frames, lower the quality level for processing these frames and increase the level after the frames are processed. The development of such a controller is difficult to realize: the anticipation of the frame decoding complexity is impossible to predict without preprocessing of the video stream, and preprocessing of that

kind requires the amount of resources that is comparable to the consumption of the decoder itself.

The combination of the two approaches combines the quality switching solution which eases the implementation of the processing-in-advance solution. The basic idea of the combined approach is to process as many frames in advance as possible, saving the budget for a possible high resource-demanding frame. When such a frame comes, the preserved resources allow the decoder to process the frame on time. The major question for the controller is to decide how many resources should stay in the reserve. A bigger reserve allows to handle a huge sudden increase in the resource consumption, whereas a smaller reserve allows processing of a larger number of frames at the higher quality. Moreover, the controller should avoid frequent changes in the quality levels, since that results in changes in the quality of the output video, known as *quality fluctuations*. As shown in [37], frequent changes in the number of layers being processed lead to a considerable drop in the user-perceived quality.

Thus, the strategy of the controller is to

- process video at the highest possible quality level, so to keep the quality of the output video high;
- avoid budget over-usage, to prevent frame losses for display;
- keep the number of quality-level changes low, so the user is not irritated by quality fluctuations.

The development of the controller strategy is presented in the following section.

5.2.1 Controller strategy

Immediately after a frame has been decoded, a decision has to be taken about the quality level at which the next frame will be processed. The set of decisions that can be taken corresponds to the set of quality levels at which the decoder can operate. It is not possible to choose a quality level that requires more layers to be decoded than the number of layers received for a given frame. Thus, the maximum quality level is given by the number of layers received.

Because a decoder receives the layers from a network, there is no guarantee for the number of layers available to the decoder at a given moment in time. This issue is resolved by obtaining information about the next frame from the network reader as shown in Figure 5.6. Every time the network reader puts a frame into the input buffers of the decoder, the information regarding the frame number and the frame affiliation with the particular layer of scalable video is noted. The network reader then communicates the gathered information to the controller in form of $\phi^n = M$, where ϕ^n is frame number n and M is the highest layer received for the frame. For

example, $M = 0$ means that only L_B is received, and $M = 2$ means that L_B , $L_{E,1}$ and $L_{E,2}$ are received.

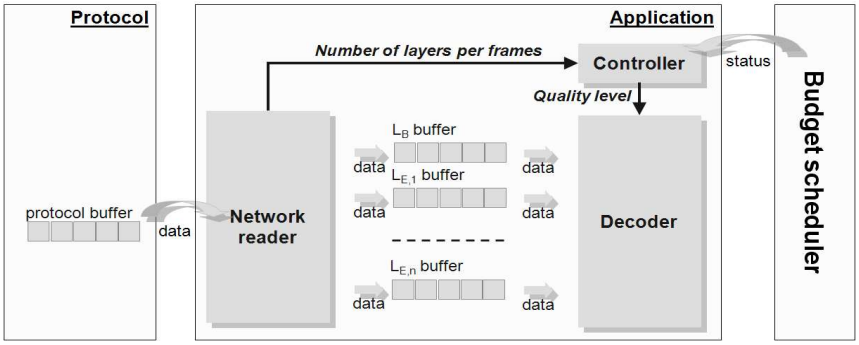

Figure 5.6. Scheme of the developed video-processing application and its collaboration with the rest of the system.

When choosing a quality level, the following three objectives are balanced to maximize the perceived quality. Firstly, the number of deadline misses should be as low as possible. Every frame should be processed within a time interval defined by the video frame rate. For example, with a frame rate of 25 fps, each frame must be processed within 40 ms. Missing a deadline means that the corresponding frame is not shown, which results in a severe drop in the perceived quality of the video in general. Secondly, the number of quality changes should be as low as possible. Finally, the quality level at which the frame is processed should be as high as possible.

One of the approaches to handle a stochastic decision problem is to use a Markov Decision Process (MDP). MDP models return policies that provide a trade-off between immediate and future benefits and costs [71].

MDP mathematical techniques are applied to model decision-making in situations where outcomes are partly random and partly under the control of the decision maker. In the decision-making problem:

- a system evolves through time,
- a decision maker controls it by taking actions at pre-specified points of time,
- actions incur immediate rewards *and* affect the subsequent system state.

An MDP is characterized by a set of states; in each state there are several actions from which the decision maker must choose. For a state s and an action a , a state-transition function $T_a(s)$ determines the transition probabilities to the next state. The decision maker earns a reward for each state visited. The states of an MDP possess the Markov property that transitions to a new state at time $t + 1$

from the current state at time t are independent of all previous states. A Markov Decision Process has *four* elements:

- the state space S ,
- the action space A ,
- the probability that action a in state s at time t will lead to state $s^{'}$ at time $t+1$:

$$P_a(s, s^{'}) = P(s_{t+1} = s^{'} \mid s_t = s, a_t = a), \tag{5.6}$$

- the immediate reward received in state s , $V(s)$.

Finding an optimal strategy is maximizing the reward function

$$\overline{V} = \sum_{t=0}^{\infty} V(s_t). \tag{5.7}$$

Relative progress

The moment at which processing of a frame has been stopped (either finished or aborted) is referred to as a milestone. Every milestone m has a completion time c_m , that is the time at which the processing of the frame is stopped, and a deadline at time d_m , that is the point at which the corresponding frame is needed for an output process (for example, a video renderer that needs to display a frame). The deadlines are strictly periodic. Period T^P is the time between two successive deadlines. In each period, the decoder is guaranteed a certain time budget, b ($0 \leq b \leq T^P$), by the budget scheduler.

At each milestone the controller calculates the relative progress, defined as the fraction of the time budget remaining until the deadline of the milestone. There is an upper bound on the relative progress that defines the maximum number of frames that can be decoded in advance. *Decoding in advance* means that frames are processed by a decoder faster than they are consumed by an output process (for example, a video renderer). The upper bound is determined by buffering and latency limitations as follows: the maximum number of frames that can be processed in advance is defined as $min(BF_i, BF_o, LT^F)$, where BF_i is the maximum number of frames in the input buffer, BF_o is the maximum number of frames in the output buffer, and LT^F the maximum latency that is allowed in the system expressed in frames. So, if the input buffer has a capability to contain *four* frames, the output buffer can store *two* frames and the maximum latency is *three* frames, the maximum number of frames that could be processed in advance is *two*.

Let us denote the deadline of milestone number m as

$$d_m = d_0 + m \cdot T^P, \tag{5.8}$$

where d_0 is an offset.

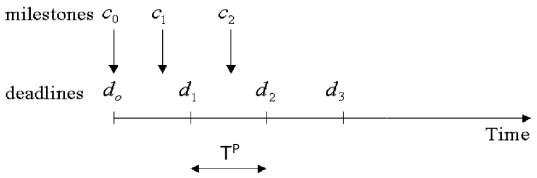

Figure 5.7. Example timeline of the algorithm.

The relative progress ρ_m at milestone m is expressed by the following equation:

$$\rho_m = \frac{d_m - c_m}{b} = m \cdot \frac{T^P}{b} - \frac{c_m - d_0}{b}. \tag{5.9}$$

If the relative progress at the $(m+1)^{th}$ milestone drops below zero, then $\lceil -\rho_m \rceil$ (the next larger integer to $-\rho_m$) deadline misses have occurred since the m^{th} milestone (for example, one deadline is missed if $0 < -\rho_m \leq 1$).

The deadline misses are dealt with in the following way. If at the deadline for milestone m the frame processing is not completed, the decoder is not allowed to continue the processing during the next frame interval, executing an aborting approach. The approach, however, does not necessarily mean that all the information that is decoded so far is lost. As shown in Figure 5.3, the decoder processes the layers of scalable video sequentially. Thus, if the decoding is interrupted after L_B is processed, then the L_B frame is delivered by the decoder for display. In the same way, if the decoding is aborted after $L_{E,1}$ is processed and merged with L_B , the combined picture is shown. In the model, however, we consider an unfinished processing as a deadline miss, despite the fact that some information can be used further. With the aborting approach the minimum relative progress is 0, i.e. the lowest possible value, which in a sense corresponds to using the available output immediately upon the frame deadline.

States and Decisions

Let us define the state of the decoding at milestone m by the relative progress at this milestone, ρ_m. In accordance with Equation (5.9), the relative progress is a real number, so we obtain an infinite number of states, whereas a Markov decision process requires a finite set. We define p as the upper bound on relative progress, while the lower bound for the relative progress is set to 0. The value of p is a measure of the number of periods that the application can work ahead, which is derived from the buffer sizes, as explained above.

The relative progress is split up between 0 and p into a finite set $\Pi =$

$\{\pi_0, ..., \pi_{n-1}\}$ of ($n \geq 1$) progress intervals. [2]

$$\pi_k = \left[\frac{k \cdot p}{n}, \frac{(k+1) \cdot p}{n}\right), \quad for \quad k = 0, ..., n-1. \tag{5.10}$$

Let us denote the lower and the upper bound of a progress interval π_k by $\underline{\pi_k}$ and $\overline{\pi_k}$, respectively.

The maximum quality, $\overline{q_m}$, that can be chosen is given by the number of layers received from the network for the frame that should have been processed at milestone $m + 1$.

A state of the system at milestone m is therefore defined by

- the relative progress interval, denoted by π_k ($0 \leq k \leq n-1$),
- the maximum quality level that is possible to choose for the next unit of work, denoted by $\overline{q_m}$, and
- the quality level previously used, denoted by q_m.

The initial state has relative progress equal to 1 , the quality level previously used is 0 and the maximal quality level that it is possible to choose for the next unit of work is defined by the number of layers of the first frame that are in the buffer.

Transition probabilities

Let p_{ij}^q denote the transition probability for making a transition from a state i at the current milestone m to a state j at the next milestone $m + 1$, when quality level q is chosen to process the next unit of work. After the transition, we have $q(j) = q_{m+1} = q$ and $q \leq \overline{q_m}$, which means that $p_{ij}^q = 0$, if $q \neq q_{m+1}$ or $q > \overline{q_m}$. In other words, the probability of moving to a state with a different quality level from the one that was chosen or moving to a state with quality level higher than the maximum possible level is impossible (zero probability). As an example, if we decide to process video at quality level 1 , it is not possible to end up processing the video at quality level 0 or 2. Furthermore, it is impossible to make a transition to a state where the quality level is 2 if the maximum level allowed in the state is 1.

Let random variable X_q denote the time required to process one unit of work in quality q (the distribution of X_q can be derived experimentally). If we assume that the computation budget per period T^P is given by b , the relative progress ρ_{m+1} can be expressed by the following equation

$$\rho_{m+1} = (\rho_m + 1 - \frac{X_q}{b}) \mid_{[0,p]}, \tag{5.11}$$

[2] The value of n can be seen as the parameter of the model. Obviously, a higher n increases the quality of the decision procedure, and also increases the time needed to resolve the model. It is shown in [72] that $n = 300$ gives a good balance between the quality of strategies and the time needed to calculate them.

where we use the notation

$$f\mid_{[0,p]}=\begin{cases} 0 & if \quad f<0 \\ f & if \quad 0\leq f\leq p \\ p & if \quad f>p \end{cases} \tag{5.12}$$

Let $Y_{\pi_k,\rho_m,q,\overline{q_m},\overline{q_{m+1}}}$ be a stochastic variable, which gives the probability that the relative progress ρ_{m+1} of the decoder at the next milestone $m+1$ will be in progress interval π_k and the maximum quality level that can be chosen in this milestone will be $\overline{q_{m+1}}$, provided that the relative progress at the current milestone is ρ_m , the maximum quality level is $\overline{q_m}$ and quality level q is chosen.

The variable $Y_{\pi_k,\rho_m,q,\overline{q_m},\overline{q_{m+1}}}$ describes the probability of two independent events – the decoder in the next milestone is in the progress interval π_k and the maximum quality level $\overline{q_m}$ is set to $\overline{q_{m+1}}$. The progress and, in turn, the progress interval depend on the performance of the decoder as a result of processing the current frame and all the previous frames. It does not depend in any way on the amount of data that comes after the current frame has been decoded. The number of layers that will arrive for the next frame depends on the sender and the network, but not on the receiver, or even on the decoder. Taking the above into consideration, we can say that

$$Y_{\pi_k,\rho_m,q,\overline{q_m},\overline{q_{m+1}}}=Y_{\overline{q_m},\overline{q_{m+1}}}\cdot Y_{\pi_k,\rho_m,q}, \tag{5.13}$$

where $Y_{\overline{q_m},\overline{q_{m+1}}}$ is the probability that the maximum quality level that can be chosen in milestone $m+1$ is $\overline{q_{m+1}}$ if the maximum quality level that can be chosen in milestone m is $\overline{q_m}$; $Y_{\pi_k,\rho_m,q}$ is the probability that the relative progress of the decoder at milestone $m+1$ is in progress interval π_k provided that the relative progress at the current milestone is ρ_m and quality level q is chosen.

$Y_{\pi_k,\rho_m,q}$ is derived as:

$$Y_{\pi,\rho_m,q}=\begin{cases} P(\rho_{m+1}<\overline{\pi})= & \\ =1-P(\rho_{m+1}\geq\overline{\pi}) & if \quad \pi_k=\pi_0 \\ & \\ P(\rho_{m+1}\geq\underline{\pi}) & if \quad \pi_k=\pi_{n-1} \\ & \\ P(\underline{\pi}\leq\rho_{m+1}<\overline{\pi})= & \\ =P(\rho_{m+1}\geq\underline{\pi})+Pr(\rho_{m+1}<\overline{\pi})= & \\ =P(\rho_{m+1}\geq\underline{\pi})+1-Pr(\rho_{m+1}\geq\overline{\pi}) & otherwise \end{cases} \tag{5.14}$$

In general terms, the probability that the relative progress is in progress interval π_k is the probability that the value of the relative progress is greater or equal than the lower bound of the progress interval and is less than the upper bound. Equation (5.14) describes also two special cases when the progress interval π_k is either the first interval(π_0) or the last interval(π_{n-1}). The lower bound for progress inter-

val π_0 is 0 and with the aborting approach the relative progress is always greater or equal than zero as discussed above. The upper bound of interval π_{n-1} is p and the relative progress is always less than p, since it is defined by the number of frames that can be processed ahead.

Equation (5.14) contains probability functions in the format

$$P(\rho_{m+1} \geq x), \tag{5.15}$$

which can be calculated as follows (taking into account Equation (5.11) and the fact that $0 \leq \rho_{m+1} \leq p$):

$$P(\rho_{m+1} \geq x) = P((\rho_m + 1 - \frac{X_q}{b}) \geq x) = P(X_q \leq b \cdot (1 + \rho_m - x)). \tag{5.16}$$

Let F_q denote the cumulative distribution function of X_q and let us make a pessimistic approximation of ρ_m by choosing the lowest value in the progress interval to which ρ_m belongs. So, if ρ_m belongs to progress interval π_k^m , the pessimistic approximation is of ρ_m is $\underline{\pi_k^m}$. Equation (5.16) can be rewritten as

$$P(X_q \leq b \cdot (1 + \rho_m - x)) = F_q(b \cdot (1 + \underline{\pi_k^m} - x)). \tag{5.17}$$

The probabilities p_{ij}^q can then be approximated by

$$\tilde{p}_{ij}^q = Y_{\overline{q_m},\overline{q_{m+1}}} \cdot \begin{cases} (1 - F_q(b \cdot (1 - \overline{\pi_k} + \underline{\pi_k^m}))) & if \quad \pi_k = \pi_0 \\ F_q(b \cdot (1 - \underline{\pi_k} + \underline{\pi_k^m})) & if \quad \pi_k = \pi_{n-1} \\ (F_q(b \cdot (1 - \underline{\pi_k} + \underline{\pi_k^m})) & otherwise \\ -F_q(b \cdot (1 - \overline{\pi_k} + \underline{\pi_k^m}))) & \end{cases} \tag{5.18}$$

Further information regarding creation of F_q and calculation of $Y_{\overline{q_m},\overline{q_{m+1}}}$ is provided below.

Revenues

Another element of an MDP relates to revenues. Let r_i^q denote the revenue for choosing quality level q in the state i. The revenue is created by the utility function, the deadline-miss penalty function and the quality-change function, as described in [72].

The utility function $u(q)$ returns a positive value that is related to the perceived quality of the output of the algorithm running at internal quality level q.

The deadline-miss penalty function returns a positive value that is related to the number of deadlines we expect to miss, if the quality level q is chosen in the current state. This value should be subtracted from the revenue.

Finally, the quality-change function $c(q(i), q)$ returns a penalty for switching the quality level from $q(i)$ to q. The value of the quality-change function should be subtracted from the revenue.

Solving MDP

The solution of an MDP is given by a decision strategy that maximizes the average revenue per transition. We used the *successive approximation* solution technique [71] to solve the MDP. The strategy calculation is done offline.

For the full definition of MDP, the cumulative distribution function F_q of variable X_q , which represents processing time (or resource consumption) for a frame at a particular quality level, has to be estimated. The estimation can be done by using statistics for frames that were processed by the decoder. The statistics are gathered offline by feeding the decoder with various video inputs. The input data for the model is given in the format $\Lambda_i = C_i$, where Λ_i is a number of processed layers for frame number i , C_i is processing time for frame number i.

The calculation of the $Y_{\overline{q_m},\overline{q_{m+1}}}$ can be done in two following ways.

- The *network-unaware* solution assumes that the probability $Y_{\overline{q_m},\overline{q_{m+1}}}$ has the same value for any pair $\overline{q_m}$ and $\overline{q_{m+1}}$ (i.e. is uniformly distributed):

$$Y_{\overline{q_m},\overline{q_{m+1}}} = \frac{1}{N_L}, \tag{5.19}$$

  where N_L is the maximum number of layers. Real changes in network conditions are not taken into account, so the chosen strategy is the same for good and bad states of the media. This approach is useful when no estimation of losses of video data during transmission is available.

- The *network-aware* solution creates controller strategy for individual pairs $(\overline{q_m}, \overline{q_{m+1}})$. Consequently, the strategy can be selected according to the observed network conditions. As discussed in Chapter 2, providing the network conditions (i.e. probabilities of the successful transmission of the layers) is a responsibility of the sender.

5.3 Conclusions

In this chapter we have presented a resource management solution for the receiver of a scalable video. The solution is based on the use of a controller that changes the number of layers processed by the decoder and controls in this way the resource consumption of the decoder. The controller bases its decisions on a precalculated strategy. A set of strategies is created offline by means of MDP.

Every strategy is calculated for a particular configuration of layers and a particular probabilities of successful transmission of the layers. If the layer configuration changes or probabilities change, the previously used strategy is substituted with

the most appropriate one. The following aspects contribute to the total number of strategies that are needed.

- At least one strategy per layer configuration is calculated. The number of configurations, N_C, is, as we showed in Chapter 4, a function of the number of layers, the maximum bit rate for the sum of the layers, the minimal bit rate for L_B, and the step for assigning bit rates to the layers.
- For every layer configuration the loss estimator calculates the probability of successful transmission of a given number of layers, E_K. The probability depends on the network conditions. So, for every network condition, the loss estimator calculates a set of probabilities, $\Upsilon = \{E_0,...,E_{N_L}\}$ (see Equation (4.11)). The number of controller strategies, in this case, equals N_C multiplied by the number of probability sets, N_Υ. Since the probability is a rational number between 0 and 1 , there can be an infinite number o probabilities. Needing probability set per probability leads to an infinite number of probability sets. In practice, the range of probability values is split up into a finite set of N_Π equal intervals. Thus, N_Υ is a function of N_Π and N_L.

Table 5.2 gives an example of the total number of strategies as a function of parameters described above. As shown in Table 5.2 a large number of strategies is calculated in advance and stored on the terminal. The consequences are high memory consumption for storing the strategies on a terminal and high computational power for the creation of strategies.

$Step\_Rate$ [Mbps]	N_Π	N_C	N_Υ	Total strategies
0.1	20	$35,990$	$1,330$	$47,866,700$
0.1	10	$35,990$	165	$5,938,350$
0.25	20	$2,300$	$1,330$	$3,059,000$
0.25	10	$2,300$	165	$379,500$
0.5	20	286	$1,330$	$380,380$
0.5	10	286	165	$47,190$

Table 5.2. Number of strategies as a function of $Step\_Rate$ and the number of probability intervals. The set of layer configurations is built assuming three-layered scalable video with minimal BL bit rate of 1 Mbps and maximal total bit rate of 6 Mbps.

In addition to the *network-aware* solution, we present a solution that does not take the loss probability into account directly. This *network-unaware* solution assumes that the probability of receiving any number of layers for a frame is equal. In general terms, the solution is aware that the number of input layers is constantly changing, but does not know the particularities of the changes.

In Chapter 6 we present the comparison of the developed solution to the *best-*

effort approach in the decoding. Moreover, the results of the comparison of the *network-unaware* with the *network-aware* solution are presented.

6

Evaluation of new system components

This chapter presents the results of system components evaluation. Since there are large dependencies between the experiments, we have integrated all experiments in a single chapter. The chapter is organized as follows. First, we evaluate the scalable coding technique, looking at the difference in the quality of video produced by the various approaches for the encoding and at the overhead induced by the approaches. Second, we evaluate our streaming techniques, focusing on the behavior of the system under various network conditions. Third, we evaluate the technique developed for terminal resource management. We look at the difference in quality resulting from the controller in comparison with a non-controlled *best-effort* solution. We also look at how the network awareness of the controller affects the quality of video processing.

6.1 Scalable video coding

This section presents evaluation of the scalable encoding technique. The evaluations include: comparison of the specific SNR encoding to the encoding based on a cascade of non-scalable encoders, comparison of the I-frame mode to B-frame mode, estimation of the overhead caused by the usage of scalable video, importance of the layer configurations, and a comparison of an open-loop transcoding to a re-encoding technique.

6.1.1 Encoder solutions and encoding modes

In Chapter 3 two approaches for making scalable video are described – a specific SNR encoder and a cascade of non-scalable encoders. In this section we investigate what encoding approach in combination with encoding mode produces the best results in terms of quality of encoding (e.g. achieves the highest picture quality for the same bit rate).

To start, we investigate the quality improvement delivered to the BL by the EL for a different GOP size. With a GOP structure IB_n , we vary n in the GOP (i.e. by changing the number of *B* frames) in the test video. When $n = 0$ the stream contains only I frames, so encoding operates in I-frame mode. The GOP size for this approach is calculated as $S_{GOP} = n + 1$. We use two DVD video sequences, each re-encoded into two-layer scalable video sets with a fixed BL bit rate and a different EL bit rate. The bit rate of the BL is 2.5 Mbps. The EL bit rates are either 1.5 Mbps or 2.5 Mbps.

Figures 6.1 and 6.2 present an *average* quality addition delivered by $L_{E,1}$ with GOP structure IB_n for different values of n. The quality delivered by a L_B is the same for both the SNR-specific and the cascade-based approach [1].

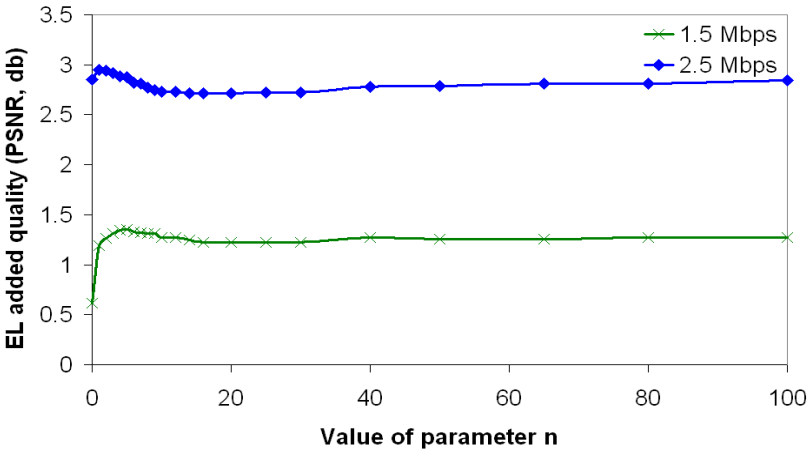

Figure 6.1. Quality added by $L_{E,1}$ in two-layer scalable coding based on a cascade of non-scalable encoders. The added quality is calculated as the difference between PSNR of the video that contains the sum of L_B and $L_{E,1}$ and PSNR of the video that consists of L_B only.

The horizontal axis in the figures represents the value of parameter n for the GOP structure in the EL. When n is equal to 0, the GOP consists of a single I

[1] Both solutions use the same software implementation for encoding BL.

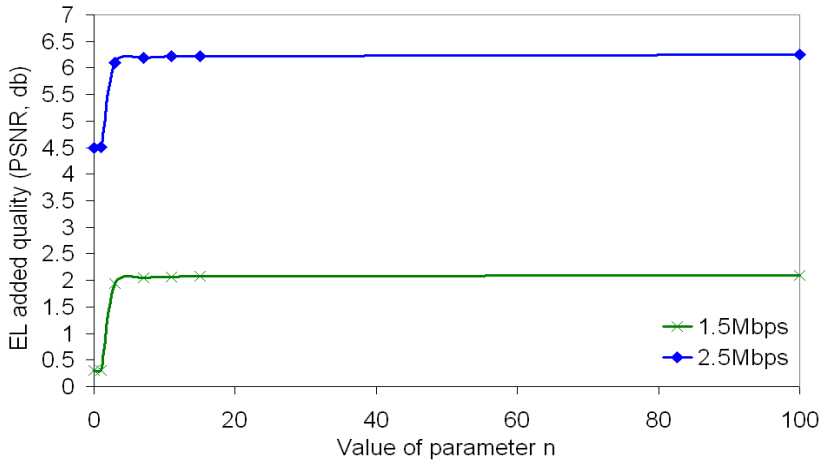

Figure 6.2. Quality added by $L_{E,1}$ in two-layer scalable coding based on an SNR-specific encoder. The added quality is calculated as the difference between PSNR of the video that contains the sum of L_B and $L_{E,1}$ and PSNR of the video that consists of L_B only.

frame, while $n = 1$ describes a GOP that consists of one I frame and one B frame (so the whole video sequence looks like *IBIBIBIBIBIB*...). The vertical axis gives the quality addition in dB. The same values are calculated for two different bit rates (1.5 Mbps, 2.5 Mbps), both shown in the figures.

The results in Figure 6.1 show that with a GOP size exceeding 8, the added quality stabilizes for the ELs made with the cascaded approach. This is the result of a lack of temporal correlation between $L_{E,1}$ frames, which is logical since EL contains residuals of the BL encoding. If the motion estimation in the encoder does not find any similarity between the current frame and a reference frame, macroblocks of the current frame are encoded in I mode. As a result, for the bigger value of n , more B frames in an EL are encoded with macroblocks of the I type. The quality shown by the IB_n GOP with large n is still higher than for a GOP that uses only I frames. The explanation for this effect is the ability of an encoder to skip empty macroblocks [2] in B frames.

On the contrary, the SNR-specific encoder produces better results with increasing GOP size in $L_{E,1}$ (see Figure 6.2). The reason for this is the absence of real motion estimation in the creation of B frames. As discussed in Chapter 3, the reference I frames are empty in the EL, so we can force prediction in every block

[2] An empty macroblock is a macroblock that has only *zero* values after quantization.

of a B frame. Therefore, the values stored in B frames are the same, independent of the distance to the reference frame. The added quality delivered by $L_{E,1}$, therefore, depends only on the amount of I frames that are present in the stream. Since I frames bear no information, an EL with less I frames produces better results.

The SNR-specific encoder delivers a video of the lowest quality with $n = 0$, which corresponds to operating in I-frame mode. The same results are observed with low-bit-rate ELs (1.5 Mbps) in cascaded approach. The quality delivered by the cascaded approach with larger ELs in I-frame mode is comparable with the quality in B-frame mode.

In general, the SNR-specific encoder performs much better. However, in an environment where internal modification of an encoder is not possible, the cascade-based approach could be a valuable option.

With respect to the GOP structure, an EL that consists of a single GOP with I frame and all other frames of type B produces the best quality improvements.

6.1.2 Overhead of the coding and layer configuration

We investigate what bit-rate overhead the scalable video has with respect to a non-scalable solution. We also look at how the bit-rate distribution over layers influences the overall quality of the scalable video, focusing on the importance of BL size. This section is meant to answer two questions:

- if a video encoded in a single layer with a bit rate R_B delivers a quality Q , what bit rate R_S is required to encode the same video into a BL and ELs to deliver the same quality;
- how does the distribution of the total bit rate R_S of a scalable video among BL and ELs influence the quality delivered by the video?

Objective tests

For the rest of our evaluation, we use only the SNR-specific encoder with IB_n GOP structure, where n is equal to the number of frames in the video sequence minus one. As discussed in Chapter 3, the first frame (I frame) of the sequence contains zero values, and all B frames use I frame as prediction.

In this test we study the bit-rate overhead created by encoding a video into a scalable video, instead of a single-layer video. For the test, we use the Foreman (CIF resolution, 300 frames) test sequence. The video is encoded into three different one-layer streams with bit rates of 1 , 2 and 3 Mbps (this provides us with reference videos that are distinguishable in quality). The same video is also encoded into a BL and one or two ELs, using the proposed scalable coding techniques. In the latter case, the bit rates of the first and second the EL are set equal ($R_{E,1} = R_{E,2}$). The bit rate of the base layer is chosen as a percentage of the

reference bit-rate. For example, if the reference video is 1 Mbps , the BL that is 25% of the reference video is a layer with bit rate 250 Kbps.

Afterwards, we choose the bit rate of the EL such that the overall objective video quality of the scalable video is equal to the quality delivered by a reference video. In the example above, it may require an EL of 2 Mbps to deliver, together with the BL of 250 Kbps, the same video quality as a single-layered 1 Mbps video.

The sum of bit rates of all layers of the test video and bit rate of the reference video is then compared. The results of the evaluation are presented in Figures 6.3 and 6.4. The horizontal axis shows the bit rate of the BL layer in relation to the bit rate of the reference video. The vertical axis gives the overhead as a percentage. An overhead of 100 percent means that twice as many bits are needed to encode scalable video of the same quality as the reference. For example, the top curve on the left graph in Figure 6.3 demonstrates that if the reference video is encoded into a single layer of 1 Mbps , it delivers the same quality as a scalable video consisting of L_B with bit rate equal to 25% of the bit rate of the reference video (1 Mbps) and $L_{E,1}$ with such a bit rate that the total bit rate of L_B and $L_{E,1}$ is 160% higher than the bit rate of the reference.

In general, the enhancement layer created in I-frame mode shows a higher overhead in the number of transported bits than the B-frame mode. Evaluation of both methods shows that the overhead depends on

- how big the fraction of L_B bit rate is in the overall bit rate of the scalable video,
- on the overall bit rate of the video, and
- on the number of enhancement layers.

Scalable video with a larger BL introduces less overhead than scalable video with a small BL. As shown in Figure 6.4, separation of the video into three layers with a BL of 25 percent of the reference bit rate produces twice as much overhead as a configuration with a BL of 50 percent of the reference bit rate. The main reason for this is that a low-bit-rate BL is of poor quality, which means that a substantial amount of additional information is required to improve it. Scalable video with a higher overall bit rate (which implies higher quality) produces less overhead than videos with lower bit rates. The syntax overhead per enhancement layer is, in the first approximation, equal. Thus, for higher bit rates the effect of the syntax overhead (the amount of bits used to describe the stream) is less important, because the fraction decreases with increasing bit rate. Syntax overhead is also the main reason for the overhead increase when an extra enhancement layer is introduced.

To further investigate how the distribution of the bit rates among BL and EL influences the overall quality, we took a real-life example with a DVD video se-

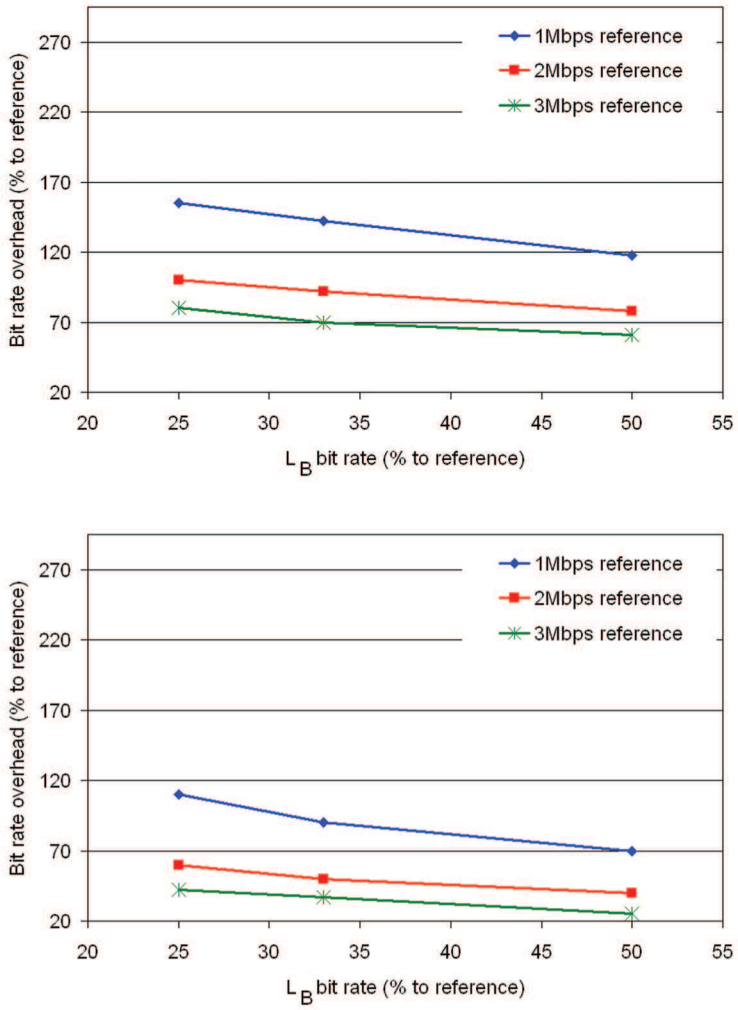

Figure 6.3. Dependency of the overhead of a scalable video (two layers) from a bit-rate distribution among L_B and $L_{E,1}$ and from the overall bit rate. Top – I-frame mode, bottom – B-frame mode.

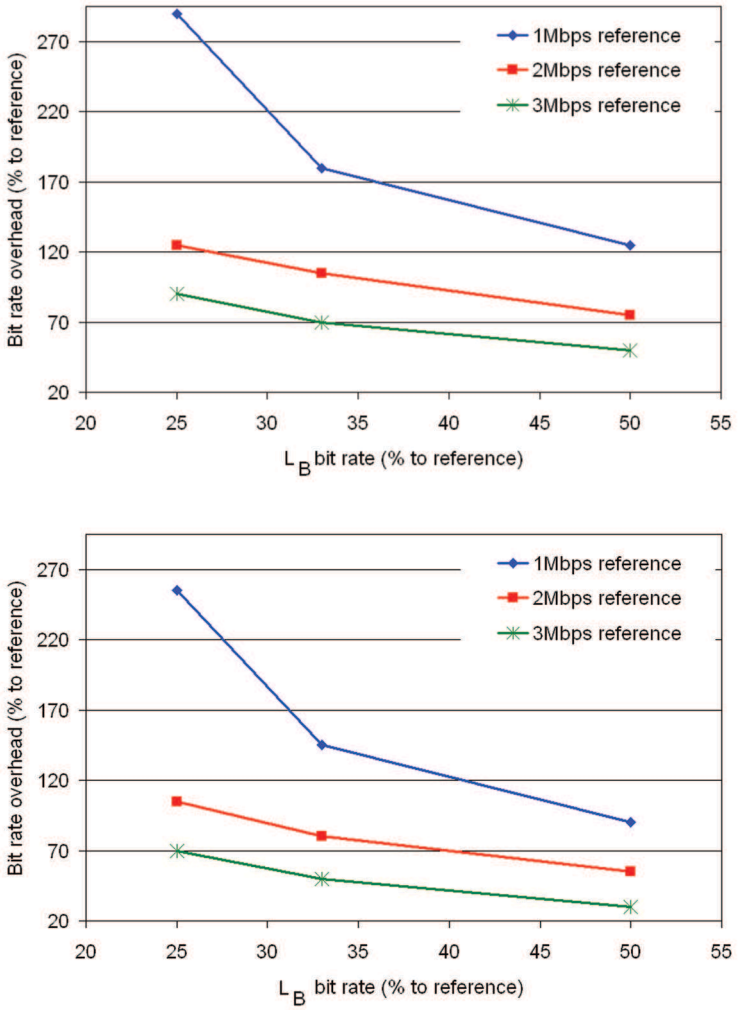

Figure 6.4. Dependency of the overhead of a scalable video (three layers) from a bit-rate distribution among L_B and $L_{E,1}$ and from the overall bit rate. Top – I-frame mode, bottom – B-frame mode.

quence (Standard Definition, 1000 frames). We encoded the video sequence into a set of test scalable videos consisting of a L_B and $L_{E,1}$. Although bit rates of L_B and $L_{E,1}$ are different for every video in the test set, the bit rate of the sum of the layers was constant at a value of 5 Mbps. For every test video, we calculated the quality delivered by only L_B and the quality delivered by both layers. The resulting values were compared to the quality of the reference, that is as single layer with bit rate of 5 Mbps. The differences in quality delivered by the reference video and by L_B and L_B with $L_{E,1}$ are shown in Figures 6.5 and 6.6.

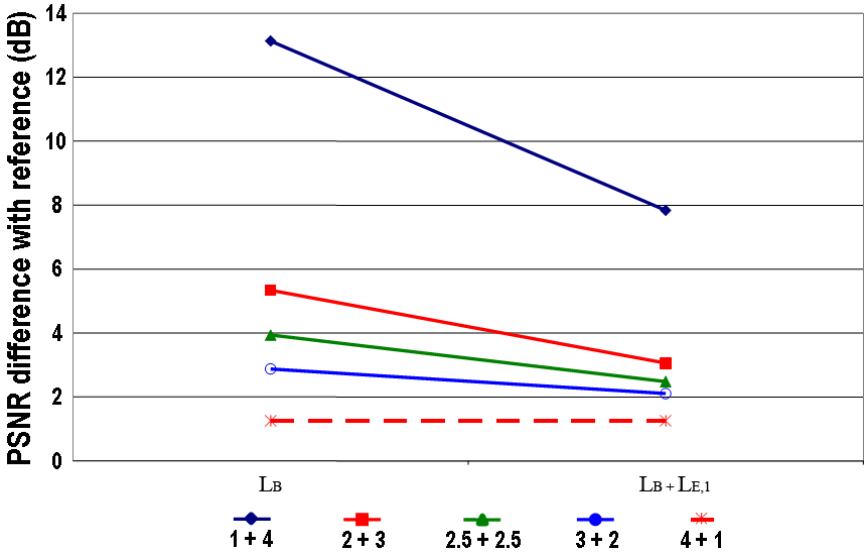

Figure 6.5. Difference in PSNR between one-layer reference and two-layer I-frame scalable coding (the overall bit-rate is 5 Mbps). The (x+y) notation at the bottom represents a video with L_B of x Mbps and $L_{E,1}$ with y Mbps.

The scalable video coding based on B frames is much better in terms of the quality delivered. The importance of the base layer size is less significant for the B-frame mode than for the I-frame mode. This is mainly due to much better compression in the B-frame enhancement layer.

From the results above, it is clear that the quality of the video delivered depends on the allocation of bit rates to the base and enhancement layers. From all SNR-scalable videos with an equal number of layers and total bit rate, the ones with the highest bit rate for BL provide the best video quality.

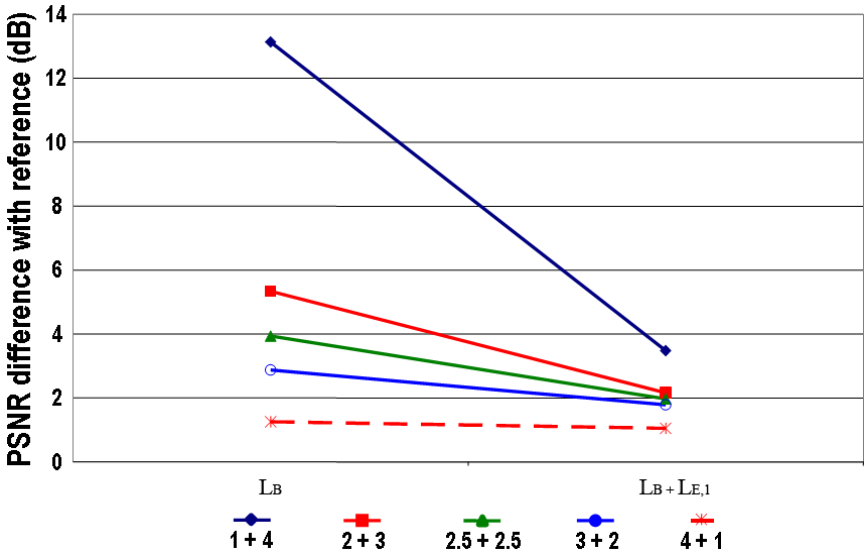

Figure 6.6. Difference in PSNR between one-layer reference and two-layer B-frame scalable coding (the overall bit-rate is 5 Mbps). The (x+y) notation at the bottom represents a video with L_B of x Mbps and $L_{E,1}$ with y Mbps.

Subjective tests

Goal

Choosing the appropriate configuration of layers is one of the main challenges in the creation of a scalable video. As shown in the previous section, out of SNR scalable videos that have the same number of layers and the same total bit rate, it is the video with the larger BL that delivers a higher video quality. The results described in the previous section are based on objective quality measurements (PSNR). This section is meant to support the obtained results with subjective assessments of the quality of the scalable video. We performed user tests to validate these results with subjective evaluations.

Method for performing experiments

The effect on people's judgment of the quality of different configurations of base layer and enhancement layers in combination with different bit rates was measured. A movie fragment from a DVD with an average bit rate of 6 Mbps was transcoded to make the various video sequences for the subjective evaluations. The following independent variables were used in the experiment:

1. Number of layers – the base layer (L_B), the base layer plus one enhancement layer (L_B+$L_{E,1}$) and the base layer plus two enhancement layers

(L_B+$L_{E,1}$+$L_{E,2}$).

2. Type of coding – four different types of coding, numbered $0 - 3$. A coding type describes the distribution of bit rate over different layer of a scalable video, e.g. whether a BL is bigger/smaller than the rest of the layer. A detailed description of the coding types is given below.
3. Bit rate of the video data – 2, 3, 3.5, 4, 5 and 6 Mbps.

Coding types:

- Coding type 0 consists of one layer based on a bit rate of approximately 3 Mbps (the video sequences were made by eliminating B-type frames from the original DVD fragment of 6 Mbps). This coding type was originally included into the tests to get a rough idea about quality of the IFD technique that is discussed in Chapter 4. Since the coding is not related to the concept of scalable video, it is not further discussed in the current chapter.
- Coding type 1 consists of a base layer of 1.5 Mbps ($R_B = 1.5$ Mbps), an enhancement layer of 2 Mbps ($R_{E,1} = 2$ Mbps), and an enhancement layer of 2.5 Mbps ($R_{E,2} = 2.5$ Mbps). For this coding type it holds that each next layer is larger than the previous layer, i.e. the bit rate increases for each layer added.
- Coding type 2 consists of a base layer at 3 Mbps ($R_B = 3$ Mbps), an enhancement layer of 2 Mbps ($R_{E,1} = 2$ Mbps) and an enhancement layer of 1 Mbps ($R_{E,2} = 1$ Mbps). For this coding type it holds that each next layer is smaller than the previous layer, i.e. the bit rate decreases for each added layer.
- Coding type 3 consists of a base layer at 2 Mbps ($R_B = 2$ Mbps), an enhancement layer of 2 Mbps ($R_{E,1} = 2$ Mbps) and an enhancement layer of 2 Mbps ($R_{E,2} = 2$ Mbps). For this coding type it holds that each next layer has the same size, i.e. the bit rate is equal for each layer.

The following comparisons were made with regard to overall bit rate:

- bit rates: 3 and 2 if only the L_B is received. This was the one-layer case in Session 1.
- bit rates: 3.5, 5 and 4 if the L_B and $L_{E,1}$ are received. This was the 2-layer case in Session 2.
- bit rates: 6, 6 and 6 if the L_B , $L_{E,1}$ and $L_{E,2}$ are received. This was the 3-layer case in Session 3.

Four different video sequences were made from these variables. These sequences were presented in random order to participants who were asked to compare them in terms of best and worst video quality. Four different sessions with

different combinations were presented. The structure and the nesting of the variables for the sessions are shown in Table 6.1.

Session number	1			2			3			4		
Num. of layers	1 (L_B only)			2 (L_B and $L_{E,1}$)			3 (L_B ,$L_{E,1}$,$L_{E,2}$)			mixed		
										1	2	3
Coding type	3	2	0	1	2	3	1	2	3	2	1	3
Bit rate (Mbps)	2	3	3	3.5	5	4	6	6	6	3	6	4

Table 6.1. Overview of configurations used in the experimental session.

In each session, a sequence of 3 different configurations was presented. A new sequence was made for Session 4 in which one configuration from Session 1, Session 2 and Session 3 was used.

A session consists of 6 repetitions of the same test run. A run consists of a sequence, which starts with a reference video clip followed by three video clips. These three video clips consist of 3 different configurations of coding schemes. The reference video clip constitutes the original DVD video with a bit rate of 6 Mbps. All video clips have the same length and are derived from the same video content. A pause in which a gray screen is presented for 5 seconds is inserted between each test run sequence. After the third video clip there is a pause of 15 seconds. During that pause subjects are asked to choose the best and the worst video clip in terms of image quality by rating them with a score of -1 (worst), 0 and 1 (best). The test runs are all the same. The order of presentation is changed continuously during a session.

The sessions can be summarized as follows:

- Session 1: L_B only (2 Mbps and 3 Mbps sequences of coding type 3 and 2, respectively; and 3 Mbps of coding type 0).
- Session 2: L_B and $L_{E,1}$ (coding types 1, 2, 3).
- Session 3: L_B , $L_{E,1}$ and $L_{E,2}$ (coding types 1, 2, 3).
- Session 4: Sequence with L_B only (coding type 2), L_B and $L_{E,1}$ (coding type 1), L_B plus $L_{E,1}$ and $L_{E,2}$ (coding type 3).

All sessions include the reference video sequence.

16 persons (8 male, 8 female) participated in the study. Their ages ranged from 23 to 30 years, with the average age being 26 years. The participants had no prior experience with video quality assessments or any professional experience with systems for video processing. The participants worked in parallel (in a single group) watching video sequences on a large plasma TV screen. The participants did not communicate during the test. The test leader explained the experiment and

its goals, provided the subjects with a form to fill in their scores and demonstrated the system.

Results

Session 1. Table 6.2 shows the results as mean scores for the different L_B bit rates in Session 1. The results of the session showed that people see very clearly the differences between the sequences.

Condition	Mean score	Std.Dev.
$R_B = 2$ Mbps	0.23	0.43
$R_B = 3$ Mbps	0.77	0.43

Table 6.2. Effect of L_B size.

According to the participants, the base layer with a bit rate of 3 Mbps earns the best marks delivering a higher quality than a base layer of 2 Mbps.

Session 2. Table 6.3 shows the results as mean scores for the different sequences in Session 2. The video that consists of a base layer at 3 Mbps and an enhancement layer of 2 Mbps has the best quality according to the participants. The base layer of 1.5 Mbps and an enhancement layer of 2 Mbps (coding type 1) has the worst video quality according to the participants.

Condition	Mean score	Std.Dev.
$R_B = 1.5$ Mbps and $R_{E,1} = 2$ Mbps	-0.83	0.51
$R_B = 2$ Mbps and $R_{E,1} = 2$ Mbps	0.16	0.38
$R_B = 3$ Mbps and $R_{E,1} = 2$ Mbps	0.67	0.69

Table 6.3. Effect of bit-rate distribution between two layers (L_B + $L_{E,1}$).

Session 3. Table 6.4 shows the results as mean scores for the different sequences in Session 3.

Condition	Mean score	Std.Dev.
$R_B = 1.5$ Mbps , $R_{E,1} = 2.5$ Mbps , and $R_{E,2} = 3$ Mbps	-0.40	0.81
$R_B = 3$ Mbps , $R_{E,1} = 2$ Mbps , and $R_{E,2} = 1$ Mbps	0.57	0.68
$R_B = 2$ Mbps , $R_{E,1} = 2$ Mbps , and $R_{E,2} = 2$ Mbps	-0.17	0.65

Table 6.4. Effect of bit-rate distribution between three layers (L_B + $L_{E,1}$+$L_{E,2}$).

A base layer of 3 Mbps, an enhancement layer of 2 Mbps ($R_{E,1} = 2$ Mbps) and a second enhancement layer of 1 Mbps ($R_{E,2} = 1$ Mbps) produces the best quality according to the participants. A base layer of 1.5 Mbps, an enhancement layer of 2 Mbps and a second enhancement layer of 2.5 Mbps produces the worst video quality according to the participants. All the sequences in this session have

the same overall bit rate. The only difference is the combinatorial scheme. It can be concluded that coding type 2 is perceived to be better than coding types 1 and 3.

The subjects did not perceive a difference between sequences of coding type 1 and 3. The BLs for coding types 1 and 3 have almost the same bit rate, i.e. 1.5 and 2 Mbps respectively. The test sequences have a total bit rate of 6 Mbps, which implies that the sum of the bit rates of the enhancement layers is 4.5 and 4 Mbps respectively. As a result, we have a case in which base layers of nearly the same quality are improved almost to an equal extent by enhancement layers.

Session 4. Table 6.5 shows the results as mean scores for the sequences in Session 4.

Condition	Mean score	Std.Dev.
$R_B = 3$ Mbps	0.53	0.73
$R_B = 1.5$ Mbps, $R_{E,1} = 2.5$ Mbps , and $R_{E,2} = 3$ Mbps	-0.60	0.62
$R_B = 2$ Mbps and $R_{E,1} = 2$ Mbps	0.07	0.69

Table 6.5. Effect of number of layers for different schemes.

A base layer of 3 Mbps has the best quality according to the participants. A base layer of 1.5 Mbps, enhancement layer $L_{E,1}$ of 2 Mbps and an enhancement layer $L_{E,2}$ of 2.5 Mbps has the worst video quality according to the participants. A base layer of 2 Mbps and an enhancement layer of 2 Mbps gives an average video quality. The results support the theory on the importance of the BL quality. It seems that a single BL of 3 Mbps provides a higher quality than L_B of 2 Mbps plus an $L_{E,1}$ of 2 Mbps. The latter, in turn, is perceived to be better than the result of processing all three layers, L_B of 1.5 Mbps, $L_{E,1}$ of 2 Mbps and $L_{E,2}$ of 2.5 Mbps.

Conclusions

When comparing the use of different layers, we find that it is better to apply one layer with a large BL than to apply two or three layers with a small BL. In other words, the quality of a small base layer cannot be improved by adding enhancement layers. The best approach for coding SNR scalable video is to create the largest possible BL within the current environment settings and to improve it with smaller ELs. These results support the previous findings in objective tests that the size of the base layer is more important than the size of the enhancement layers.

Another outcome of the user test is that a single BL layer is always better than a set of BL and ELs if the overall bit rate is the same. The system should therefore minimize the number of layers in the scalable video coding whenever possible.

6.1.3 Open-loop transcoding vs. re-encoding

In this section we present a basic comparison of the quality of the open-loop transcoding technique for scalable video to the re-encoding technique.

We use a movie fragment with an average bit rate of 10 Mbps. In the first test the fragment was transcoded into a set of single layer video streams, where the bit rate of the output video was in the range of 1 Mbps to 8 Mbps. Figure 6.7 shows the quality results produced by the open-loop transcoding and the re-encoding approach. The quality of the output video is calculated as PSNR characteristics in comparison to the quality delivered by the original video. The horizontal axis represents the bit rate of the BL for both approaches. The vertical axis gives the PSNR value of the output video.

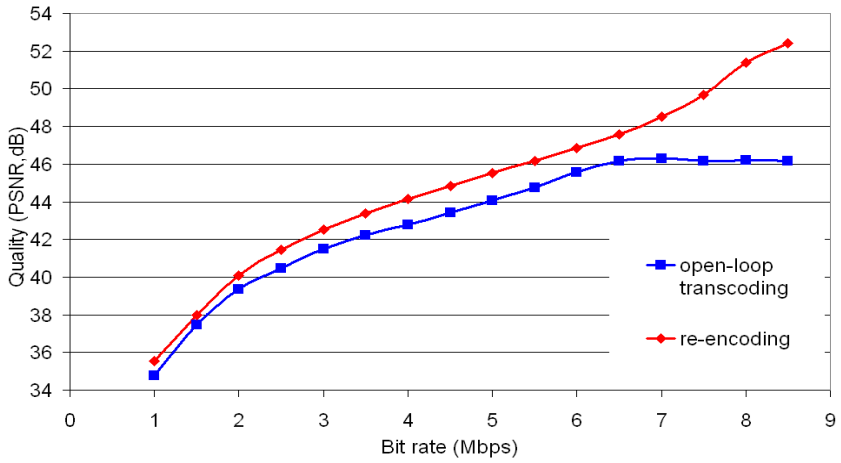

Figure 6.7. Quality delivered by L_B created with different transcoding techniques.

It is clear from Figure 6.7 that the re-encoding approach outperforms the open-loop transcoding in terms of quality. A substantial quality difference can be observed with bit rates over 6.5 Mbps. At the high bit rates, the errors that are introduced by re-quantization in the open-loop transcoder are small. However, he propagation of the errors via motion compensation creates a significant quality drop as can be observed in Figure 6.8. The figure shows the PSNR values for every frame of the BLs. The BLs have bit rate 7 Mbps and the GOP contains 12 frames. It can be seen in the figure that, despite the high bit rate of the video, the open-loop transcoding still shows a significant quality decrease in predicted frames (frames with a frame number that is *not* a multiple of 12).

For the second test, we compare the quality delivered by two-layers scalable

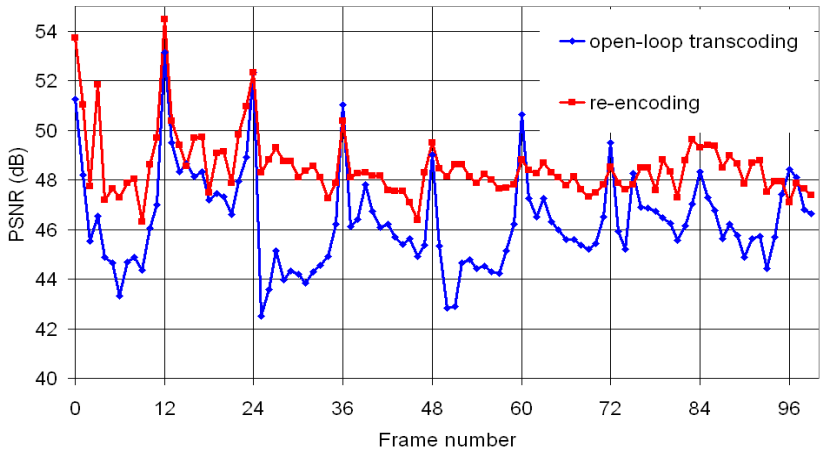

Figure 6.8. PSNR measurements of 100 frames of L_B with a bit rate of 7 Mbps.

video that are produced by the investigated solutions. The bit rate of the L_B is chosen differently for re-encoding (2.5 Mbps) and for open-loop transcoding (3 Mbps). The intention is to produce BLs of the same quality. The video is encoded with varying $L_{E,1}$ bit rate. The measurements of an overall quality delivered by the layers is presented in Figure 6.9.

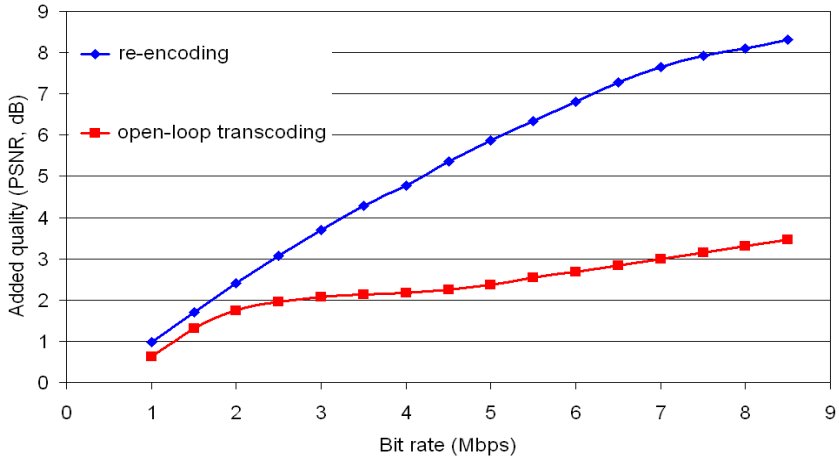

Figure 6.9. Additional quality delivered by $L_{E,1}$ of two-layers scalable video created with different transcoding techniques. The L_B delivers the same quality.

The results again show that the re-encoding approach outperforms the open-

loop transcoding in terms of quality, as is predicted in Chapter 3. At low bit rates, both solutions show similar results. With increase of the bit rate, the re-encoding solution progresses much better than the open-loop transcoding. The latter shows only small improvements not being able to overcome the shortcomings of the open-loop scheme.

6.2 Streaming solution

6.2.1 Experimental setup

The network environment influences the behavior of the streaming solution by influencing the performance of the transmission protocol, since the observations of the protocol performance are directly used to build the streaming strategy at runtime. The performance of the transmission protocol is characterized by the speed of the transmission, which is influenced by both the throughput of the network links and packet losses on the links. The losses can be categorized in two types: protocol-level losses and MAC-level losses (in wireless networks).

When a packet is lost over the wireless link, it is retransmitted by the MAC layer as many times as is allowed by the network interface settings. The information about such losses does not reach a transmission protocol, so for the protocol it results in a network throughput decrease.

The losses that are visible at the protocol level are more critical since the protocol that is used in the proposed streaming solution is TCP. The TCP protocol is developed for lossless wired links, so any packet loss is considered by TCP to be caused by congestion. TCP is developed in such a way that if congestion is detected, the protocol slows down drastically and then slowly recovers its speed. In our system protocol level losses may occur due to congestion caused by competing traffic, or due to Access Point / Route buffer overflow.

Various factors that influence the protocol performance can be modelled by additional packet error rates at either the MAC or TCP level. For example, the absence of competing traffic is modelled by zero TCP packet error-rate, whereas the presence of the traffic from an additional node is represented by the positive value of the TCP packet error-rate. The presence of an additional wireless device in the network with wireless sender/receiver is modelled by elevated packet error rates on MAC level, since transmissions on the wireless link may interfere.

So, independent of the network configuration, the conditions, which are relevant for the transmission protocol performance, can be modelled by a simple single-link communication system (see Figure 6.10) via setting packet error rates.

To investigate the behavior of the created streaming solution we use a simulation that is based on the NS2 network simulator [73]. NS2 is a discrete-event simulator targeted at networking research. It provides support for simulation of

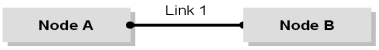

Figure 6.10. A single-link network.

TCP, routing and multicast protocols over wired and wireless networks. The simulation experiments involve the transmission of a video that contains 2500 frames through a network that consists of a sender and a receiver. The link has a certain packet error rate, TCP PER, with packet losses occurring at the protocol level. The link might experience some losses in the MAC layer. The amount of losses is given by the packet error rate within the MAC layer, MAC PER. The values of TCP PER and MAC PER are given in Table 6.6 and are used by NS2 at the start of the simulation session.

During a session, the network simulator simulates transmission of a set of data packets over the pre-described network and calculates various statistics regarding the transmission. NS2 models the transmission starting from the moment data is put into the protocol buffer at the sender side and ending when the data is received and stored in the protocol buffer at the receiver side. The simulator uses a separate software module to generate the input data description. This software module indicates when and how many data arrives to the protocol buffer. The module implements all the algorithms of our streaming solution described in Chapter 4. It takes as an input TCP buffer fullness (as is measured by NS2), estimates bandwidth, chooses layers configuration and updates the "transcoder". The latter is a simple function that generates a value of the current frame sizes, based on the layer configuration. The functioning of the simulated streaming solution depends on the number of internal parameters that are discussed in details below. The settings of the parameters, which are set at the start of the simulation, are given in Table 6.6.

EL data lifetime (frames)	Observation window (ms)	Bandwidth addition (Kbps)	Step rate (Kbps)	TCP packet error rate	MAC packet error rate
4	200	100	50	$1e^{-6}$	0
8	400	250	100	$1e^{-3}$	0.1
12	1000	500	250	$1e^{-2}$	0.2
			500	$1e^{-1}$	0.3
					0.4
					0.5
					0.6
					0.7
					0.8
					0.9

Table 6.6. Parameters used in the evaluation of the streaming solution.

Dropping of outdated packets is introduced in our system to ensure that the bandwidth is not wasted on sending data that will arrive too late for the decoder to process it. The dropping of outdated packets is implemented within EL buffers as described in Chapter 4. The time that data can spend in the buffer is limited by the value called *EL data lifetime*.

The network appraiser keeps the history of estimated bandwidth values over a period of time called *observation window*. Furthermore, the values are filtered to remove 'noise', i.e. short-term fluctuations. The bigger window allows for more filtering and, as a result, less sensitivity of the system towards short-term fluctuations.

As discussed in Chapter 4, under certain conditions the bandwidth estimated by the network appraiser represents the minimal value of the available bandwidth and not the estimation of the real available bandwidth. To compensate for the unaccounted bandwidth, a value called *bandwidth addition* is added to the measured bandwidth.

Layer configurations differ from each other by the number and bit rate of layers. The minimal difference in bit rate between layers is defined by *step rate*, which is the step used by the layer configurator for going through the possible bit rates for the layers.

6.2.2 Research questions, hypotheses, and measurements

The system should satisfy real-time requirements for the streaming. As mentioned in Chapter 1, the start-up latency should not exceed 2 seconds, which corresponds to buffering of 50 – 60 frames at the receiver.

The first question we want to answer is: under what network conditions (i.e. TCP PER and MAC PER) can the 2 seconds latency be guaranteed and what is the quality of the delivered video under these conditions. The hypothesis is that an increase of TCP PER or MAC PER increases the start-up latency and decreases the quality of the video.

The second question is how the choice of the internal parameters influences the latency of the system and, more important, the quality of the delivered video. The following hypotheses are formulated.

- Longer *data lifetimes* increase the quality of the delivered video, since more EL frames have a chance to overstay a bandwidth drop and to get successfully transmitted to the receiver.
- Larger *observation windows* decrease the quality of the video. A large observation window results in low utilization of short-term bandwidth increases, which, in turn, implies low quality of the delivered video.
- Bigger values of the *bandwidth addition* increase quality of the delivered

video. When the available bandwidth increases significantly, a small bandwidth addition may be insufficient for compensation for the unaccounted bandwidth, which leads to the bandwidth under-utilization and, consequently, low video quality. An addition that puts the estimated bandwidth above the real available bandwidth has little effect on the system behavior, since an overestimation of the real available bandwidth is detected upon the transmission of "oversized" frames.

- Smaller *step rates* increase the quality of the delivered video, due to finer bit-rate allocation to the video layers.

To answer the questions above, during the experiment the following characteristics where measured:

- *Minimal latency (or start-up latency)* defines the time that is spent from the moment the streaming starts until the moment the decoding of the video begins. The decoder should be able to process every frame within a frame period after the preceding frame is processed. The transcoder also produces one frame every frame period. If the delivery of a frame takes longer than a frame period, the frame is late for the decoder. To avoid that situation, a certain number of frames has to be buffered, i.e. stored at the receiver, before the decoder starts working. The longer the transmission takes, the more frames should be buffered. The buffering mainly targets BL frames, since the absence of EL frames is not critical for the system – the decoder can use only BL. During the experiment we calculated the number of BL frames that should be buffered before the start of the decoding, so during the whole session the decoder can process every BL frame on time.
- *PSNR under minimal latency* is the average quality of the delivered *and* decoded video if the decoding starts with the minimal latency, as it is defined above.
- *PSNR under* 2 *seconds latency* is the average quality of the decoded video if the decoding starts in 2 seconds after the beginning of the streaming application. The 2 seconds buffering (50 – 60 frames) is the maximum allowed buffering as discussed in Chapter 1. When the transmission conditions are not good and the buffering is insufficient for ensuring timely decoding of BL frames, the late BL frames are not processed, so it is assumed that the last decoded frame is given to a renderer.

The results of the simulation are shown below. There are two separate cases that are studied: influence of network conditions as external parameters and internal settings of the streaming solution.

6.2.3 Network parameters

As mentioned above, the network conditions are given in terms of error rates – protocol packet error rate (TCP PER) and MAC layer packet error rate (MAC PER). Figures 6.11- 6.13 present relations between measured system characteristics and the value of error rates. The horizontal scales of the figures give values of MAC PER, while the lines on the graph are presented for different TCP PER values.

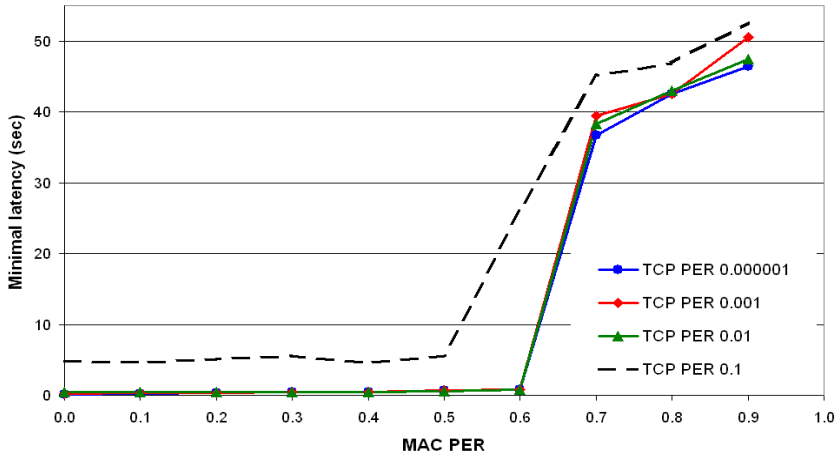

Figure 6.11. Minimal latency for various packet error rates.

Figure 6.11 shows that the higher MAC PER or TCP PER results in a bigger latency. The minimal latency is directly dependent on the transmission throughput. The longer it takes for the system to send a frame to the receiver, the more frames have to be buffered.

Figure 6.12 shows that the quality decreases with increasing MAC PER. The influence of TCP PER on the quality is clearly visible for TCP PER value of 0.1 , whereas the lower values of TCP PER have less impact on the results. These effects could be explained as follows. If the network conditions are poor (MAC PER > 0.5 or TCP PER > 0.01) the streaming solution cuts off the production of ELs and decreases significantly the bit rate of the BL. The aim of these actions is to decrease the amount of data that is offered to the transmission protocol and, in turn, make sure that the time between transmission of two successive frames is close to the frame period. Less layers and low bit rates result in a significant decrease in the quality of the video, as shown in Figure 6.12.

As we discussed in Chapter 1, the developed system should allow real-time video streaming, which means that buffering in the system should be limited to 2 seconds. A very important characteristic of the system behavior under various

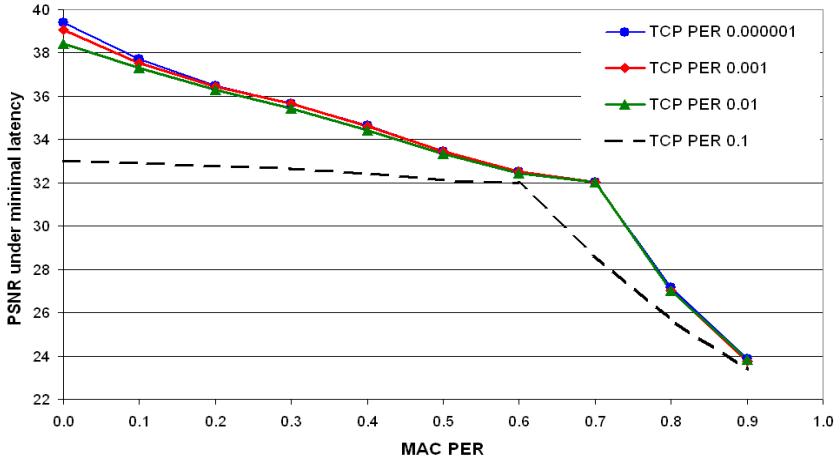

Figure 6.12. Video quality (PNSR) for various packet error rates in the system with minimal latency.

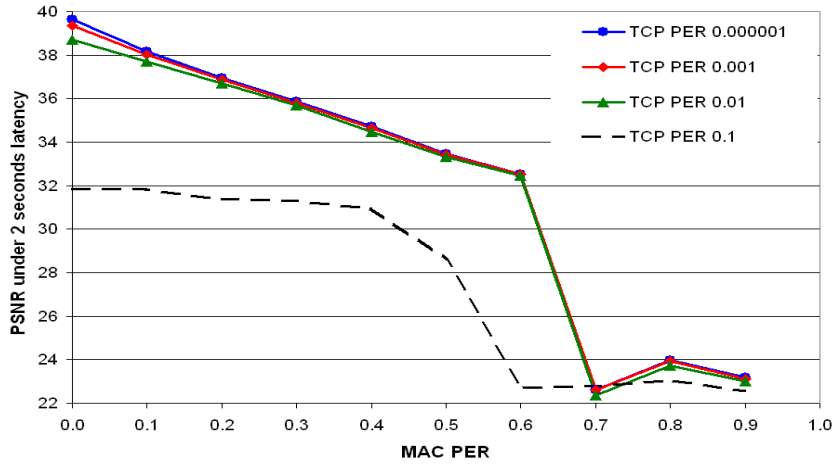

Figure 6.13. Video quality (PNSR) for various packet error rates in the system with 2 second latency.

network conditions, therefore, is the average quality that can be delivered by the system when the start-up latency is at most 2 seconds. The quality of the video for operating with 2 seconds start-up latency is shown in Figure 6.13.

The figure demonstrates that two seconds start-up delay allows a sufficient buffering to the receiver, when MAC PER is less than 0.6 and TCP PER is less than 0.1. The quality, therefore, is high (with better quality for lower PER). If the packet error rates are high, the system transmits mostly BL frames. Moreover, due to a number of retransmissions some of the frames from EL or, even, from BL arrive at the receiver too late to be processed. Because the system operates under strict latency requirements, these frames are dropped at the receiver side without processing. Therefore, the quality delivered by the developed streaming solution is lower when the processing of late frames is not allowed.

6.2.4 Internal parameters

The studied network conditions represent a system with an either slightly, moderately, or heavily disturbed wireless link and none or plenty of competing traffic in the network.

The results show that only three internal parameters have a significant influence on the behavior of the developed streaming solution: EL data lifetime, observation windows and step rate. The fourth parameter, bandwidth addition, does not influence the behavior of the system.

- *Bandwidth addition.* Equation (4.3) in Chapter 4 shows that the bandwidth addition, B^e , is added only if the protocol buffer fullness, S_t is zero. In the tests, the value of bandwidth addition had no influence on the outcome. The developed system reacts fast to an increase in the available bandwidth by increasing the bit rates of the produced video, keeping the protocol busy and avoiding under-utilization of the bandwidth.
- *EL data lifetime.* The effect of different settings for the time that EL data can spend in the buffer is seen only on an undisturbed or moderately disturbed wireless link with no competing traffic in the network. Under these conditions, the number of buffers at the receiver that is needed for obtaining a video of maximal quality increases with increasing EL data lifetime. The effect can be explained as follows. The good network conditions allow to send all layers of the scalable video in time in most of the cases. Sometimes, however, there could be a small short-term drop in the bandwidth availability of the wireless medium. In this case, frames of the highest EL or, in rare situations, of all ELs get delayed in the output buffer of the transcoder. After the bandwidth drop the frames are sent to the receiver. A bigger value of EL data lifetime allows the frames to spend more time in the buffer, so the longer

drops in the bandwidth could be overcome. The frames, however, could be, and usually are, late for the decoding if the receiver has small buffers. When the buffering is unlimited the frames are used by the decoder, increasing the quality of the video.

A more important issue is that the EL data lifetime has no influence on the system behavior for other network conditions (heavily disturbed wireless link or a presence of competing traffic). This is because drops of bandwidth availability under poor network conditions are neither small nor short-term. The developed system detects the drops and timely changes the layer configuration, decreasing the amount of video data that is produced by the transcoder. So, while EL buffers are getting fuller following the bandwidth drop, the transcoder terminates production of additional EL frames. That action effectively prevents application buffers overflow.

- *Observation window.* Changing the size of the observation windows changes the sensitivity of the streaming solution towards variations in the available network bandwidth. A smaller window allows the system to react to insignificant events, whereas a large window averages out small bandwidth changes allowing to concentrate on big changes only. The test results show that an increase of the observation window decreases the quality of the video that is delivered by the system, while also decreasing the number of layer-configuration changes. A system that is more sensitive to changes in the bandwidth availability tries to change layer configuration immediately after the bandwidth change increasing bandwidth utilization, but risking that the very next moment the chosen configuration will be invalid due to the next bandwidth change.

  Thus, the choice of the observation windows is a trade-off between the quality of the video and the number of layer-configuration changes. The latter, in turn, influences the user's perception of the video with many changes being perceived badly as discussed in Chapter 1.

  The influence of the size of the observation window on the system is negligible when the network conditions are extremely poor. The tests show that for a moderately or heavily disturbed wireless link and plenty of competing traffic in the network (TCP PER = 0.1 , MAC PER > 0.5) the size of observation window has no effect on the system. This is because the available bandwidth is constantly low during the whole transmission.

- *Step rate.* A lower step rate allows the system to inspect more layer configurations, which, hypothetically, should enable a better utilization of the bandwidth. The tests results show that the values of the step rate that we've chosen for the tests have a very small influence on the behavior of the sys-

tem. Only in the case of an undisturbed wireless link and no competing traffic in the network, the streaming solution made a fine-tuning of the used layer configuration by adjusting bit rates of the layers with steps smaller than 250 Kbps and improving the quality of the video by as little as 0.1 dB. In the rest of the cases the adjustments of the bit rates were in a scale of 250 Kbps – 500 Kbps.

A small, 50 Kbps , increase in the bit rate of a layer brings insignificant quality improvements, but increases the risks associated with the transmission. Therefore, small bit-rate changes are not favored by the streaming solution. Moreover, a bigger step rate requires less computational effort from the system, since the number of inspected configurations decreases. Thus, the choice of the step rate value is a trade-off between the performance of the sender and the quality increase that can be made by the added bit rate.

6.2.5 Conclusions

The system performance depends significantly on the network conditions and is only slightly influenced by the chosen internal parameters. The required 2 seconds start-up latency cannot be guaranteed when the TCP PER is higher than 0.01 or when MAC PER is higher than 0.6 . The basic hypothesis that an increase of TCP PER or MAC PER increases the start-up latency and decreases the quality of the video is confirmed.

Answering the question how the choice of internal parameters influence the latency of the system and the quality of the delivered video, the following hypothesis was confirmed:

- Larger *observation windows* decrease the quality of the video.

The following hypotheses were confirmed partly (the conditions are described in the section above):

- Longer *data lifetimes* increase the quality of the delivered video.
- Smaller *step rate* increase quality of the delivered video.

The following hypothesis was rejected:

- Bigger values of the *bandwidth addition* increase quality of the delivered video.

6.3 Terminal management

In this section we evaluate the quality of controlled decoding in comparison with the un-managed decoding. In addition we evaluate the difference between *network-aware* and *network-unaware* controller solutions.

6.3.1 Tests preparation

As input for the evaluation we use decoding-trace files of two video sequences consisting of $100,000$ frames. The video sequences were encoded into SNR-scalable video consisting of three layers (one base and two enhancement). The bit rate of each layer was 1 Mbps.

We performed decoding of the test video using

1. only the base layer (quality level q_0),
2. the base layer and one enhancement layer (quality level q_1), and
3. all the layers (quality level q_2).

The time measurements for the decoding were stored in trace files. The trace files contain time measurements for decoding a frame using the base plus zero up to 2 enhancement layers. The time measurements provide the amount of CPU time spent solely on the decoding.

We use trace files from the one sequence to create controller strategies by solving the Markov Decision Process as described in Chapter 5. Trace files of the other file are used as a test input for evaluation of the strategies.

We used two decoders to create the trace files: a software decoder on a PC and a hardware-based decoder on a MT3 set-top box (both decoders are discussed in the sections below).

Software decoder

The software decoder is implemented as an application that decodes an BL and EL using a non-scalable decoder and then merges the layers together (see Figure 3.2, Type-I decoding process). The decoding of the BL and EL is performed in parallel using a time-division technique and is followed by a summation. For example, for a three-layer video, the decoder processes frame FB^j, then processes frame FE_1^j, after that processes frame FE_2^j, and, finally, performs summation of the frames.

The following observations were made while timing the decoding process.

1. Decoding of the BL takes more time than decoding of the EL. B frames in the EL consist of macroblocks with uni-directional prediction as opposed to mixed macroblock types in B frames. Moreover, motion vectors for all predicted macroblocks in the EL are zero, which makes the motion-compensation task easier. Additionally, for enhancement layers with a low bit rate, many empty macroblocks are skipped during encoding, which results in a fast decoding. We found that over 40% of macroblocks are skipped in a EL of 1 Mbps.
2. On a PC, the summation of the layers does not take a significant time in comparison with the decoding of a video layer.

Hardware-based decoder

The platform for a hardware-based decoder is the Viper-1 [74] chip integrated in a MT3 set-top box. The MT3 platform was developed by Philips Digital Networks for advanced set-top-box products featuring enhanced digital TV services, Internet services, local storage and streaming media. The Viper chip contains a MIPS processor for general purpose processing and a TriMedia processor [75] for the media processing functions, accompanied by hardware support for video decoding. The chip can decode up to 6 MPEG-2 video streams in parallel (using a time-division technique).

The organization of the decoding process corresponds to Type-I (see Figure Figure 3.2). Every MPEG-2 decoder works in a separate thread, receives data from the input buffer and sends decoded frames to the output buffer. After all the decoders have updated the output buffer (i.e. a new frame has been processed), the summation function is called. The summation function is implemented as a separate module with its own thread. It takes input from the decoders and passes the results of the summation to the output buffer. Every frame period (e.g. 40 ms for a PAL video stream), a frame is taken from summation's output buffer to the display. If the buffer is full (no frame was taken between the filling), the summation task is blocked. This ensures that we have a fixed maximum number of frames on which the summation module can work ahead.

Time measurements show that the summation is the most CPU consuming part of the video processing. It consumes around 25% of the CPU when three layers are summed up, 18% for two layers and less than 0.5% when only the base layer is present.

At the same time, the CPU consumption of the decoder tasks is negligible. In addition to the decoding of video data which is performed by a hardware unit, a pre-processing of video streams is done in software. As a result, each decoder tasks consumes up to 2% of the CPU for the needs of stream parsing, headers decoding, etc.

Time traces

Based on the processing time recorded in the trace files, we calculate, for every quality level, a cumulative distribution function (CDF) of the time required to process a frame. Figures 6.14 and 6.15 show CDF of the time for hardware-based and software decoders.

6.3.2 *Controlled* vs. *best-effort* decoding

For the first evaluation we compare the *controlled* solution with the *best-effort* solution. The *best-effort* solution decodes as many layers as possible within a given budget. The processing of layers progresses sequentially, starting with L_B

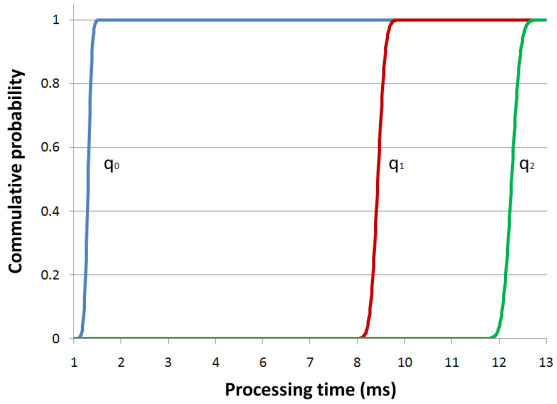

Figure 6.14. A cumulative distribution function of the time required to process a frame at a given quality level with the hardware-based decoder.

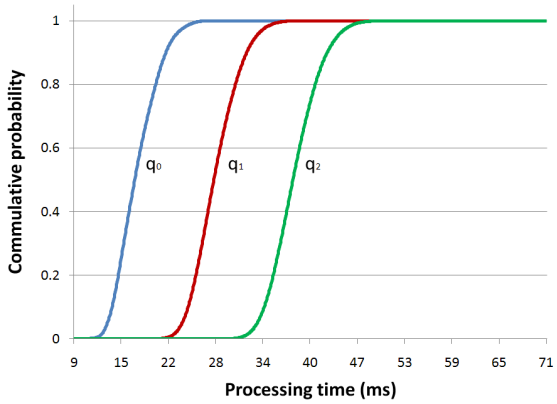

Figure 6.15. A cumulative distribution function of the time required to process a frame at a given quality level with the software decoder.

and $L_{E,1}$, until all available layers are processed. On a deadline, the decoding result (if any) is shown and the decoder starts to process the next frame. If BL is not decoded by the deadline, a deadline miss occurs. If the decoding of a frame finishes before its deadline, the remaining budget can be transferred to the next frame.

The *controlled* solution performs the decoding in accordance with a pre-calculated strategy. The comparison is performed under the assumption that all layers are always available to the decoder (i.e. we have a stable input).

Setup

We define the parameters of MDP as follows. The upper bound on relative progress is set to 2, which assumes that we can work up to two decoded frames ahead (it is assumed that the output buffer of the decoder can store at most two frames). Since the perceived quality of the video depends on the actual bit rate of a video stream and given the fact that we increase the total bit rate by the same value with every quality level, we define the reward for being at a particular quality level as 2, 4, 8 for levels 0, 1, and 2 respectively [3]. In the revenue function, we set the penalty for missing a deadline to $100,000$. This means that we allow around 1 deadline miss per 12000 frames or, in other words, on average we skip 1 frame per 8 minutes of video. The penalties for increasing the quality level are set to 5 and 50 if the quality level is increased by 1 or 2 respectively. For decreasing the quality level the penalties are set to 10 and 100 for going down by 1 or 2 levels respectively. For this test we assume to receive all layers (i.e. a stable input), so it is possible to choose freely any quality level. Since we have a stable input, for calculation of the strategy we set $Y_{\overline{q_m},\overline{q_{m+1}}} = 1$ if $\overline{q_{m+1}} = 2$ and $Y_{\overline{q_m},\overline{q_{m+1}}} = 0$ otherwise. For the evaluation we consider budgets from 4 to 40 ms, with steps of 1 ms.

Results

First, we look at the results for the software decoder. Most attention should be devoted to the deadline misses for both solutions, because a deadline miss causes visible stalls of the video frames during video playback. Figure 6.16 presents the percentage of deadline misses as a function of the budget. The solutions perform almost identical for budgets below 15 ms and above 25 ms. This can be explained as follows. With low budgets none of the solutions has resources to successfully process BL frames, because, on average, 17 ms are required to process these frames (see Figure 6.15). At the same time, having a budget bigger than 25 ms guarantees that any BL frame can be decoded on time, so no deadline misses can occur. For budgets from 15 to 25 ms, the *controlled* solution outperforms the *best-effort* solution.

[3] We based the choice on the data from our user tests and from the results of user tests in [37]

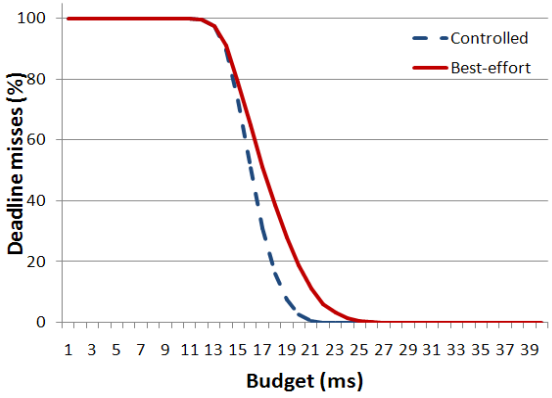

Figure 6.16. Comparison of the *controlled* and a *best-effort* solutions – deadline misses of the software decoder (as a percentage of the total number of processed frames).

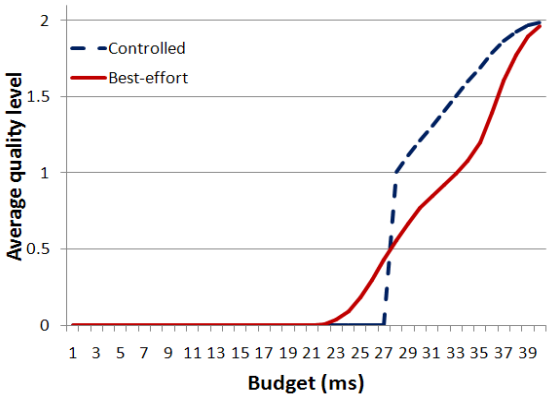

Figure 6.17. Comparison of the *controlled* and a *best-effort* solutions – average quality level of the software decoder.

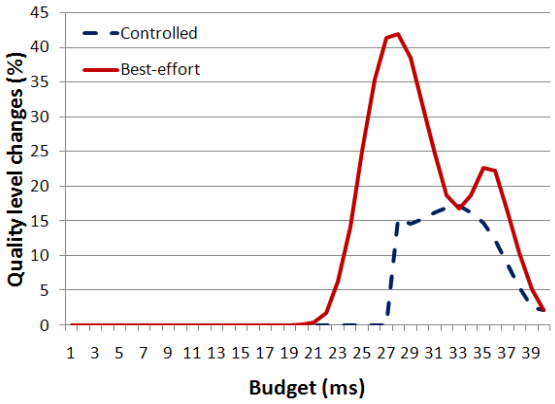

Figure 6.18. Comparison of the *controlled* and a *best-effort* solutions – quality level changes of the software decoder (as a percentage of the total number of processed frames).

The average quality level (Figure 6.17) of the *best-effort* solution is higher under low budgets (smaller than 27 ms). However, the penalty for indiscriminate increases in quality level is a higher number of quality level changes (Figure 6.18) and a slightly higher percentage of deadline misses. At the same time, the *controlled* solution allows a quality level increase only when it can guarantee that the number of deadline misses for the given budget will be in the pre-defined limit (which is roughly 1 per 12000 frames, as mentioned above).

Second, we look at the results for the hardware-based decoder. As shown in Figure 6.14, the successful decoding of frames from the BL is possible under the budget of 1 ms, so both solutions experience deadline misses. Under larger budgets, the solutions suffers no deadline misses at all, because processing of any frame from the BL is possible.

Both solutions demonstrate comparable results for the average quality level and the number of quality level changes (see (Figures 6.19 and 6.20). Under budgets between 7 and 11 ms, the average quality level of the *best-effort* solution is higher, but it experiences more quality level changes than the *controlled* solution. Under budgets from 12 ms to 14 ms, the situation is reversed and the *controlled* solution demonstrates a better average quality as well as an increased number of quality changes.

6.3.3 Controlled *network-aware* vs. *network-unaware* decoding

For the second evaluation we compared the *network-aware* solution with the *network-unaware* solution. Both solutions perform the decoding in accordance

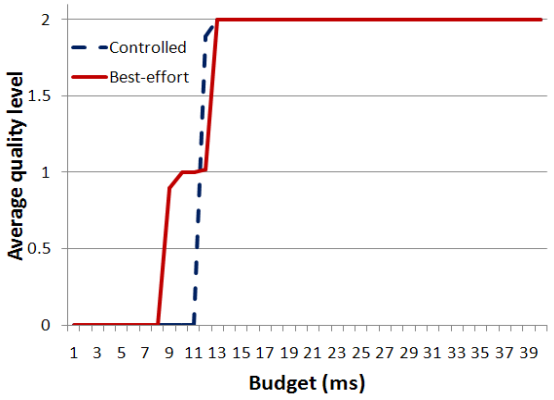

Figure 6.19. Comparison of the *controlled* and a *best-effort* solutions – average quality level of the hardware-based decoder.

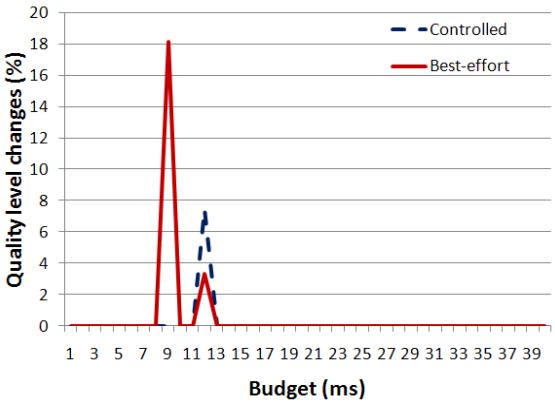

Figure 6.20. Comparison of the *controlled* and a *best-effort* solutions – quality level changes of the hardware-based decoder (as a percentage of the total number of processed frames).

with a pre-calculated strategy. The *network-unaware* solution uses a strategy with all probabilities $Y_{\overline{q_m},\overline{q_{m+1}}}$ equal to $\frac{1}{3}$ (i.e. chances of getting the same amount of layers or any other amount are equal). The *network-aware* solution has probabilities of receiving exactly one, two or three layers, as shown in Table 6.7. We define three different sessions with different probabilities settings. In Session 1, all layers are available to the decoder, so ideally the controller strategy should be defined only by terminal resource limitations. In Session 2, the transmission of the BL is guaranteed and the probabilities to receive one or two Els are equal, so the network condition is as important as the terminal resources. In Session 3, the transmission of the BL needs most of the bandwidth, so the network condition should play the most important role in the controller decisions.

Session	L_B only	$L_B + L_{E,1}$	$L_B + L_{E,1} + L_{E,2}$
1	0.0	0.0	100.0
2	0.0	50.0	50.0
3	70.0	30.0	0.0

Table 6.7. Probability (%) of receiving the given layers.

The rest of the parameters of MDP are defined in the same way as in the previous test. We make a pairwise comparison of the solutions, looking at the average quality level and the number of quality-level changes. We consider budgets from 4 to 40 ms, with steps of 1 ms.

Session 1

For Session 1, both solutions behave in the same way, delivering equal quality and experiencing almost the same number of quality changes. The reason for this is that all three layers are constantly available for processing. Consequently, both solutions take into account only terminal resources (which are equal), resulting in nearly the same strategies.

Session 2

The behavior of solutions differs significantly for the software decoder (Figures 6.21, and 6.22) and stays nearly the same for the hardware-based decoder (Figures 6.23, and 6.24).

First, we look at the results for the software decoder. Figure 6.21 shows that starting from a budget of 29 ms, which allows successful decoding of two layers, the *network-aware* solution does not attempt to increase the quality level. The reason for this is that, according to the network conditions (see Table 6.7), $L_{E,2}$ is not available in half of the cases. Thus, choosing quality level 2 will lead to frequent quality-level changes. Moreover, the solution is aware of the fact that L_B is always available. So, the quality provided by the decoder at quality level 1 is

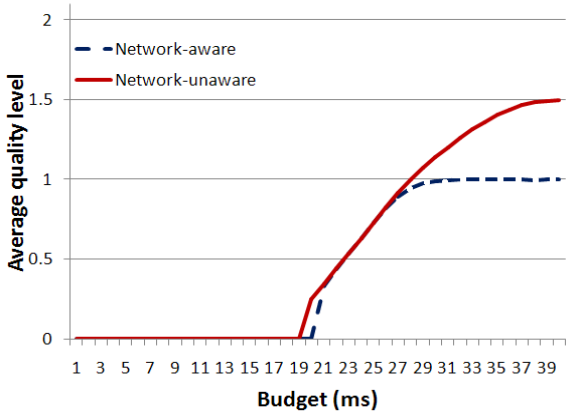

Figure 6.21. Comparison of *network-aware* and *network-unaware* solutions for software decoder – average quality level.

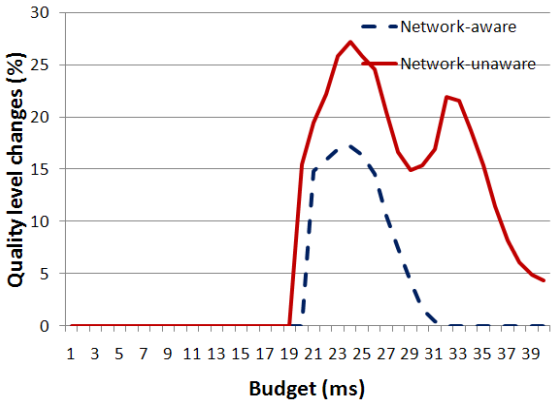

Figure 6.22. Comparison of *network-aware* and *network-unaware* solutions for software decoder – quality-level changes.

average, but stable. From the point of view of the *network-unaware* solution, the probability of receiving L_B and $L_{E,1}$ is higher (66%). Moreover, when processing at quality level 1 there is 33% probability that, due to network conditions, the next frame will be processed at quality level 0. As a result, the *network-unaware* solution increases the quality level, which leads to a higher average quality but also a high level of quality fluctuations (Figure 6.22).

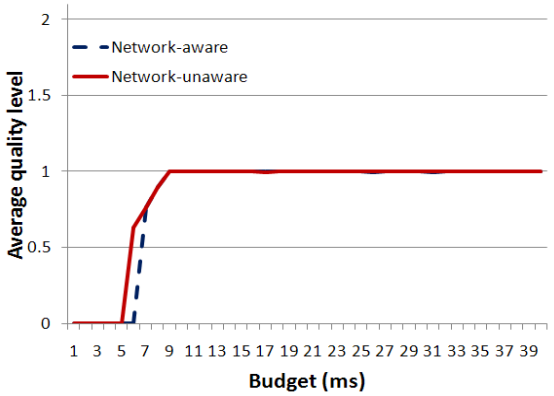

Figure 6.23. Comparison of *network-aware* and *network-unaware* solutions for hardware-based decoder – average quality level.

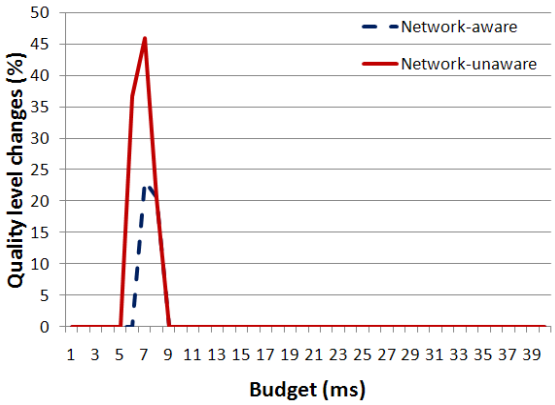

Figure 6.24. Comparison of *network-aware* and *network-unaware* solutions for hardware-based decoder – quality-level changes.

Second, we look at the results for the hardware-based decoder. Figure 6.23 shows that the solutions have identical results for the average quality level, except for the budget of 6 ms. Under the budget of 6 ms the *network-aware* solution

processes all frames at quality level 0, and the *network-unaware* solution attempts to process some of the frames at quality level 1. That results in an extremely large number of quality fluctuations for the *network-unaware* solution. Figure 6.24 demonstrates that, overall, the network awareness leads to fewer quality-level changes.

Session 3

The results for Session 3 are the same again. Since it is not probable that BL and two ELs will be received, the controller is left with the choice between processing one or two layers. However, given that most of the time (70%) only L_B is available, both controllers behave conservatively, trying not to choose quality level 1. Thus, the strategy of the controllers is fully defined by the network conditions.

6.3.4 Conclusions

For the evaluation we've chosen realistic examples, which show the behavior of the controller in an expected home network environment, instead of showing the maximal theoretical gain that can be obtained. The results suggest that the hardware decoder benefits less from the controller than the software decoder. This can be explained as follows.

The effectiveness of the controller depends a lot the total amount of available resources vs. the difference in resource consumption on individual layers. The bigger the difference in consumption, the more difficult for the controller to find a stable strategy. Suppose we have a resource budget of 10 units and the system has two quality levels (1 and 2) that require 7 and 12 units respectively. Let us assume that the system can process at most 2 frames in advance. That means that the controller can preserve at most 20 units. At the beginning, processing at the second level is not possible, because 10 units are available and 12 are needed. So, the controller stays at the first level. Saving 3 units per frame, the controller reaches the preservation limit in 7 frames. At this moment the controller should make a decision whether to continue at the same level, or try to increase the quality level. In the latter case, the saved resources will be used within 10 frames of processing at level two, forcing the controller down to the first level. So, the controller should make a clear trade-off between the quality of the delivered video and number of quality level changes. Note, that for a smaller difference between resource consumption at the layers, the controller strategy is much easier. Indeed, suppose we have the same system, but the processing at quality level 1 requires 9 units and 10.5 units at quality level 2. In this case, the controller can process 20 frames at the first quality level and then process 40 frames at the second quality level. The result would be a higher quality and a fewer quality changes.

In general, the evaluation of the terminal resource management mechanism

demonstrated that the controlled decoder of a scalable video performs better with respect to quality of the output video than a non-controlled best effort approach.

Moreover, controlled decoding allows a significantly better utilization of resources. The controller of the decoder bases its work on the resource preserving algorithm that allows the decoder to continue video processing even if the amount of resources needed by the decoder is small.

The evaluation shows that bringing the network-awareness to the terminal resource management has potential to improve the performance of the controller and, consequently, to increase the quality of the output video.

7

Conclusions and future research

The first stage of the work on this thesis consisted of an evaluation of existing approaches, techniques, algorithms in the domain of video coding, networking and terminal resource management. At that time there was no solution that satisfies the requirements for video streaming in a home network as discussed in Chapter 1 of this thesis. A solution that provides a full management of network and terminal resources was unavailable.

The majority of the research was, and still is, domain specific. In the domain of video coding MPEG-4 FGS gives a good example of network-friendly encoding. That coding technique allows for high utilization of the network resources, but is very challenging for terminal resource management.

In the domain of adaptable video streaming, popular solutions are based on a transcoder that adapts its output according to feedback from the receiver. These solutions, however, experience problems when handling short term bandwidth fluctuations because their reaction time is significantly larger than the bandwidth fluctuations themselves. Moreover, continuously changing the bit rate of video stream results in constant fluctuations of video quality which, in turn, complicates terminal resource management.

In the domain of terminal resource management, the best results are achieved by solutions utilizing a budget scheduler and a quality manager in combination with a scalable video algorithm. These solutions, however, were not directly applicable

in the wireless environment because they are built on the assumption that the input is stable and lossless.

7.1 Revisiting the research questions

While discussing the results of the presented work, we look back at the research questions that are stated in Chapter 1. The first major research questions for this thesis work was: is it feasible to create a system for real-time video streaming over a wireless network to resource-constrained terminals satisfying the given requirements (see Chapter 1)?

The overall home network with multiple senders/receivers connected via a combination of wireless and wired network is too complex to resolve without having a simple underlying solution. Therefore, we looked at the small subsystem consisting of a sender and a receiver connected to each other via wireless/wired links.

This thesis presents a novel approach to building a framework for in-home wireless video streaming to CE devices. In this thesis we use scalable video to transmit video frames in a timely fashion over a wireless network in spite of the fluctuating bandwidth. A terminal resource management thechnique is developed that allows to change the consumption of the video processing at the receiver, thus enabling a smooth video processing on a resource-constrained terminal.

The developed system combines an easy adaptability to wireless bandwidth fluctuations with robustness to partial data losses and supports heterogeneous clients with regard to their buffer availabilities and computing capability limitations. The system reuses existing software systems as much as possible, uses legacy devices, and complies with existing standards. These results are achieved by developing a system that is based on TCP as a transmission protocol for the video streaming, and that uses scalable video that can be decoded by a non-scalable MPEG-2 decoder.

The second major research question was: what methods and techniques should be used and how can they collaborate? This question, in turn, was subdivided into a subset of questions:

- What is the minimal set of components that are necessary to build the system?

  We demonstrated that the system requires three major components: a video producer that creates video streams with a configuration that is best suited to the observed operational conditions, a streaming solution that supports prioritization of the video data and stays in control of the data losses that happen due to poor network conditions, and a decoding application with a built-in controller that enables an efficient resource management at the

receiver.

- What video-encoding technique should be used, so it is suitable for streaming and resource management?

  Our research shows that a video encoding based on scalable video coding is the best choice for both, the streaming and resource management solutions. Short term fluctuations of the bandwidth could be overcome with the help of scalable video coding, whereas a transcoder of a scalable video enables better handling of long term fluctuations. Scalable video coding allows for resource management at the receiver side. Changing the number of decoded layers changes resource consumption of the decoding algorithms altering, at the same time, the output quality of the decoder. The investigation of the different scalability techniques shows that although the temporal scalability is very easy to implement and the spatial scalability is historically the most popular technique, the signal-to-noise ratio scalability provides the greatest flexibility in creation of multiple layers and allows for the best compression efficiency under the requirements that we have for the encoding.

  We developed a SNR scalable video-encoding technique that produces multiple layers, BL and one or more ELs that can be decoded by a standard non-scalable MPEG-2 decoder. Two modes for creation of an EL were proposed. In I-frame mode an EL contains only I frames. In B-frame mode an EL consists of GOPs that have IB_n structure. The evaluation of the approaches demonstrated that an EL that consists of a single GOP where the first frame is I-frame and the reset of the frames are B frames delivers a video of the highest quality for a given bit rate.

  We proposed an SNR-specific encoder as an alternative to a cascade of non-scalable encoders. The specific encoder produces video of a higher quality than the cascade, while a solution that is based on a cascade of non-scalable encoders is simpler to implement. Thus, the presentation of the two approaches allows a trade-off between the quality of encoding and the computational complexity.

  After the evaluation of the SNR-scalable encoding, we found that the best approach for coding SNR-scalable video is to create the largest possible BL within the current environment settings and to improve it with a smaller ELs.

- What connectivity solution should be used in terms of transmission protocol and the low levels of network communication. In particular, is it possible to reuse an existing protocol and work only on application level?

  The changes in the bandwidth availability of a wireless network differ from the wired counterpart. In addition to long-term changes that are caused most of the time by the competing network traffic, we need to account for

short term fluctuations that originate from the physical characteristics of the medium.

Wireless networks experience data losses that we cannot avoid. However, it is not necessary to follow the network behavior blindly – if losses of video data are unavoidable, we need to control them, i.e. choose by ourselves what part of the data should be dropped as a result of network-bandwidth limitations.

In this thesis we presented a solution for streaming a scalable video over wireless/ wired links that is based on TCP. The choice of the protocol prevents data losses during the transmission. The potential losses are traded for the delays in the transmitted data. To compensate for the delay the solution either decreases the amount of data that is offered to the transmission protocol, or forces the video producer to decrease the bit rate. The first mechanism is implemented by EL frame dropping and is intended for handling short-term bandwidth variations, whereas the second mechanism handles long term bandwidth variations. The frame dropping is the simplest approach to implement. The adaptation of the video-producer configurations is an approach that requires a full spectrum of activities including network-bandwidth estimation, loss estimation and quality estimations for the transmitted video. A distributed control of video streaming is always very complex to develop and implement, so a solution with centralized control at the sender side is preferred. The major advantage of both suggested approaches is that all functionality can be fully implemented at the sender side.

The developed streaming solution assigns priorities to layers of a scalable video to guarantee that transmission of the BL does not suffer from ELs and the higher EL has no influence on the transmission of the lower, more important, ELs. The streaming is based on a single-stream TCP, so the prioritization scheme is valid over the whole transmission path and does not depend on the network environment.

The proposed streaming solution is not video-encoding specific and can be used with other scalable video coding techniques.

- What management technique allows to control resources at the receiver side, while guaranteing a good watching experience to the user?

  In this thesis we presented a resource management solution for the receiver of a scalable video. The solution is based on the use of a controller that changes the number of layers processed by the decoder and controls the resource consumption of the decoder. The controller bases its decisions on a pre-calculated strategy. A set of strategies is created offline by means of MDP.

The successful functioning of the proposed solution assumes that a terminal has basic components for the resource management – a budget scheduler, which takes care that the resources assigned to an application is guaranteed with a certain amount of resources and the application does not exceed that amount, and a quality manager, which is responsible for definition of the priorities of various applications and distribution of the system resources among the applications.

In addition to the *network-aware* resource management solution, we present a simpler solution that does not take the network conditions into account. The simpler solution requires a substantially lower number of controller strategies and yet shows a good result with respect to the output video quality. Thus, the choice between the two solutions allows a trade-off between the computational complexity and the quality of the resource management.

The last major research question was: what are the practical limitations of such a solution?

The evaluation of the system components discovered some practical limitations of the developed solution. It is shown that the bit-rate overhead associated with scalable video coding can be very high if the balance between the bit rates of BL and EL is not right. A small BL requires significantly larger EL in order to deliver a video of a good quality. Therefore, configurations with a relatively large BL and small EL are preferable.

The evaluation of the streaming solution suggests that the solution should perform well in a typical home network. The conditions under which the streaming does not satisfy the requirements stated in Chapter 1 are: a high packet error rate on the MAC layer (> 0.6) and/or a high protocol packet error rate (> 0.01).

The major limitation of the terminal resource management is the necessity to have a budget scheduler at the receiver. Moreover, the performance of the solution drops significantly, in terms of delivered video quality, if the amount of resources that are given to the decoding application is close to the amount of resources that are needed to decode a single BL.

7.2 Future plans

This section suggests possible improvements to the system of wireless video streaming and identifies the scope of future work in this area.

7.2.1 Video coding

In Chapter 3 we discussed three approaches for making a transcoder – re-encoding, open-loop SNR transcoding and closed-loop SNR transcoding. Only the open-loop SNR transcoder and the re-encoding based transcoder were used during the work

on this thesis. The elaborated research and development of the transcoders for scalable video must take place if the system should be developed further.

The open-loop transcoding and re-encoding have unique properties that clearly differentiate them from each other. The computational complexity of the open-loop transcoding is low, whilst it is very high for the re-encoding. On the other hand, the quality of the video that is produced by the re-encoding method clearly tops the quality of the open-loop transcoder.

Closed-loop transcoding takes the best from the two above-mentioned approaches. The expected result should combine the quality of the re-encoding with not-so-high complexity of the open-loop transcoder.

7.2.2 Video streaming

For scenarios where content is transmitted simultaneously to more than one receiver, an adapted streaming solution is needed. For such system, an approach that is based on multicast is more efficient than unicast in terms of bandwidth usage. Our solution is ready to deal with bandwidth limitations as well as with heterogeneous terminals by layered multicast. The system should transmit the layers of scalable video coding to separate multicast groups, so that the receivers can choose individually which groups to join or leave, i.e. which layers to receive. Although layered multicast is very well suited to the transmission of layered video, some of the specific problems regarding enforcement of the timing aspects and detection of the available bandwidth need to be resolved.

With multicast, receivers can join and leave a multicast group independently at any time by sending join or leave messages to the corresponding multicast address. A problem exists for multicast solutions in wireless networks where a receiver may leave the group due to the bad reception conditions without sending a message to the rest of the system. If a wireless receiver is situated on the border of a communication range and, as a result, frequently joins/leaves the group, the system should be able to identify that the receiver is unreachable either because of the network conditions or because it has left the group. Failure to identify the state of the receivers leads to extra waiting time, making timing requirements difficult to satisfy. Moreover, because receivers join and leave the group, there are not only changes in channel conditions, but the optimum transmission rate to a multicast group also varies even more than with the unicast solution. Once we have resolved the multicast issues, we can use the framework developed for simultaneous video transmission to heterogeneous terminals.

7.2.3 Terminal resource management

In our current implementation a system developer has to define explicitly the values for the video quality settings for a terminal (e.g. what is the relative quality value

for a certain number of processed layers) at design time. It would be highly beneficial to have a system that can derive the quality values through analysis performed at run time and adapt the terminal controller strategy accordingly. The system should provide good initial quality estimates for a set of layer configurations, learn quickly from the incoming video streams and, at the same time, adapt controller strategies quickly and effectively. It might be useful here to work towards machine learning. Another approach might be to use statistics relating to the quality of a scalable video coding to create recommendations for assigning some "average" values when dealing with a particular implementation of a transcoder/encoder.

A

Video processing basics

A video signal is made up of a sequence of successive still images (frames). The illusion of motion, i.e. the "liveliness", arises from the rapid rate at which the frames are displayed. Frame rate is an important factor for the quality of an image sequence. Whether the frame rate is high enough depends on the image sequence content – in particular the amount and speed of motion. Also, video quality is affected considerably by the resolution of each individual still image, i.e. the number of image pixels used to present it. Table A.1 gives a list of the most common image resolutions.

The resolution of a standard TV picture is 720x576 pixels and the frame rate is 25 or 30 frames per second. The transmission of a digital video signal requires a considerable channel capacity (more than 200 Mbps). From a storage perspective, one DVD (4.5 gigabytes) is capable of storing about 3 minutes of video. The numbers given above illustrate the need for efficient compression of video information.

There are two modes of compression, 'lossless' and 'lossy'. 'Lossless' compression retains the original data so that the individual image sequences remain the same. Compression is achieved by exploiting the similarities or redundancies that exist in a typical video. For example, consecutive frames in a video sequence exhibit temporal redundancy since they typically contain the same objects, perhaps undergoing some movement between frames. Within a single frame there is spatial

Application domain	Resolution	'NTSC' – USA	'PAL' – EU
	SQCIF	128x96	128x96
Desktop videophone, Internet streaming video	QCIF	176x144	176x144
Internet streaming video, video conferencing	CIF	352x288	352x288
	$\frac{1}{2}$ D1	360x486	360x576
	$\frac{3}{4}$ D1	540x486	540x576
Computer pictures and games	VGA	640x480	640x480
Television broadcast (SD)	D1	720x486	720x576

Table A.1. Video resolutions.

redundancy as the color and intensity of nearby pixels are often correlated. The compression rate is usually no better than 3:1 [2]. The low compression rate makes most lossless compression less desirable.

'Lossy' compression methods reduce irrelevancy in the video signal by removing image information that is unlikely to be noticed by the viewer. Only video features that are perceptually important are coded. The compression is achieved at the expense of quality. Lower quality can mean a lower resolution, lower frame rates and imprecise representation of image pixels.

A combination of the two methods is used in modern video compression standards.

A.1 Video-encoding standards

There are two bodies of experts who develop the standards for video compression: the International Telecommunication Union's (ITU) Video Coding Experts Group (VCEG) and the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) Moving Picture Experts Group (MPEG). VCEG develops standards for video coding methods aimed at video conferencing applications. MPEG looks at a wide variety of applications, developing standards for compression, decompression, processing and coded representation of a video. These groups form the Joint Video Team (JVT) that has developed some of the most popular video standards – H.262/MPEG-2 and H.264/MPEG-4 Advanced Video Coding (AVC).

Table A.2 gives an overview of the most popular video coding standards, a description of their application domain and the year they were introduced.

H.261, defined by the ITU, is the first major video compression standard. The standard was developed for video conferencing. The complexity of the coding techniques is low since the videophones require simultaneous real-time encoding

Video Compression Standard	Primary Intended Applications	Bit Rate	Year
H.261	Video telephony and teleconferencing	$20-320$ Kbps	1990
MPEG-1	Video on digital storage media	1.5 Mbps	1993
H.262	Video conferencing, video telephony	$20-200$ Kbps	1994
MPEG-2	Digital television	$2-20$ Mbps	1994
H.263	Video conferencing, video telephony, Internet video streaming	$20-320$ Kbps	1995
MPEG-4	Object-based coding, synthetic content, interactivity, video streaming	Variable	1999
H.264	Internet video streaming	$20-200$ Kbps	2002
MPEG-4/AVC	Internet video streaming, video over 3G wireless	$20-200$ Kbps	2002

Table A.2. Video compression standards.

and decoding. The standard is intended for ISDN networks and therefore supports bit rates from 20Kbps up to 320Kbps.

MPEG-1 is the first video compression algorithm developed by the ISO. The main application for this standard is storage and retrieval of video and audio on digital media such as video CD. MPEG-1 may achieve better compression than H.261 due to more complicated encoding techniques, which are not possible on resource-limited telephony devices.

MPEG-2/H.262 arose from the first joint effort by the VCEG and MPEG. While H.262 was meant as an improvement for video compression in video conferencing applications, MPEG-2 was developed specifically for digital television. Due to the broad scope of applications, MPEG-2 became the most successful video compression standard.

H.263 was developed as an improvement to H.261. The standard aimed to provide better quality at lower bit rates to satisfy slow connections via telephone modems (28.8 Kbps). Since H.263 generally offered improved efficiency over H.261, it became used as the preferred algorithm for video conferencing, with H.261 support still required for compatibility with older systems. H.263 expanded over time to become H.263+ and H.263++. H.263 and its annexes formed the basis for many of the coding tools in MPEG-4.

MPEG-4 is inspired by the success of MPEG-2. MPEG-4 was initially meant to provide improved error robustness over wireless networks, better support for low bit rates and tools to merge graphic objects with video. The latter objective has not yet found any significant application in products. Unlike MPEG-1 and MPEG-2, where Internet protocol (IP) delivery was not explicit, MPEG-4 fully embraces IP networking and is targeted at Internet streaming and mobile streaming. Moreover,

some of today's most popular proprietary codecs, such as DivX and Xvid, are based on the MPEG-4 standard.

H.264/ MPEG-4 AVC is the latest development of the JVT of the ITU. The standard allows significantly better compression efficiency than the standards previously used (most authors agree that it offers compression that is about 2 times better than that with MPEG-2).

A video compression system comprises an encoder and a decoder with a common interpretation for compressed bitstreams. The encoder takes the original video and compresses it to a bitstream, which is passed to the decoder to produce the reconstructed video. The standards specify neither the encoder nor the decoder. Instead, they specify the bitstream syntax and the decoding process. The bitstream syntax is the format for representing the compressed data. The decoding process is the set of rules for interpreting the bitstream. The specific implementation is not standardized and this allows different designers and manufacturers to provide standard compatible enhancements and thereby to differentiate their work. The encoder process has deliberately not been standardized. The only constraint is that the encoder produces a syntactically correct bitstream that can be properly decoded by a standard compatible decoder.

As mentioned in Section 1.1.2, the MPEG-2 standard is used in the work related to this thesis. The basic principles of the video compression techniques used in MPEG-2 are explained in the next section.

A.2 MPEG-2 encoding

During the encoding process the image is not treated pixel by pixel; instead the whole image is divided into a number of macroblocks consisting of 16x16 pixels. Each macroblock is divided into four blocks of 8x8 pixels. The separation into macroblocks and blocks lasts from encoding until decoding. During the encoding, macroblocks are combined into slices, which make up a picture. Pictures are combined into GOPs (groups of pictures), which in turn form a sequence (Figure A.1).

In MPEG, frames (pictures) can be encoded in three types: intra-frames (I frames), forward predicted frames (P frames), and bi-directional predicted frames (B frames).

An I frame is encoded as a single image, with no reference to any past or future frames. Each 8x8 block is first transformed from the spatial domain into a frequency domain using the DCT (Discrete Cosine Transform) [76], which separates the signal into independent frequency bands (Figure A.2). From now on the block is represented by DCT coefficients. Most frequency information is in the upper left corner of the resulting 8x8 block. After DCT, the data is quantized. The quantization reduces the number of bits required to represent DCT coefficients. Higher

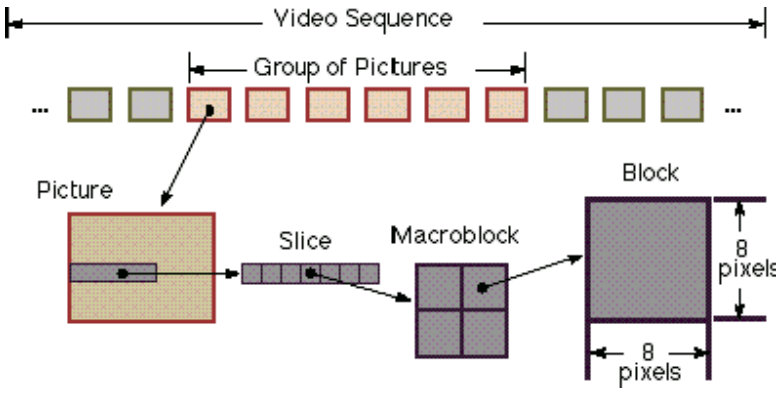

Figure A.1. The MPEG data hierarchy.

frequency coefficients are usually quantized more coarsely than lower frequency coefficients.

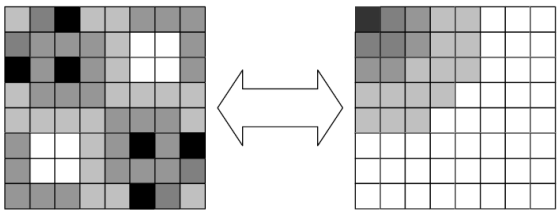

Figure A.2. The DCT operation. The original picture [left] and the corresponding DCT mapping [right].

The resulting data is run-length encoded [77] in a zig-zag ordering to optimize compression. This zig-zag ordering (Figure A.3) reads the two-dimensional array from the most significant element to the least significant and, thus, produces longer runs of 0s by taking advantage of the fact that there should be little high-frequency information (more 0s as one zig-zags from the upper left corner towards the lower right corner of the 8x8 block) [46].

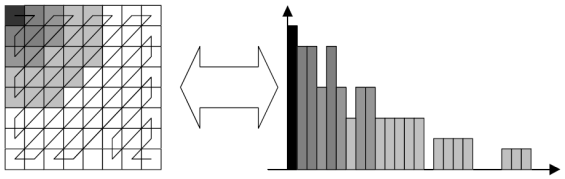

Figure A.3. Zig-zag scanning.

A P frame (predicted frame) is encoded relative to the past reference frame.

The reference frame is the closest preceding P or I frame. The encoder scans the reference frame and the P frame, looking for macroblock-size areas of the picture that appear similar. If the video contains moving objects, the encoder detects this. For areas of the image which have not changed between frames, macroblocks are skipped. Skipped macroblocks do not consume any data in the video stream. The decoder simply copies the macroblock from the previous reference frame. For areas that have changed slightly compared with the reference, the encoder takes the pixel difference and encodes this using DCT and quantization techniques. For areas where the encoder can detect the movement of an object from one macroblock position to another, it encodes a motion vector and difference information. The motion vector tells the decoder how far and in what direction the macroblock has moved. If the encoder cannot find a similar macroblock in the reference frame, the macroblock is encoded as if it belonged to an I frame [46]. A B frame is encoded relative to the past reference frame, the future reference frame, or both. The future reference frame is the closest following reference frame (I or P).

A group of pictures (Figure A.4) is a sequence of frames from one I frame to, but not including, the next, e.g., *IBBPBB*. Note that different videos may have different GOP sequences.

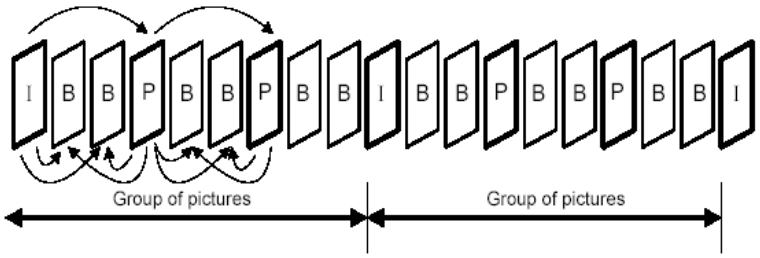

Figure A.4. Example of GOP structure.

A.3 Scalable video coding

MPEG-2 has methods to support multiple layer video coding, i.e. temporal scalability, spatial scalability, signal-to-noise ratio (SNR) scalability and data partitioning [78].

A scalable video coding scheme describes the encoding of video frames into multiple layers, including a base layer of relatively low-quality video and several enhancement layers that contain increasingly more video data to enhance the base layer and thus give rise to video of increasingly higher quality.

A.3.1 Temporal scalability

Temporal scalability is achieved by using bidirectionally-predicted pictures (B frames) to create enhancement layer(s) while storing all I frames and P frames in the base layer. B pictures are not used as reference pictures for the prediction of any other pictures. This property allows B pictures to be discarded if necessary, without impacting the visual picture quality of the future pictures.

A.3.2 SNR Scalability

The other basic method for achieving scalability is by SNR enhancement. SNR scalability uses quantization residues (the difference between actual DCT values and the values that are obtained after quantization / dequantization steps) to encode the enhancement layer. The extra data serves to increase the signal-to-noise ratio of the video picture, hence the term SNR scalability. According to the MPEG-2 standard, the reference for the predicted frame in the base layer consists of both EL and BL frames.

A.3.3 Spatial scalability

Spatial scalability is achieved by encoding a lower-resolution base layer, where an enhancement layer represents the coding loss between an up-sampled version of the reconstructed base layer picture and a the original picture.

A.3.4 Data partitioning

Data partitioning and SNR scalability are very similar. The difference is that the data partitioning approach breaks the block of quantized DCT coefficients into two parts. The first part contains the more critical lower frequency coefficients and side information (such as motion vectors). The second part carries higher frequency data.

A.4 Evaluation of the quality of video compression

The advent of digital video systems has exposed the limitations of the techniques traditionally used for video quality measurement and forced the designers of compression algorithms to resort to subjective viewing tests in order to obtain reliable ratings for the quality of compressed images or video [79]. However, these tests are complex and time-consuming. In their search for faster alternatives, researchers have turned to simple error measurement such as mean squared error(MSE) or peak signal-to-noise ratio(PSNR). These simple error measures operate on a pixel-by-pixel basis and look at the quality of single pictures, ignoring any interdependencies between pictures.

The MSE is the cumulative squared error between the compressed and the original picture, whereas PSNR is a measure of the peak error. The mathematical formulae for the two are

$$MSE = \frac{1}{M \cdot N} \sum_{y=1}^{M} \sum_{x=1}^{N} \left[I(x,y) - I^{'}(x,y) \right]^2 \tag{A.1}$$

$$PSNR = 20 \cdot \log_{10} \frac{255}{\sqrt{MSE}}, \tag{A.2}$$

where $I(x,y)$ is a pixel of the original picture, $I^{'}(x,y)$ is the approximated version (which is actually the decompressed picture) and M , N are the dimensions of the pictures. A lower value for MSE means less error and, as is apparent from the inverse relation between the MSE and $PSNR$, this translates to a high value of $PSNR$.

The peak signal-to-noise ratio is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Because many signals have a very wide dynamic range, PSNR is expressed in terms of the logarithmic decibel scale.

A.5 MPEG-2 traffic modelling

Modeling of video traffic involves the development of a mathematical model to accurately characterize the statistical properties of the frame size of MPEG-encoded video. Of particular interest in video traffic modeling is the frame size distribution. The frame size depends largely on the bit-rate control of the encoding algorithm.

Using the MPEG-2 encoding algorithm it is possible to follow one of the three following approaches in respect of bit rate:

- constant bit rate, CBR, where the number of bits spent in each GOP is controlled to a mean value. This approach leads to variable picture quality or distortion.
- variable bit rate, VBR, which involves controlling the distortion. This approach enables the production of video sequences that exhibit a constant quality level throughout their duration.
- CBR, always using the highest bit rate needed to satisfy a minimum level of distortion at all times.

CBR is common in digital television broadcasting, with the mean bit rate set to a value where the distortion in the scenes that are most difficult to encode is only barely noticeable. VBR is often used in relation with storage applications. When using the VBR mode, it is usual to obtain peak-to-average ratios of about 3:1 for the bit rate. The peak bit rate is used for high-detail and complex motion scenes.

It has been shown in a number of research projects (see [80, 63, 64, 81]) that the characterization of MPEG-2 traffic depends on the type of video, capture rate, amount of action, etc. The distribution-modeling of the traffic shows that the distribution of frame sizes in a video can be represented via Normal, Gamma or Lognormal distributions. The type of distribution that will give the best results depends on the video characteristics.

The work in [63] demonstrated that the distribution of video frame sizes could be approximated as a Normal distribution. It is shown in [81] that the Normal distribution matches very well with distributions of all three frame types in MPEG-2 video.

With respect to VBR traffic, in [64] the Gamma distribution is considered to be the best fit, as Normal distribution does not accommodate the heavy tail that results from large *I* frames. However, even with VBR traffic an encoder manages its average bit rate over a GOP. According to [81], the average bit rate over the GOP or the sliding window of half a second is bounded within +/- 8% with a probability greater than 99%.

B

Networking basics

Networking is the exchange of data such as text, audio and video between remote parties via a transmission medium (e.g. cable). The data exchange is only possible if the parties involved are part of the same communication system. Figure B.1 shows the basic components of a data communication system (DCS).

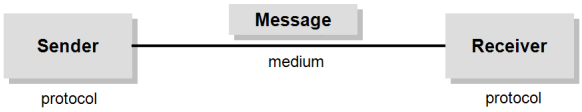

Figure B.1. Data communication system.

A data communication system consists of five components [82]: the message, the sender that sends the message, the receiver that receives the message, the medium that is the physical means by which the message travels, and a protocol that is a set of rules that govern communication. A link is a communication path that transfers data. The sender and the receiver can be a computer, PDA, telephone, television, etc. Two or more devices connected to each other with a communication link form a network. There are two types of connections: point-to-point, which provides a dedicated link between two devices, and multipoint, which connects more than two devices on a single link.

Transmission media used in telecommunications can be divided into two categories: guided (e.g. twisted-pair cable, coaxial cable and fiber-optic cable) and unguided (e.g. electromagnetic waves) [82].

Guided media provide a conduit from one device to another. A signal travelling along any of the guided media is directed and contained by the physical limits of the medium. Unguided media transport is often referred to as wireless communication.

B.1 OSI Model

The International Standards Organization established a framework for standardizing communications systems, called Open Systems Interconnection Reference Model (OSI Model). OSI is a layered abstract description for data communications and networking. The model defines the communications process as an ordered set of seven layers, with specific functions isolated to and associated with each layer [83]. Each layer hides low layer processes, effectively isolating them from higher layer functions. In this way, each layer performs a set of functions that are necessary to provide a set of services to the layer above it.

The model is composed of seven layers:

Layer 1: Physical Layer This layer defines all the electrical and physical specifications for devices. It implements the functions required to transmit data over a physical medium

Layer 2: Data Link Layer This layer is responsible for the transfer of data between devices on the same link and for the correction of errors that may occur in the physical layer.

Layer 3: Network Layer This layer transfers data from a source to a destination across one or more networks.

Layer 4: Transport Layer This layer provides transparent transfer of data between end users. It keeps track of the packets, recognizes relationships between the packets and retransmits any of the packets that fail.

Layer 5: Session Layer This layer establishes, maintains and synchronizes the interaction between end-user application processes.

Layer 6: Presentation Layer This layer handles syntactical differences in a data representation. It is also intended for data encryption, decryption and compression.

Layer 7: Application Layer This layer enables the user to access the network.

Layer isolation allows the characteristics of a given layer to change without impacting on the remainder of the model, provided that the supporting services remain the same [83].

B.2 Network types

Networks are subdivided into categories depending on size, ownership, distance covered and physical architecture [82]. Three primary categories are: Local Area Network (LAN), Metropolian Area Network (MAN) and Wide Area Network (WAN). A LAN is a group of devices connected together, usually within the same office or building. A MAN is a larger network that extends over several buildings in the same city. A WAN provides long-distance transmission that is not restricted to a geographical location. A WAN connects several LANs.

B.3 Local Area Network

B.3.1 Topologies

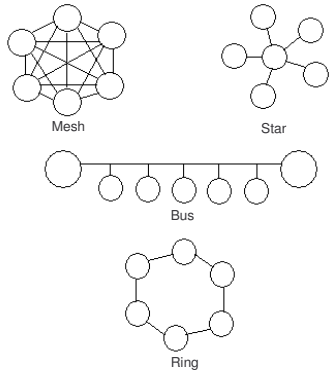

Figure B.2. LAN topology types: mesh, star, bus, ring.

The physical topology describes the way in which a network is laid out physically. There are four basic topologies: *mesh*, *star*, *bus* and *ring* (see Figure B.2). With a *mesh* topology devices are connected to many redundant interconnections – each device is connected to every other device. In a *star* topology all devices are connected to a central controller, called a hub. Devices communicate across the network by passing data through the hub. In a *bus* topology, which is the only one that uses multipoint connections, all devices are connected to a central cable, called the bus or backbone. With a *ring* topology devices are connected to one another in a closed loop, so that each device is connected directly to two other devices. A signal is passed along the *ring* in one direction, from device to device, until it reaches its destination.

B.3.2 Equipment

Digital signals are affected by noise, attenuation and other impairments which limit the distance over which they can be transmitted before the signal becomes unrec-

ognizable [83]. A repeater allows the connection of LAN segments, extending the network beyond the maximum length of a single segment. A multi-port repeater is known as a hub.

A hub is the center of the star topology. It connects individual devices. The connection between hubs make it possible to expand the number of devices connected to the LAN. Both hub and repeater work at the physical layer of the OSI model.

Bridges are similar to repeaters in that they are used to connect two LAN segments. However, unlike a repeater which simply duplicates all traffic on one segment to the other, a bridge reads addresses and determines whether or not to forward a packet on to another segment. A bridge at the data link layer works as well as a switch. A switch is a multi-port bridge, where each port of the bridge decides whether to forward data packets to the attached network.

Routers operate at the network layer and have two primary functions – to determine the 'best path' and to share details of routes with other routers. A router keeps track of the routes to networks in a routing table. While static routing uses data entered manually by a network administrator, dynamic routing adjusts automatically to changes in network topology by obtaining information from other routers. A router can interconnect different network types, changing packet size and format to match the requirements of the destination network.

A gateway allows different networks to communicate by offering a translation service at all levels of the OSI model.

B.4 Wireless LAN

Much like a traditional wired LAN, a wireless LAN (WLAN) is a grouping of devices that share a common communications backbone. A WLAN allows users to connect wirelessly to the LAN via radio transmission. Until 1997 wireless LANs represented proprietary technology. In 1997 the Institute of Electrical and Electronics Engineers (IEEE) published its 802.11 standard for wireless LANs which allows vendors to develop products that are interoperable [83]. The IEEE 802.11 standard governs the two lower layers of the ISO Reference Model, the physical layer and the data link layer.

The initial specification used the 2.4 GHz frequency and supported a maximum data rate of 1 to 2 Mbps. The 802.11b specification, introduced in 1999, increased the performance to 11 Mbps in the 2.4 GHz range while the 802.11a specification utilized the 5 GHz range and supported up to 54 Mbps. Unfortunately, the two new specifications were incompatible because they used different frequencies. This incompatibility has been overcome with the new standard known as 802.11g. 802.11g supports up to 54 Mbps and is interoperable with 802.11b products.

The 802.11 specification defines two types of operational modes: ad hoc (peer-to-peer) mode and infrastructure mode. In ad hoc mode, the networked devices communicate directly with one another. In infrastructure mode, all wireless devices communicate with an access point. The access point acts as a base station in an 802.11 network and all communications from all of the wireless clients go through the access point. The access point also provides the connection from the wireless radio frequency world to the wired LAN world.

The level of performance of a WLAN is dependent on a number of environmental factors, such as:

Distance between WLAN devices Typically, wireless LAN performs best in line-of-sight or open area environments.

Radio frequency interference The 802.11b standard uses the unlicensed radio spectrum that is commonly shared by a variety of consumer devices: baby monitors and cameras, 2.4 GHz cordless phones, microwave ovens, and Bluetooth-enabled devices like cellular phones or personal digital assistants. These devices transmit in the 2.4 GHz range and can impair WLAN performance.

Signal propagation The materials used in a building have a dramatic impact on the quality of the signal obtained with an 802.11 wireless network. Wood, metal and other building materials have a direct impact on signal propagation and absorption. Other factors include:

- Multi-path interference that occurs when signal strength and timing are altered due to the signal being reflected off walls, filing cabinets, beams and other objects, causing a device to receive two or more identical signals.
- Fading, i.e. the reduced amplitude of a signal caused by the signal passing through radio-transparent objects such as walls.
- Dead zones, which are locations that are never reached by the radio signals due to reflections, obstructions or other environmental factors.

B.5 TCP/IP Protocols

TCP/IP protocols, also known more formally as the Internet Protocol Suite, facilitate communications across interconnected, heterogeneous computer networks. They are a combination of different protocols, which are normally organized into four layers:

1. The link layer, which handles all the hardware details to provide data transmission for the network layer. Network layer protocols can be supported by various link layer technologies, such as Ethernet or Wireless LAN.

2. The network layer, which handles routing of packets across the networks. It includes the Internet Protocol (IP) – the core of the TCP/IP protocol stack.
3. The transport layer, which provides data transport for the application layer, including the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP).
4. The application layer.

In the rest of this thesis only transport and application layers will be addressed. Transport layer protocols are discussed in the following sections.

B.5.1 UDP

The User Datagram Protocol is a connectionless service, which means that the packets are transported without any form of control. UDP does not provide reliability and ordering guarantees, so the data may arrive out of order or get lost without notice. Because UDP lacks reliability, UDP applications must generally be willing to accept some loss, errors or duplication. Most often, UDP applications do not require reliability mechanisms and may even be hindered by them. Streaming media, real-time multiplayer games [84, 85] and voice-over IP (VoIP [86, 87]) are examples of applications that often use UDP.

B.5.2 TCP

The Transmission Control Protocol provides a service for applications that require connection setup, error detection and automatic retransmission.

Unlike UDP, which can start sending packets immediately, TCP requires a connection establishment before it can send data and a connection termination when it has finished sending data.

Each unit of data carried by TCP is referred to as a segment. Segments are created by TCP subdividing the stream of data passed down by application layer protocols. This segment identification process enables a receiver, if required, to reassemble data segments into their correct order [83]. The receiver sends back an acknowledgement for data packets which have been received successfully; a timer at the sending TCP will cause a timeout if an acknowledgement is not received within a reasonable time, referred to as round-trip time (or RTT), and the data (which has presumably been lost) will then be re-transmitted.

TCP has been optimized for wired networks. Packet loss is considered to be the result of congestion. TCP uses a congestion window at the sender side for the purpose of congestion avoidance. The congestion window indicates the maximum amount of data that can be sent out on a connection without being acknowledged. TCP detects congestion when it fails to receive an acknowledgement for a packet within the estimated timeout. In such a situation, it decreases the congestion window. It increases the congestion window in other cases. However, wireless links

are known to experience sporadic losses, for example due to fading, which cannot be considered congestion. Erroneous back-off of the window size due to wireless packet loss is followed by a congestion avoidance phase with a conservative decrease in window size which causes the link to be under-utilized.

B.5.3 RTP

The Real-time Transport Protocol (RTP) is the protocol designed to handle real-time traffic. RTP comes between the application program and UDP since it has no delivery mechanism and, therefore, reuses UDP. Applications using RTP are less sensitive to packet loss, but typically very sensitive to delays.

The protocol contributions are: payload-type identification, sequence numbering, time stamping and delivery monitoring. The protocol itself does not provide mechanisms to ensure timely delivery. In addition, out-of-order delivery is still possible, and flow and congestion control are not supported directly. However, the protocol delivers the necessary data to the application to make sure it can put the packets received in the correct order.

A special protocol, Real-time Transport Control Protocol (RTCP) , has been designed for the purpose of providing information about reception quality from the receiver to the sender.

B.6 Quality of Service

The goal of QoS is to provide preferential delivery service for the applications that need it by ensuring sufficient bandwidth, controlling latency and jitter, and reducing data loss.

The network characteristics are that are useful for applications (e.g. ‘throughput’, ‘delay’, ‘residual error rate’, or ‘priority’) summarized in a set of so-called network performance parameters [88]. Applications have to specify their requirements with respect to network performance parameters to request a special communication service. QoS provision mechanisms are present to provide applications with service within the specified parameters.

Two basic mechanism that are used for QoS provision are admission control and traffic control. The admission control determines which applications are allowed to use the network. These mechanisms specify how, when, and by whom network resources can be used. The traffic control regulates data flows by classifying, scheduling and marking packets on the basis of priority and by shaping traffic (smoothing bursts of traffic by limiting the rate of flow).

C

Enhancement Layer input in SNR video encoder

Originally proposed non-compliant MPEG-2 encoder that is based on SNR scalability with EL formed on the difference between original and encoded DCT coefficients is shown in Figure C.1.

The scheme can be modified to use the difference between pixel values, based on a property of DCT:

$$DCT(f+g) = DCT(f) + DCT(g). \tag{C.1}$$

This means that a video can be separated into layers in the spatial domain as well (Figure C.2).

The upper rectangle of Figure C.2 presents EL decoder. The functionality blocks inside the rectangle repeats the exact scheme of a simple MPEG-2 non-scalable encoder with no motion prediction. The input data, however, contains data that is transformed by the BL encoder.

The scheme shown in Figure C.3 produce exactly the same results as the one above. The input data, however, is formed based on the difference between the original pixel values and the values encoded in BL stream. The equality of the results can be proven as follows. Imaging the original pixel has a value v. After motion prediction the values is modified to $p = v - r$, where r is a value of

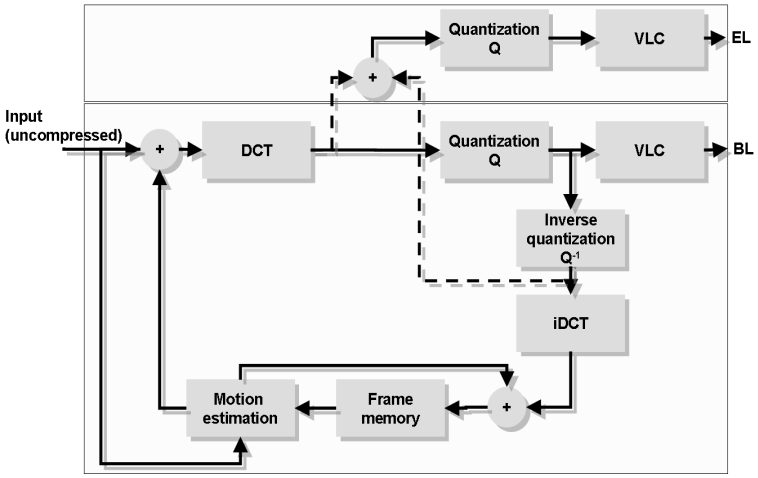

Figure C.1. Non-compliant MPEG-2 encoder based on SNR scalability with EL data based on DCT coefficients.

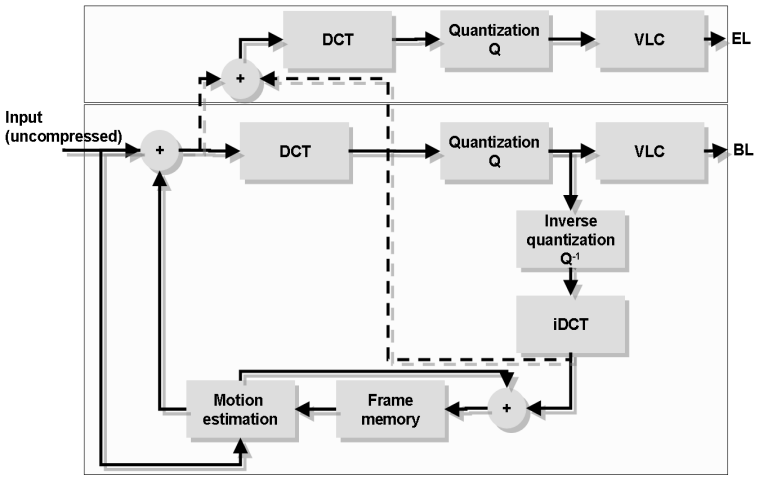

Figure C.2. Non-compliant MPEG-2 encoder based on SNR scalability with EL data based on pixel values after performing motion prediction.

the correspondent pixel in the reference frame (if there is no reference frame, r is zero). Due to the lossy nature of the encoding, the value p of the pixel transforms to $\tilde{p}$ after iDCT. To calculate the full value of the pixel, r is added back to $\tilde{p}$, so the resulting value $\tilde{v} = \tilde{v} + r$. The encoder in Figure C.2 calculates the difference $e = p - \tilde{p}$ and stores it in EL. Given that $p = v - r$ and $\tilde{p} = \tilde{v} - r$, the pixel difference can be calculated as $e = v - \tilde{v}$.

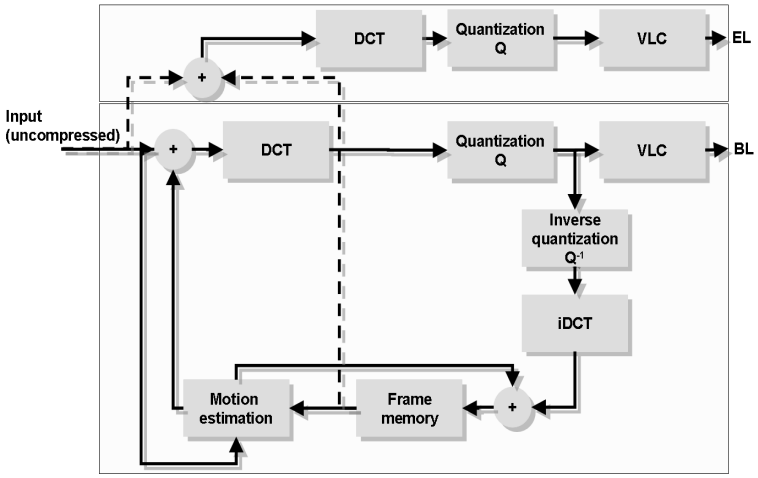

Figure C.3. Non-compliant MPEG-2 encoder based on SNR scalability with EL data based on original pixel values.

D

Rate-Distortion characteristics of the scalable video transcoder

Exponential model is a popular technique to model rate and distortion characteristics for video encoders and transcoders (in the Appendix we refer to encoders and transcoder as *video producers*).

The exponential model describes relation between bit rate of the video data R and distortion of the data D as

$$D = \alpha + \beta \cdot \log\left(\frac{1}{R}\right) \tag{D.1}$$

The value of α and β coefficients depend on implementation of the video producer and on characteristics of video data being processed. Thus, the coefficients need to be recalculated for every new video. The output of the video producer, aside from the video itself, contains distortion value (MSE) that is calculated on 'per frame basis'. A set of MSE values together with the corresponding bit rate values of the output video is used to calculate α and β at the run-time.

The values of α and β are calculated using method of least squares linear regression. With this method we try to find such α and β that the sum of the squares of the differences (also referred to as *square error*) to all data points has the smallest possible value. Linear least square error regression is used, since we deal with a function that is linear in the parameters (even though nonlinear in the variables).

From Equation D.1, the square error for the i^{th} point is

$$\begin{aligned} S_i^{err} &= \left(D_i - \alpha - \beta \cdot \log\left(\frac{1}{R_i}\right)\right)^2 \\ &= \alpha^2 - 2 \cdot D_i \cdot \alpha + 2 \cdot \alpha \cdot \beta \cdot \log\left(\frac{1}{R_i}\right) \\ &\quad - 2 \cdot D_i \cdot \beta \cdot \log\left(\frac{1}{R_i}\right) + \beta^2 \cdot \log^2\left(\frac{1}{R_i}\right) + D_i^2. \end{aligned} \tag{D.2}$$

The square error for all N data points is

$$\begin{aligned} S^{err} &= \sum_{i=1}^{N} \left(D_i - \alpha - \beta \cdot \log\left(\frac{1}{R_i}\right)\right)^2 \\ &= \sum_{i=1}^{N} (\alpha^2 - 2 \cdot D_i \cdot \alpha + 2 \cdot \alpha \cdot \beta \cdot \log\left(\frac{1}{R_i}\right) \\ &\quad - 2 \cdot D_i \cdot \beta \cdot \log\left(\frac{1}{R_i}\right) + \beta^2 \cdot \log^2\left(\frac{1}{R_i}\right) + D_i^2). \end{aligned} \tag{D.3}$$

In linear least squares regression, we seek the values of α and β that minimize S^{err}. The most direct approach is to take the partial derivatives of the function with respect to the coefficients:

$$\begin{aligned} \frac{\delta S^{err}}{\delta \alpha} &= \sum_{i=1}^{N} \left[2 \cdot \alpha - 2 \cdot D_i + 2 \cdot \beta \cdot \log\left(\frac{1}{R_i}\right)\right], \\ \frac{\delta S^{err}}{\delta \beta} &= \sum_{i=1}^{N} \left[2 \cdot \alpha \cdot \log\left(\frac{1}{R_i}\right) - 2 \cdot D_i \cdot \log\left(\frac{1}{R_i}\right) + 2 \cdot \beta \cdot \log^2\left(\frac{1}{R_i}\right)\right]. \end{aligned} \tag{D.4}$$

Setting these partial derivatives to zero yields simultaneous linear equations for α and β, the normal equations for simple regression:

$$\begin{aligned} \sum_{i=1}^{N} \left[\alpha - D_i + \beta \cdot \log\left(\frac{1}{R_i}\right)\right] &= 0, \\ \sum_{i=1}^{N} \left[\alpha \cdot \log\left(\frac{1}{R_i}\right) - D_i \cdot \log\left(\frac{1}{R_i}\right) + \beta \cdot \log^2\left(\frac{1}{R_i}\right)\right] &= 0. \end{aligned} \tag{D.5}$$

From Equation D.5 α and β are calculated as

$$\alpha = \frac{\sum_{i=1}^{N}\left(\log^2\left(\frac{1}{R_i}\right)\right)\cdot\sum_{i=1}^{N}(Q_i) - \sum_{i=1}^{N}\left(Q_i\cdot\log\left(\frac{1}{R_i}\right)\right)\cdot\sum_{i=1}^{N}\left(\log\left(\frac{1}{R_i}\right)\right)}{N\cdot\sum_{i=1}^{N}\left(\log^2\left(\frac{1}{R_i}\right)\right) - \sum_{i=1}^{N}\left(\log\left(\frac{1}{R_i}\right)\right)\cdot\sum_{i=1}^{N}\left(\log\left(\frac{1}{R_i}\right)\right)}$$

$$\beta = \frac{N\cdot\sum_{i=1}^{N}\left(Q_i\cdot\log\left(\frac{1}{R_i}\right)\right) - \sum_{i=1}^{N}\left(\log\left(\frac{1}{R_i}\right)\right)\cdot\sum_{i=1}^{N}(Q_i)\,.}{N\cdot\sum_{i=1}^{N}\left(\log^2\left(\frac{1}{R_i}\right)\right) - \sum_{i=1}^{N}\left(\log\left(\frac{1}{R_i}\right)\right)\cdot\sum_{i=1}^{N}\left(\log\left(\frac{1}{R_i}\right)\right)}. \tag{D.6}$$

Equation D.6 contains four similar elements:

$$Sum_1 = \sum_{i=1}^{N}\left(\log\left(\frac{1}{R_i}\right)\right), \tag{D.7}$$

$$Sum_2 = \sum_{i=1}^{N}\left(\log^2\left(\frac{1}{R_i}\right)\right), \tag{D.8}$$

$$Sum_3 = \sum_{i=1}^{N}(Q_i), \tag{D.9}$$

$$Sum_4 = \sum_{i=1}^{N}\left(Q_i\cdot\log\left(\frac{1}{R_i}\right)\right). \tag{D.10}$$

Recalculation of α and β is, therefore, very easy at the run-time. The values of Sum_1, Sum_2, Sum_3 and Sum_4 are get updated with every frame, adding new pair of Q_i and R_i. The amount of calculations consists of 4 summations, 4 subtractions, 10 multiplications, 1 division, and 3 calculations of natural logarithm.

Bibliography

[1]MRG. *Home gateway report: Worldwide multi-carrier digital settop and services analysis and forecast 2003-2006. Technical report*, Multimedia Research Group, Inc. (2003).

[2]J. Apostolopoulos, W. tian Tan, and S. Wee. *Handbook of Video Databases: Design and Applications*, chapter Video Streaming: Concepts, Algorithms, and Systems (CRCPress, 2003).

[3]M. Margaritidis and G. Polyzos. *Application-assisted adaptation of realtime streams over wireless links*. In *Proceedings of the Second Annual UCSD Conference on Wireless Communications* (San Diego, CA, 1999).

[4]J. Bolliger and T. Gross. *A framework-based approach to the development of network-aware applications. IEEE Trans. Softw. Eng.*, volume 24(5), (1998) pp. 376–390.

[5]P. Amon and J.Pandel. *Evaluation of adaptive and reliable video transmission technologies*. In *Proc. of the 13th Packet Video Workshop* (Nantes, France, 2003).

[6]B. G. Haskell, A. Puri, and A. N. Netravali. *Digital Video: An introduction to MPEG-2* (Chapman & Hall, Ltd., London, UK, UK, 1996).

[7]M. Kouras and A. Asif. *Noncausal predictive video codec offering hierarchical qos*. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, volume 3, pp. 737–740 (2004).

[8]K. Rose and S. L. Regunathan. *Toward optimality in scalable predictive coding. IEEE Transactions on Image Processing*, volume 10(7), (2001) pp. 965–976.

[9]M. Johanson. *A scalable video compression algorithm for real-time internet applications*. In *Proceedings of the 4th EURASIP Conference on Video / Image Processing and Multimedia Communications* (2003).

[10]M. Domanski and S. Mackowiak. *Modified mpeg-2 video coders with efficient multi-layer scalability.* In *ICIP (2)*, pp. 1033–1036 (2001).

[11]H. Wang and S. Venkatesan. *Adaptive video transmission over a single wireless link*. In *Proc. 10th Int'l Conf. Distributed Multimedia System*, (San Francisco, CA, US, 2004).

[12]A. Asif, U. Nguyen, X. Guohua, and S. Bin. *Streaming video with bandwidth adaptation and error concealment for lowbit rate live wireless applications*. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, volume 2, pp. 313–316 (2005).

[13]T. Tian, A. H. Li, J. Wen, and J. D. Villasenor. *Priority dropping in network transmission of scalable video.* In *ICIP* (2000).

[14]Z. Bing and M. Atiquzzaman. *Tsfd: two stage frame dropping for scalable video transmission over data networks.* In *High Performance Switching and Routing, 2001 IEEE Workshop on*, pp. 43–47 (2001).

[15]J. Cai, X. Li, and C. W. Chen. *Layered unequal loss protection with pre-interleaving for fast progressive image transmission over packet-loss channels*. *ACM Trans. Multimedia Comput. Commun. Appl.*, volume 1(4), (2005) pp. 338–353.

[16]M. G. Podolsky, S. McCanne, and M. Vetterli. *Soft arq for layered streaming media*. *J. VLSI Signal Process. Syst.*, volume 27(1-2), (2001) pp. 81–97.

[17]Z. Miao and A. Ortega. *Optimal scheduling for streaming of scalable media*. In *Asilomar Conference on Signals, Systems, and Computers* (2000).

[18]P. de Cuetos and K. W. Ross. *Adaptive rate control for streaming stored rine-grained scalable video*. In *NOSSDAV '02: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pp. 3–12 (ACM Press, New York, NY, USA, 2002).

[19]H. Sun, W. Kwok, and J. Zdepski. *Architectures for mpeg compressed bitstream scaling*. In *ICIP '95: Proceedings of the 1995 International Conference on Image Processing (Vol. 1)-Volume 1*, p. 81 (IEEE Computer Society, Washington, DC, USA, 1995).

[20]P. Assuncao and M. Ghanbari. *A frequency-domain video transcoder for dynamic bit-rate reductionof mpeg-2 bit streams. IEEE Transactions on Circuits and Systems for Video Technology*, volume 8(8), (1998) pp. 953 – 967.

[21]K.-D. Seo, S.-H. Lee, J.-K. Kim, and J.-S. Koh. *Rate control algorithm for fast bit-rate conversion transcoding. IEEE Transactions on Consumer Electronics*, volume 46(4), (2000) pp. 1128 – 1136.

[22]W. Lie and Y. Chen. *Dynamic rate control for mpeg-2 bit stream transcoding*. In *ICIP01*, pp. 477–480 (2001).

[23]L. Lu, S. Xiao, J. Kouloheris, and C. A. Gonzales. *Efficient and low-cost video transcoding.* In *VCIP*, edited by C. C. J. Kuo, volume 4671 of *Proceedings of SPIE*, pp. 154–163 (SPIE, 2002).

[24]J. I. Khan and Q. Gu. *Network aware video transcoding for symbiotic rate adaptation on interactive transport.* In *NCA*, pp. 201–213 (IEEE Computer Society, 2001).

[25]A. Vetro, H. Sun, and Y. Wang. *Object-based transcoding for adaptable*

video content delivery (2001).

[26]Z. Lei and N. D. Georganas. *Rate adaptation transcoding for precoded video streams*. In *MULTIMEDIA '02: Proceedings of the tenth ACM international conference on Multimedia*, pp. 127–136 (ACM Press, New York, NY, USA, 2002).

[27]Z. Lei and N. Georganas. *Rate adaptation transcoding for video streaming over wireless channels*. In *In Proceedings of IEEE ICME 2003.* (2003).

[28]J. R. Taal and I. L. Lagendijk. *Asymmetric multiple description coding using layered coding and lateral error correction. Technical report* (2005).

[29]I. F. Díaz, D. Epema, and J. de Jongh. *Multipath routing and multiple description coding in ad-hoc networks: a simulation study*. In *PE-WASUN '04: Proceedings of the 1st ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pp. 46–51 (ACM Press, New York, NY, USA, 2004).

[30]V. Stankovic, R. Hamzaoui, and Z. Xiong. *Robust layered multiple description coding of scalable media data for multicast. IEEE Signal Processing Letters*, volume 12(2), (2005) pp. 154–157.

[31]J. R. Taal and R. I. L. Lagendijk. *Scalable multiple description video for fair streaming to many clients. Technical report* (2005).

[32]C. C. Wust, L. Steffens, W. F. Verhaegh, R. J. Bril, and C. Hentschel. *QoS control strategies for high-quality video processing. Real-Time Syst.*, volume 30(1-2), (2005) pp. 7–29.

[33]T. Lan, Y. Chen, and Z. Zhong. *Mpeg2 decoding complexity regulation for a media processor*. In *Multimedia Signal Processing, 2001 IEEE Fourth Workshop on*, pp. 193–198 (2001).

[34]C. Wüst. *Intelligent Control for Scalable Video Processing*. Ph.D. thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands (2006).

[35]D. McNamee, C. Krasic, K. Li, A. Goel, E. Walthinsen, D. Steere, and J. Walpole. *Control challenges in multi-level adaptive video streaming*. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 3, pp. 2228–2233 (2000).

[36]J. Walpole, R. Koster, S. Cen, C. Cowan, D. Maier, D. McNamee, C. Pu, D. Steere, and L. Yu. *A player for adaptive mpeg video streaming over the internet*. In *Proceedings of the SPIE Applied Imagery Pattern Recognition Workshop* (1997).

[37]M. Zink, O. Kunzel, J. Schmitt, and R. Steinmetz. *Subjective impression of variations in layer encoded videos*. In *Eleventh International Workshop on Quality of Service (IWQoS 2003), Monterey, CA, USA* (Springer Verlag, 2003).

[38]D. Jarnikov, J. Lukkien, and P. van der Stok. *Adaptable video streaming*

over wireless networks. In *ICCCN2005* (2005).

[39]D. Jarnikov, J. Lukkien, and P. van der Stok. *A framework for video streaming to resource-constrained terminals*. In *EUC2005* (2005).

[40]D. Jarnikov, P. van der Stok, and J. Lukkien. *Influence of network awareness on perceived video quality*. In *RM4NES workshop* (2005).

[41]J. Kang, H. de Groot, P. van der Stok, D. Jarnikov, I. Nitescu, and F. Ogg. *Dynamic and Robust Streaming in and between Connected Consumer-Electronic Devices*, chapter Robust video streaming over wireless in-home networks, pp. 193–212 (Kluwer, 2005).

[42]R. Haakma, D. Jarnikov, and P. van der Stok. *Dynamic and Robust Streaming in and between Connected Consumer-Electronic Devices*, chapter Perceived quality of wirelessly transported videos, pp. 213–239 (Kluwer, 2005).

[43]D. Jarnikov, P. van der Stok, and C. C. Wüst. *Predictive control of video quality under fluctuating bandwidth conditions.* In *ICME*, pp. 1051–1054 (2004).

[44]D. Jarnikov, P. van der Stok, and J. Lukkien. *Wireless streaming based on a scalability scheme using legacy mpeg-2 decoders.* In *IMSA*, pp. 371–376 (2005).

[45]P. van der Stok, D. Jarnikov, S. Kozlov, M. van Hartskamp, and J. Lukkien. *Hierarchical resource allocation for robust in-home video streaming*. *J. Syst. Softw.*, volume 80(7).

[46]ISO/IEC. *'MPEG-2' generic coding of moving pictures and associated audio information. ISO/IEC 13818*.

[47]S. Aramvith and M. Sun. *Handbook of Image and Video Processing*, chapter MPEG-1 and MPEG-2 video standards, pp. 597–610 (Academic Publishers, 2000).

[48]D. Jarnikov. *Towards balancing network and terminal resources to improve video quality*. *Sai technical report*, Eindhoven University of Technology (2003).

[49]M. Ghanbari. *Two-layer coding of video signals for vbr networks*. *IEEE Journal on Selected Areas in Communications*, volume 7(5), (1989) pp. 771–781.

[50]T. Halbach and T. R. Fischer. *Snr scalability by transform coefficient refinement for block-based video coding.* In *VCIP*, edited by T. Ebrahimi and T. Sikora, volume 5150 of *Proceedings of SPIE*, pp. 135–140 (SPIE, 2003).

[51]J. F. Arnold, M. R. Frater, and Y. Wang. *Efficient drift-free signal-to-noise ratio scalability. IEEE Trans. Circuits Syst. Video Techn.*, volume 10(1), (2000) pp. 70–82.

[52]C. Brouwers. *A real-time SNR scalable transcoder for MPEG-2 video streams*. Master's thesis, TECHNISCHE UNIVERSITEIT EINDHOVEN

(2006).

[53]The Editors of IEEE 802.11. *IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications* (1997).

[54]C. E. Palazzi, G. Pau, M. Roccetti, and M. Gerla. *In-home online entertainment: Analyzing the impact of the wireless mac-transport protocols interference*. In *Proceedings of IEEE International Conference on Wireless Networks, Communications and Mobile Computing (WIRELESSCOM 2005)* (Maui, HI, USA, 2005).

[55]S. Xu and T. Saadawi. *Does the ieee 802.11 mac protocol work well in multihop wireless ad hoc networks? Communications Magazine, IEEE*, volume 39(6), (2001) pp. 130–137.

[56]S. Xu and T. Saadawi. *Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks. Comput. Networks*, volume 38(4), (2002) pp. 531–548.

[57]E. Altman and T. Jim.

[58]C. H. Nam, S. C. Liew, and C. P. Fu. *An experimental study of arq protocol in 802.11b wireless lan*. In *of Wireless Personal Multimedia Communications (WPMC'02)* (2002).

[59]G. Xylomenos and G. C. Polyzos. *Tcp and udp performance over a wireless lan.* In *INFOCOM*, pp. 439–446 (1999).

[60]S. Kozlov, P. van der Stok, and J. Lukkien. *Adaptive scheduling of mpeg video frames during real-time wireless video streaming.* In *Proc. IEEE WoWMoM2005*, pp. 460–462 (2005).

[61]A. Oppenheim and R. Schafer. *Discrete-Time Signal Processing.* (Prentice-Hall, 1989).

[62]M. R. Izquierdo and D. S. Reeves. *A survey of statistical source models for variable-bit-rate compressed video. Multimedia Syst.*, volume 7(3), (1999) pp. 199–213.

[63]P. Goyal and H. M. Vin. *Network algorithms and protocol for multimedia servers. Technical Report CS-TR-95-19*, Austin, TX, USA (1995).

[64]M. W. Garrett and W. Willinger. *Analysis, modeling and generation of self-similar vbr video traffic.* In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, pp. 269–280 (ACM Press, New York, NY, USA, 1994).

[65]W. Ding and B. Liu. *Rate quantization modeling for rate control of mpeg video coding and recording*. In *Proc. SPIE Vol. 2419, p. 139-150, Digital Video Compression: Algorithms and Technologies 1995, Arturo A. Rodriguez; Robert J. Safranek; Edward J. Delp; Eds.*, pp. 139–150 (1995).

[66]J.-J. Chen and H.-M. Hang. *A transform video coder source model and its application.* In *ICIP (2)*, pp. 967–971 (1994).

[67]J. Katto and M. Ohta. *Mathematical analysis of mpeg compression capability and its application to rate control.* In *ICIP*, pp. 2555–2558 (1995).

[68]Y. Huang, S. Chakraborty, and Y. Wang. *Using offline bitstream analysis for power-aware video decoding in portable devices.* In *ACM Multimedia*, pp. 299–302 (2005).

[69]S. Peng. *Complexity scalable video decoding via idct data pruning*. In *International Conference on Consumer Electronics*, p. 74 (2001).

[70]S. O. Mietens. *Complexity Scalable MPEG Encoding*. Ph.D. thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands (2004).

[71]M. Puterman. *Markov decision processes: discrete stochastic dynamic programming* (Wiley, New York, 1994).

[72]C. C. Wüst and W. F. J. Verhaegh. *Quality control for scalable media processing applications. J. of Scheduling*, volume 7(2), (2004) pp. 105–117.

[73]K. Fall and K. Varadhan. *The ns manual*. LBL, USC/ISI, and Xerox PARC (2005).

[74]S. Dutta, R. Jensen, and A. Rieckmann. *Viper: A multiprocessor soc for advanced set-top box and digital tv systems. IEEE Design and Test of Computers*, volume 18(5), (2001) pp. 21–31.

[75]S. Rathnam and G. Slavenburg. *An architectural overview of the programmable multimedia processor, tm-1. compcon*, volume 00, (1996) p. 319.

[76]K. R. Rao and P. Yip. *Discrete cosine transform: algorithms, advantages, applications* (Academic Press Professional, Inc., San Diego, CA, USA, 1990).

[77]S.W.Golomb. *Run-length encodings. IEEE Trans. Information Theory*, volume 12, (1966) pp. 399–401.

[78]Y. Wang, Y. quin Zhang, and J. Ostermann. *Video Processing and Communications* (Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001).

[79]ITU Telecom. Standardization Sector of ITU. *Methodology for the Subjective Assessment of the Quality of Television Pictures, Recommendation ITU-R BT.500-7* (1997).

[80]P. Bahl and I. Chlamtac. *Influence of available bandwidth on the statistical characterization of compressed video* (1996).

[81]S. Agrawal, C. F. Barry, V. Binnai, and L. G. Kazovsky. *Characterization, adaptive traffic shaping, and multiplexing of real-time MPEG II video*. In *Proc. SPIE Vol. 2915, p. 51-63, Video Techniques and Software for Full-Service Networks, Tzi-cker Chiueh; Andrew G. Tescher; Eds.*, edited by T.-C. Chiueh and A. G. Tescher, pp. 51–63 (1997).

[82]B. A. Forouzan. *Data Communications and Networking* (McGraw-Hill, Inc., New York, NY, USA, 2003).

[83]G. Held. *Understanding data communications: from fundamentals to networking (Third Edition)* (John Wiley & Sons, Inc., New York, NY, USA, 2001).

[84]M. R. Mine, J. Shochet, and R. Hughston. *Building a massively multiplayer game for the million: Disney's toontown online*. *Comput. Entertain.*, volume 1(1), (2003) pp. 15–15.

[85]J. Smed, T. Kaukoranta, and H. Hakonen. *Aspects of networking in multiplayer computer games*. *The Electronic Library*, volume 20(2), (2002) pp. 87–97.

[86]P. Mehta and S. Udani. *Voice over ip*. *IEEE Potentials*, volume 20(4), (2001) pp. 36–40.

[87]T. Kanter and C. Olrog. *Voip in applications for wireless access*. *IEEE Post-Conference Journal Publication of the 10th IEEE Workshop on Local and Metropolitan Area Networks*, pp. 122–125.

[88]ITU Telecom. Standardization Sector of ITU. *General Aspects of Quality of Service and Network Performance in Digital Networks, including ISDN, Recommendation ITU-T E.350* (1993).

Symbol Index

The numbers refer to the pages of first occurrence.

L_B	base layer	23
$L_{E,i}$	enhancement layer i ($1 \leq i \leq n_E$)	23
B^R	effective available bandwidth	24
$\Phi(f)$	priority of frame f	29
FB^j	frame number j in the base layer ($0 \leq j < n_F$)	29
FE_i^j	frame number j in enhancement layer i ($0 \leq j < n_F$ and ($1 \leq i \leq n_E$))	29
n_F	number of frames in a video sequence	30
n_E	number of enhancement layers	33
S_{GOP}	number of frames in GOP	35
P_N	probability to loose a frame during the transmission	36
P_D	probability that a frame cannot be decoded	36
$P_d(x)$	probability that frame x cannot be decoded	36
B_t^R	effective available bandwidth at time t	63
B_t	calculated available bandwidth at time t	63
S_t	protocol buffer fullness at time t	63

$W_{(t-\delta t;t]}$	amount of data written to the buffer in the time period $(t-\delta t;t]$	63
R	bit rate of data stream	63
B^e	unaccounted bandwidth	63
R^j	average bit rate of layer j ($j=0$ for L_B , j=k for $L_{E,k}$)	65
ξ_N	probability that layers 0 to N can be transmitted successfully (layer 0 is L_B)	65
$P(x)$	probability of x	65
F_{D^N}	cumulative probability function of D^N	66
E_K	probability that only K layers can be transmitted successfully	66
R_B	bit rate of L_B	68
$d(x)$	distortion (MSE) as function of bit rate x	68
$R_{E,i}$	bit rate of $L_{E,i}$	69
R^O	bit rate overhead for having an additional enhancement layer	69
$\delta d(R_B, R_{ELs})$	difference in distortion introduced by transcoding video into a BL with bit rate R_B plus ELs with overall bit rate R_{ELs} and a single BL with bit rate R_B	69
Q^i	quality of configuration i	72
Q_j	quality achieved by the sum of layers from 0 to j	72
N_C	number of layer configurations	72

N_L	number of layers of scalable video	77
q_i	quality level i	79
ϕ^j	highest layer received for frame number j	81
$P_a(s, s^{'})$	probability that action a in state s at time t will lead to state $s^{'}$ at time $t+1$	83
$V(s)$	immediate reward received in state s	83
$\overline{V}$	reward function	83
T^P	time between two successive deadlines (period)	83
b	time budget	83
Λ_i	number of processed layers for frame number i	88
C_i	processing time for frame number i	88
R_S	bit rate of all layers of a scalable video	94

Summary

In this thesis we present a new SNR scalable video coding scheme. An important advantage of the proposed scheme is that it requires just a standard video decoder for processing each layer. The quality of the delivered video depends on the allocation of bit rates to the base and enhancement layers. For a given total bit rate, the combination with a bigger base layer delivers higher quality. The absence of dependencies between frames in enhancement layers makes the system resilient to losses of arbitrary frames from an enhancement layer. Furthermore, that property can be used in a more controlled fashion.

An important characteristic of any video streaming scheme is the ability to handle network bandwidth fluctuations. We made a streaming technique that observes the network conditions and based on the observations reconfigures the layer configuration in order to achieve the best possible quality. A change of the network conditions forces a change in the number of layers or the bit rate of these layers. Knowledge of the network conditions allows delivery of a video of higher quality by choosing an optimal layer configuration. When the network degrades, the amount of data transmitted per second is decreased by skipping frames from an enhancement layer on the sender side. The presented video coding scheme allows skipping any frame from an enhancement layer, thus enabling an efficient real-time control over transmission at the network level and fine-grained control over the decoding of video data. The methodology proposed is not MPEG-2 specific and can be applied to other coding standards.

We made a terminal resource manager that enables trade-offs between quality and resource consumption due to the use of scalable video coding in combination with scalable video algorithms. The controller developed for the decoding process optimizes the perceived quality with respect to the CPU power available and the amount of input data. The controller does not depend on the type of scalability technique and can therefore be used with any scalable video. The controller uses the strategy that is created offline by means of a Markov Decision Process. During the evaluation it was found that the correctness of the controller behavior depends on the correctness of parameter settings for MDP, so user tests should be employed to find the optimal settings.

Samenvatting

In dit proefschrift presenteren we een nieuw SNR schaalbare video coderingstechniek. Een belangrijk voordeel van de voorgestelde techniek is dat een standaard video decodeerder gebruikt kan worden voor het verwerken van de verschillende lagen. De kwaliteit van de geleverde video hangt af van de verdeling van bandbreedte tussen de basislaag en de verbeteringslagen. Voor een gegeven totale bandbreedte levert de combinatie met een grotere basislaag een betere kwaliteit. De afwezigheid van afhankelijkheden tussen beelden van een verbeteringslaag maakt de techniek robuust tegen het verliezen van willekeurige beelden van een verbeteringslaag. Verder kan deze eigenschap in een gecontroleerde manier worden gebruikt.

Een belangrijke eigenschap van een video transmissie techniek is de mogelijkheid om veranderingen in de bandbreedte van het netwerk correct af te handelen. Wij hebben een transmissie techniek gemaakt die de eigenschappen van het netwerk waarneemt en deze waarnemingen gebruikt om de configuratie van de lagen aan te passen, zodanig dat de best mogelijke kwaliteit wordt bereikt. Een verandering van de netwerkeigenschappen resulteert in een verandering van het aantal lagen of in de toekenning van bandbreedte aan deze lagen. Kennis van de netwerkeigenschappen maakt het mogelijk om een video van hogere kwaliteit te leveren door de beste configuratie van lagen te selecteren. Wanneer het netwerk verslechtert zal de zender de hoeveelheid data die per seconde wordt verstuurd verminderen door middel van het overslaan van beelden van de verbeteringslaag. De gepresenteerde video coderingstechniek staat het overslaan van willekeurige beelden van een verbeteringslaag toe, waardoor het mogelijk is om efficiënt in real-time de transmissie op netwerk niveau te controleren, in combinatie met een fijnmazige controle van de decodering van de video data. De voorgestelde techniek is niet specifiek voor MPEG-2 en kan worden toegepast in combinatie met andere coderingsstandaarden.

Wij hebben een manager van de beschikbare middelen gemaakt die een afweging tussen kwaliteit en beschikbare middelen gebruik toestaat door het toepassen van schaalbare videocodering in combinatie met schaalbare videoalgoritmen. Het ontwikkelde controlemechanisme voor het decodeerproces optimaliseert de waargenomen kwaliteit met betrekking tot de beschikbare rekenkracht en de

hoeveelheid video gegevens. Het controlemechanisme hangt niet af van de schaalbaarheidstechniek en kan daardoor voor willekeurige schaalbare video worden gebruikt. Het controlemechanisme gebruikt de strategie die van te voren is berekend met behulp van een Markov beslissingsmodel. Tijdens de evaluatie is gevonden dat de juistheid van het controlemechanisme afhangt van de juistheid van de parameter configuratie voor het Markov beslissingsmodel, zodat gebruikerstests moeten worden aangewend om de optimale configuratie te vinden.

Acknowledgements

> Without friends no one would choose to live, though he had all other goods.
>
> Aristotle (384 BC – 322 BC), Nichomachean Ethics.

The last lines of this thesis should be words of gratitude to all those who helped to make the thesis possible. First of all, I thank Peter de With for being a patient supervisor, for his confidence in my work and for guiding me through the most difficult times of my PhD work – the last year. I especially thank him for helping me to make my thesis something to be proud of. I thank Peter van der Stok, my supervisor at Philips Research, for creating the opportunity to continue my Software Technology project as a PhD research project. I want to thank him for continuous motivation and support. He has been a constant source of inspiration, enthusiasm and knowledge all these years. I'd like to thank Johan Lukkien, my supervisor at Eindhoven University of Technology, for welcoming me in his group, for weekly supervision, for supporting this work with ideas, criticism, etc. He has always offered continual support for my research. I specially thank my supervisors for their fundamental guidance in these years. They helped me to keep my research connected to a real environment, to avoid spreading my attention on too many subjects, and to remember that doing research for the purpose of doing a research is not worthwhile. At the same time, they continuously encouraged me to look beyond the boarders of my project, to study problems from different directions, and to abstract, whenever is necessary, from pure implementation aspects. I'm really grateful to my supervisors for helping me to keep the right balance between the research and development.

I especially thank my colleagues at Philips Research, and particularly Jeffrey Kang, Harmke de Groot, Laurentiu Papalau, and Iulian Nitescu for helping me in research and implementation of some of the ideas presented in this thesis. I thank Maddy Jansen and Reinder Haakma for introducing me to user testing. This was a new experience for me. I want to thank Michael van Hartskamp for inspiring and helpful discussions on the topic of in-home networking. I thank Clemens Wüst for being a good colleague. His work on scalable video processing provided important basics for my research on terminal resource management. I'd like to thank Felix

Ogg who inspired my work on video streaming with his solution for smoother streaming over wireless networks. I also want to thank Jean Gelissen for providing me with an environment and support for my work at Philips Research.

My thanks have to be extended to the members of the SAN group who have contributed greatly to my personal and professional time at Eindhoven University of Technology. I'm grateful for their continues interest in my work, expiring questions and suggestions. I'd like to thank Reinder Bril for providing me with good hints for the thesis. My first thanks goes to Richard Verhoeven for his help with testing my ideas, for fruitful discussions about video streaming, for reviewing my thesis, and, separately, for helping me with the Dutch summary.

I want to thank Sergei Kozlov and Alina Weffers-Albu for the joined work on PROGRESS project "Quality of Service for In-Home Digital Networks". I also thank the other members of the project, for helpful comments and suggestions. I thank project leader Emile Aarts for the fruitful discussions we had.

I want to thank Harold Weffers and Maggy de Wert for supporting me those years when I was a student at Software Technology program and for extending their support for my PhD times.

For this dissertation I would like to thank my reading committee members: Jean-Dominique Decotignie, Christian Hentschel, and Jan Bergmans for their time, interest, and helpful comments. I would also like to thank another member of my defense committee, Kees van Berkel, for his time and insightful questions.

I thank my girlfriend Snezhana for making me happy and giving me the extra strength, motivation and love necessary to get things done. Without her this thesis would not be possible. I would like to express my gratitude to my parents, to my sister Katya and to the rest of my family for their constant moral support, love and care. Finally, I would like to thank my friends for the support they gave me during my work on this thesis.

Curriculum Vitae

Dmitri Jarnikov was born on 11 January 1978 in Minsk. He received his M.Sc. in Radiophysics at the Faculty of Radiophysics and Electronics of Belarussian State University in 2000. His graduation project “Speaker Identification” was completed at Philips Research in Eindhoven. Later that year, Dmitri Jarnikov graduated in Business Economics at Belarussian State University. Dmitri joined the post-master Software Technology education and training program at the Stan Ackermans Institute in 2001. During his industrial design and development project, Dmitri contributed to “Kean Integration of SubSystems” project at Philips Research in Eindhoven. After the completion of the Software Technology education and training program Dmitri was granted the title Professional Doctorate in Engineering (PDEng).

Dmitri Jarnikov started his PhD work within the Department of Mathematics and Computer Science in group System Architecture and Networking in 2003. The project was co-hosted by Philips Research in Eindhoven. During his PhD, Dmitri published four papers, co-authored two book chapters, two Philips technical reports and a journal article. He contributed to national and international projects and presented his work at various international conferences. Dmitri has been awarded two patents for the inventions made during his project.

Titles in the IPA Dissertation Series since 2002

M.C. van Wezel. *Neural Networks for Intelligent Data Analysis: theoretical and experimental aspects*. Faculty of Mathematics and Natural Sciences, UL. 2002-01

V. Bos and J.J.T. Kleijn. *Formal Specification and Analysis of Industrial Systems*. Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2002-02

T. Kuipers. *Techniques for Understanding Legacy Software Systems*. Faculty of Natural Sciences, Mathematics and Computer Science, UvA. 2002-03

S.P. Luttik. *Choice Quantification in Process Algebra*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-04

R.J. Willemen. *School Timetable Construction: Algorithms and Complexity*. Faculty of Mathematics and Computer Science, TU/e. 2002-05

M.I.A. Stoelinga. *Alea Jacta Est: Verification of Probabilistic, Real-time and Parametric Systems*. Faculty of Science, Mathematics and Computer Science, KUN. 2002-06

N. van Vugt. *Models of Molecular Computing*. Faculty of Mathematics and Natural Sciences, UL. 2002-07

A. Fehnker. *Citius, Vilius, Melius: Guiding and Cost-Optimality in Model Checking of Timed and Hybrid Systems*. Faculty of Science, Mathematics and Computer Science, KUN. 2002-08

R. van Stee. *On-line Scheduling and Bin Packing*. Faculty of Mathematics and Natural Sciences, UL. 2002-09

D. Tauritz. *Adaptive Information Filtering: Concepts and Algorithms*. Faculty of Mathematics and Natural Sciences, UL. 2002-10

M.B. van der Zwaag. *Models and Logics for Process Algebra*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-11

J.I. den Hartog. *Probabilistic Extensions of Semantical Models*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2002-12

L. Moonen. *Exploring Software Systems*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-13

J.I. van Hemert. *Applying Evolutionary Computation to Constraint Satisfaction and Data Mining*. Faculty of Mathematics and Natural Sciences, UL. 2002-14

S. Andova. *Probabilistic Process Algebra*. Faculty of Mathematics and Computer Science, TU/e. 2002-15

Y.S. Usenko. *Linearization in μCRL*. Faculty of Mathematics and Computer Science, TU/e. 2002-16

J.J.D. Aerts. *Random Redundant Storage for Video on Demand*. Faculty

of Mathematics and Computer Science, TU/e. 2003-01

M. de Jonge. *To Reuse or To Be Reused: Techniques for component composition and construction*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2003-02

J.M.W. Visser. *Generic Traversal over Typed Source Code Representations*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2003-03

S.M. Bohte. *Spiking Neural Networks*. Faculty of Mathematics and Natural Sciences, UL. 2003-04

T.A.C. Willemse. *Semantics and Verification in Process Algebras with Data and Timing*. Faculty of Mathematics and Computer Science, TU/e. 2003-05

S.V. Nedea. *Analysis and Simulations of Catalytic Reactions*. Faculty of Mathematics and Computer Science, TU/e. 2003-06

M.E.M. Lijding. *Real-time Scheduling of Tertiary Storage*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2003-07

H.P. Benz. *Casual Multimedia Process Annotation – CoMPAs*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2003-08

D. Distefano. *On Modelchecking the Dynamics of Object-based Software: a Foundational Approach*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2003-09

M.H. ter Beek. *Team Automata – A Formal Approach to the Modeling of Collaboration Between System Components*. Faculty of Mathematics and Natural Sciences, UL. 2003-10

D.J.P. Leijen. *The λ Abroad – A Functional Approach to Software Components*. Faculty of Mathematics and Computer Science, UU. 2003-11

W.P.A.J. Michiels. *Performance Ratios for the Differencing Method*. Faculty of Mathematics and Computer Science, TU/e. 2004-01

G.I. Jojgov. *Incomplete Proofs and Terms and Their Use in Interactive Theorem Proving*. Faculty of Mathematics and Computer Science, TU/e. 2004-02

P. Frisco. *Theory of Molecular Computing – Splicing and Membrane systems*. Faculty of Mathematics and Natural Sciences, UL. 2004-03

S. Maneth. *Models of Tree Translation*. Faculty of Mathematics and Natural Sciences, UL. 2004-04

Y. Qian. *Data Synchronization and Browsing for Home Environments*. Faculty of Mathematics and Computer Science and Faculty of Industrial Design, TU/e. 2004-05

F. Bartels. *On Generalised Coinduction and Probabilistic Specification Formats*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2004-06

L. Cruz-Filipe. *Constructive Real Analysis: a Type-Theoretical Formal-*

ization and Applications. Faculty of Science, Mathematics and Computer Science, KUN. 2004-07

E.H. Gerding. *Autonomous Agents in Bargaining Games: An Evolutionary Investigation of Fundamentals, Strategies, and Business Applications*. Faculty of Technology Management, TU/e. 2004-08

N. Goga. *Control and Selection Techniques for the Automated Testing of Reactive Systems*. Faculty of Mathematics and Computer Science, TU/e. 2004-09

M. Niqui. *Formalising Exact Arithmetic: Representations, Algorithms and Proofs*. Faculty of Science, Mathematics and Computer Science, RU. 2004-10

A. Löh. *Exploring Generic Haskell*. Faculty of Mathematics and Computer Science, UU. 2004-11

I.C.M. Flinsenberg. *Route Planning Algorithms for Car Navigation*. Faculty of Mathematics and Computer Science, TU/e. 2004-12

R.J. Bril. *Real-time Scheduling for Media Processing Using Conditionally Guaranteed Budgets*. Faculty of Mathematics and Computer Science, TU/e. 2004-13

J. Pang. *Formal Verification of Distributed Systems*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2004-14

F. Alkemade. *Evolutionary Agent-Based Economics*. Faculty of Technology Management, TU/e. 2004-15

E.O. Dijk. *Indoor Ultrasonic Position Estimation Using a Single Base Station*. Faculty of Mathematics and Computer Science, TU/e. 2004-16

S.M. Orzan. *On Distributed Verification and Verified Distribution*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2004-17

M.M. Schrage. *Proxima - A Presentation-oriented Editor for Structured Documents*. Faculty of Mathematics and Computer Science, UU. 2004-18

E. Eskenazi and A. Fyukov. *Quantitative Prediction of Quality Attributes for Component-Based Software Architectures*. Faculty of Mathematics and Computer Science, TU/e. 2004-19

P.J.L. Cuijpers. *Hybrid Process Algebra*. Faculty of Mathematics and Computer Science, TU/e. 2004-20

N.J.M. van den Nieuwelaar. *Supervisory Machine Control by Predictive-Reactive Scheduling*. Faculty of Mechanical Engineering, TU/e. 2004-21

E. Ábrahám. *An Assertional Proof System for Multithreaded Java -Theory and Tool Support-* . Faculty of Mathematics and Natural Sciences, UL. 2005-01

R. Ruimerman. *Modeling and Remodeling in Bone Tissue*. Faculty of Biomedical Engineering, TU/e. 2005-02

C.N. Chong. *Experiments in Rights Control - Expression and Enforcement*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-03

H. Gao. *Design and Verification of Lock-free Parallel Algorithms*. Faculty of Mathematics and Computing Sciences, RUG. 2005-04

H.M.A. van Beek. *Specification and Analysis of Internet Applications*. Faculty of Mathematics and Computer Science, TU/e. 2005-05

M.T. Ionita. *Scenario-Based System Architecting - A Systematic Approach to Developing Future-Proof System Architectures*. Faculty of Mathematics and Computing Sciences, TU/e. 2005-06

G. Lenzini. *Integration of Analysis Techniques in Security and Fault-Tolerance*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-07

I. Kurtev. *Adaptability of Model Transformations*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-08

T. Wolle. *Computational Aspects of Treewidth - Lower Bounds and Network Reliability*. Faculty of Science, UU. 2005-09

O. Tveretina. *Decision Procedures for Equality Logic with Uninterpreted Functions*. Faculty of Mathematics and Computer Science, TU/e. 2005-10

A.M.L. Liekens. *Evolution of Finite Populations in Dynamic Environments*. Faculty of Biomedical Engineering, TU/e. 2005-11

J. Eggermont. *Data Mining using Genetic Programming: Classification and Symbolic Regression*. Faculty of Mathematics and Natural Sciences, UL. 2005-12

B.J. Heeren. *Top Quality Type Error Messages*. Faculty of Science, UU. 2005-13

G.F. Frehse. *Compositional Verification of Hybrid Systems using Simulation Relations*. Faculty of Science, Mathematics and Computer Science, RU. 2005-14

M.R. Mousavi. *Structuring Structural Operational Semantics*. Faculty of Mathematics and Computer Science, TU/e. 2005-15

A. Sokolova. *Coalgebraic Analysis of Probabilistic Systems*. Faculty of Mathematics and Computer Science, TU/e. 2005-16

T. Gelsema. *Effective Models for the Structure of pi-Calculus Processes with Replication*. Faculty of Mathematics and Natural Sciences, UL. 2005-17

P. Zoeteweij. *Composing Constraint Solvers*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-18

J.J. Vinju. *Analysis and Transformation of Source Code by Parsing*

and Rewriting. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-19

M.Valero Espada. *Modal Abstraction and Replication of Processes with Data*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2005-20

A. Dijkstra. *Stepping through Haskell*. Faculty of Science, UU. 2005-21

Y.W. Law. *Key management and link-layer security of wireless sensor networks: energy-efficient attack and defense*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-22

E. Dolstra. *The Purely Functional Software Deployment Model*. Faculty of Science, UU. 2006-01

R.J. Corin. *Analysis Models for Security Protocols*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-02

P.R.A. Verbaan. *The Computational Complexity of Evolving Systems*. Faculty of Science, UU. 2006-03

K.L. Man and R.R.H. Schiffelers. *Formal Specification and Analysis of Hybrid Systems*. Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2006-04

M. Kyas. *Verifying OCL Specifications of UML Models: Tool Support and Compositionality*. Faculty of Mathematics and Natural Sciences, UL. 2006-05

M. Hendriks. *Model Checking Timed Automata - Techniques and Applications*. Faculty of Science, Mathematics and Computer Science, RU. 2006-06

J. Ketema. *Böhm-Like Trees for Rewriting*. Faculty of Sciences, VUA. 2006-07

C.-B. Breunesse. *On JML: topics in tool-assisted verification of JML programs*. Faculty of Science, Mathematics and Computer Science, RU. 2006-08

B. Markvoort. *Towards Hybrid Molecular Simulations*. Faculty of Biomedical Engineering, TU/e. 2006-09

S.G.R. Nijssen. *Mining Structured Data*. Faculty of Mathematics and Natural Sciences, UL. 2006-10

G. Russello. *Separation and Adaptation of Concerns in a Shared Data Space*. Faculty of Mathematics and Computer Science, TU/e. 2006-11

L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices*. Faculty of Science, Mathematics and Computer Science, RU. 2006-12

B. Badban. *Verification techniques for Extensions of Equality Logic*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2006-13

A.J. Mooij. *Constructive formal methods and protocol standardization*. Faculty of Mathematics and Computer Science, TU/e. 2006-14

T. Krilavicius. *Hybrid Techniques for Hybrid Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-15

M.E. Warnier. *Language Based Security for Java and JML*. Faculty of Science, Mathematics and Computer Science, RU. 2006-16

V. Sundramoorthy. *At Home In Service Discovery*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-17

B. Gebremichael. *Expressivity of Timed Automata Models*. Faculty of Science, Mathematics and Computer Science, RU. 2006-18

L.C.M. van Gool. *Formalising Interface Specifications*. Faculty of Mathematics and Computer Science, TU/e. 2006-19

C.J.F. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Faculty of Mathematics and Computer Science, TU/e. 2006-20

J.V. Guillen Scholten. *Mobile Channels for Exogenous Coordination of Distributed Systems: Semantics, Implementation and Composition*. Faculty of Mathematics and Natural Sciences, UL. 2006-21

H.A. de Jong. *Flexible Heterogeneous Software Systems*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-01

N.K. Kavaldjiev. *A run-time reconfigurable Network-on-Chip for streaming DSP applications*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-02

M. van Veelen. *Considerations on Modeling for Early Detection of Abnormalities in Locally Autonomous Distributed Systems*. Faculty of Mathematics and Computing Sciences, RUG. 2007-03

T.D. Vu. *Semantics and Applications of Process and Program Algebra*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-04

L. Brandán Briones. *Theories for Model-based Testing: Real-time and Coverage*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-05

I. Loeb. *Natural Deduction: Sharing by Presentation*. Faculty of Science, Mathematics and Computer Science, RU. 2007-06

M.W.A. Streppel. *Multifunctional Geometric Data Structures*. Faculty of Mathematics and Computer Science, TU/e. 2007-07

N. Trčka. *Silent Steps in Transition Systems and Markov Chains*. Faculty of Mathematics and Computer Science, TU/e. 2007-08

R. Brinkman. *Searching in encrypted data*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-09

A. van Weelden. *Putting types to good use*. Faculty of Science, Math-

ematics and Computer Science, RU. 2007-10

J.A.R. Noppen. *Imperfect Information in Software Development Processes*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-11

R. Boumen. *Integration and Test plans for Complex Manufacturing Systems*. Faculty of Mechanical Engineering, TU/e. 2007-12