Design of a multi-channel streaming video server for surveillance systems

Wijnhoven, R.G.J.

Award date:
2004

Link to publication
Design of a Multi-Channel Streaming Video Server for Surveillance Systems

by R.G.J. Wijnhoven

Master of Science thesis

Project period: May 2003 – May 2004
Report Number: 11-04

Commissioned by:

Supervisors: Prof. dr. ir. P.H.N. de With
Dr. ir. E.G.T. Jaspers
Ir. M.C. Jacobs

Additional Commission members: Prof. dr. ir. P.H.N. de With
Dr. ir. F.M.J. Willems

The Department of Electrical Engineering of the Eindhoven University of Technology accepts no responsibility for the contents of M.Sc. theses or practical training reports.
Design of a Multi-Channel Streaming Video Server for Surveillance Systems

R.G.J. Wijnhoven

May 27, 2004
Preface

This thesis presents the research done during the graduation project as part of the Master of Science (MSc) studies at the Department of Electrical Engineering of the Technische Universiteit Eindhoven (TU/e). The project was part of the Signal Processing Systems (SPS) chair of the Measurement and Control (MBS) group of the department. The research was carried out at Bosch Security Systems B.V. in Eindhoven.
Abstract

In the video surveillance market, digital video recorders (DVRs) are available for some time. DVRs store video from multiple video channels (cameras) and offer remote display of recorded and live video over a network connection.

There is a shift from still-image based video compression towards compression methods that exploit the temporal relation of video (e.g. as addressed by the MPEG standards). Because of the temporal relationship of the compressed video frames, scalability in bit-rate by modifying the frame-rate cannot easily be obtained. For transmission of this video over a network connection with varying bandwidth, however, such scalability is required. Several methods to obtain bit-rate scalability are considered. For our system, multiple independently encoded video streams will be used for each video channel. Each stream will have a different frame-rate, and thus a different bit-rate.

Based on this system approach, an architecture for a streaming video server is proposed. The architecture allows multiple users to connect and receive video, simultaneously. Each user can request multiple video channels. The transmission bit-rate is adjusted by selecting different combinations of compressed video streams for the requested channels. A stream-selection module calculates a set of streams for each user, depending on the available network bandwidth.

The main focus of our work was on the stream-selection process. To distinguish between multiple sets of streams, parameters are introduced, which are used to find the optimal set of streams. Based on these parameters, cost values are introduced to allow numerical comparison of different sets of streams. Because a full-search over the total solution space is not feasible, several heuristic selection algorithms are proposed.

Multiple stream-selection algorithms are compared for a scenario in which the network bandwidth is varied over time. A comparison is made in terms of computational complexity versus the performance. Next, because each heuristic method considers only a limited set of combinations of streams, the scalability for each method is compared to the full-search method. We show that, the considered sub-set contains relevant combinations of streams, so a reduction in the computational load still results in an optimal combination of streams.
Contents

Preface i
Abstract iii

1 Introduction 1
1.1 Problem definition 2
1.2 Related work 3
1.3 Document Layout 4

2 Bit-Rate Scalability 5
2.1 Introduction 5
2.2 MPEG video compression 5
2.3 System approaches 6
   2.3.1 Approach 1: multiple independent encoders 7
   2.3.2 Approach 2: scaling methods 8
   2.3.3 Approach 3: layered encoding 10
   2.3.4 Approach 4: adjusting encoder 10
2.4 Architectural choice 11

3 Server Architecture 13
3.1 Introduction 13
3.2 Modules 14
   3.2.1 Source reader module 14
   3.2.2 Bit-rate scaler module 14
   3.2.3 Dispatcher module 14
   3.2.4 Controller module 15
   3.2.5 Stream-selection module 15
   3.2.6 Network module 15
3.3 System implementation 17
   3.3.1 Modules 17
   3.3.2 Usage 19
   3.3.3 Software details 19

4 Stream Selection 21
4.1 Introduction 21
4.2 Optimal stream-combination: parameters 23
   4.2.1 Feasibility parameters 23
   4.2.2 Optimum parameters 24
4.3 Cost values

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.1 Bandwidth utilization cost</td>
<td>26</td>
</tr>
<tr>
<td>4.3.2 Switch-delay cost</td>
<td>27</td>
</tr>
<tr>
<td>4.3.3 Time-since-last-switch cost</td>
<td>28</td>
</tr>
<tr>
<td>4.3.4 Video quality</td>
<td>29</td>
</tr>
<tr>
<td>4.3.5 Priority partitioning cost</td>
<td>30</td>
</tr>
<tr>
<td>4.3.6 Combined cost</td>
<td>31</td>
</tr>
</tbody>
</table>

4.4 Methods for selection

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4.1 Introduction</td>
<td>33</td>
</tr>
<tr>
<td>4.4.2 Heuristic: switch limit number of channels</td>
<td>34</td>
</tr>
<tr>
<td>4.4.3 Heuristic: independent sub-problems</td>
<td>37</td>
</tr>
<tr>
<td>4.4.4 Heuristic: Pruning</td>
<td>38</td>
</tr>
</tbody>
</table>

4.5 Combinations of heuristics                   | 41   |
4.6 Frequency of stream-selection                 | 41   |
4.7 Conclusions                                  | 41   |

5 Results

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Introduction</td>
<td>43</td>
</tr>
<tr>
<td>5.2 Performance vs. computational complexity</td>
<td>44</td>
</tr>
<tr>
<td>5.3 Scalability vs. optimality cost</td>
<td>51</td>
</tr>
<tr>
<td>5.4 Priorities</td>
<td>57</td>
</tr>
<tr>
<td>5.5 Constant bit-rate vs. variable bit-rate video</td>
<td>57</td>
</tr>
</tbody>
</table>

6 Conclusions                                    | 61   |

7 Recommendations                                 | 63   |

Glossary                                         | 65   |
List of Figures                                  | 67   |
List of Tables                                   | 69   |
Acknowledgements                                 | 71   |
References                                       | 73   |

A Video Compression                              | 77   |

B Network Transmission                           | 79   |
| B.1 Introduction                                | 79   |
| B.2 Congestion control                          | 79   |
| B.2.1 Binomial congestion control               | 80   |
| B.3 Receiver buffering                          | 80   |
| B.4 Multiplexing of multiple MPEG streams       | 80   |
| B.5 Network protocols                           | 81   |
Chapter 1

Introduction

In the video security market, digital video recorders (DVRs) have been available for some time. DVRs store video from multiple video channels (cameras) and offer remote display of recorded and live video over a network connection. Multiple clients connect to a DVR and want to view video simultaneously (see Figure 1.1).

DVRs often use still-image-based compression techniques like wavelet- or JPEG-compression. Because each frame is independently encoded, adapting the bit-rate of each video stream to the network bandwidth can be done by sending out less frames per second. To increase video quality and storing times, there is a shift to compression techniques that exploit the temporal correlation of video, called inter-coded compression. These techniques are often based on one of the MPEG standards. Because of the temporal dependency between frames, it is harder to adjust the video bit-rate to adapt to changes in the network bandwidth.

To be able to adapt the transmission rate to the available network bandwidth, it is clear that some form of bit-rate scalability is required. Methods to obtain scalability are presented in Chapter 2.

Figure 1.1: Digital video recorder (DVR) with remote video streaming.
1.1 Problem definition

The constraints in the surveillance world are quite different, compared to the consumer world. Very low delays are required, for manual control of pan-tilt-zoom cameras. Also very low frame-rates are common use (consider frame-rates of 1 frame-per-second, or even less).

The following constraints are identified in our system:

- **Multiple users** can connect to the system. Compared to streaming servers in the consumer world, this number is highly limited. Where thousands of users can connect to a server broadcasting a live pop-concert, in the surveillance world only a limited number of users is allowed to stream video. This means that per user more computation power is available. Streaming servers for the consumer market, on the other hand, are often only used to stream video to clients and don’t have other tasks that are required for a surveillance system.

- **Bandwidth Limitation.** The available network bandwidth can be limited. This limit can be set by the system administrator, the connected user, or is restricted by the network conditions. We assume there is a module available that continuously gives us the available network bandwidth for each connected user. The system should make sure that the video that is sent to each user remains within this bandwidth limitation. Much research in the field of network transmission for streaming video applications has been conducted and can be found in Appendix B.

- **Decoding Power Limitation.** For decoding of received video, connected users can use different systems, that can all have different decoding power (compare a desktop PC with a hand-held device). In each case, the user should be able to decode the received video, so the system should take this limit into account when sending video for the requested channels.

- **Limited End-to-End Delay.** The end-to-end delay in surveillance is different from the consumer world, where often a high amount of buffering can be applied (up to a few seconds). For live video, especially when cameras are controlled by the user (consider joystick-control for pan-tilt-zoom (PTZ) cameras) this delay should be strictly limited. As a result, buffering of compressed video data at the user should be minimal. For streaming of pre-recorded video however, this delay constraint is more relaxed.

For each connected user, the considered system should transmit video for each requested video channel, so that

- The total bit-rate of the transmitted video remains below the available network bandwidth,

- The user should be able to decode all the received video,

- The perceived quality at each user is optimal.

Chapter 4 will discuss identified constraints and explains methods for achieving the mentioned goal.
1.2 Related work

Most systems currently on the market that transmit inter-coded video are single-user based, or broadcast a single video channel to multiple users. Furthermore, they don't offer scalability in bit-rate: only a limited number of (pre-)compressed streams is available, where often the selection has to be made manually. Examples are streaming video servers on the Internet: watching movie trailers (on-demand) or broadcasts of concerts (live). Multi-channel video transmission is done in the broadcast world, where video is transmitted over links with dedicated bandwidth (e.g. satellite links or cable-network channels). Such systems are proposed by Keesman in [1] and Haskell and Reibman in [2].

Multiple systems have been designed for the transmission of compressed video. Perhaps best known are the mediaplayers that of playback of streaming video over the Internet, often used for movie trailers or news bulletins. The big players in this field are Microsoft's Windows Media Player¹, Apple's Quicktime² and Real Networks' RealPlayer³. In this case, the video is pre-compressed at a constant bit-rate. If the bandwidth of the network link is not sufficient, the buffer at the client will run empty and needs to be re-filled. This results in smooth playback (because of the amount of buffering) with periods of video-pause caused by buffering. Often the user can manually select between some streams with different bit-rate (e.g. 56 kbps and 300 kbps).

Part 6 of the MPEG-4 standard also does some recommendations on the transport of MPEG-4 data. The framework for this transport is the Delivery Multimedia Integration Framework (DMIF). This framework is discussed by Wu et al. in [3] and Kalva et al. in [4].

Another system is presented by Rejaie et al. in [5]. The authors identify three main building blocks for an Internet streaming application: end-to-end congestion control, quality adaptation and error control. A combination of buffering and layered encoding is used to obtain quality adaptation. Error control is done in the form of selective retransmission of lost packets. More recent work is presented in [6]. Feamster et al. also present retransmission of important parts of the MPEG bit-stream in [7]. McCanne presents methods for scalability in multicast video in [8]. Koliver et al. propose a mapping of QoS parameters onto metrics, a fuzzy controller that adapts the application, and filter algorithms that adapt the bandwidth utilization in [9].

¹http://www.microsoft.com/windows/windowsmedia/
²http://www.apple.com/quicktime/
³http://www.real.com/
1.3 Document Layout

Several methods to obtain bit-rate scalability are considered in Chapter 2. For our system, multiple independently encoded video streams for each video channel will be used. Each stream will have a different frame-rate, and thus a different bit-rate.

Based on this selected approach, a server architecture is proposed in Chapter 3. The architecture allows multiple users to connect and receive video simultaneously. Each user can request multiple video channels. Input to the architecture are multiple independently encoded streams per video channel as explained in the chapter on bit-rate scalability. The use of bit-rate scalers can enhance the bit-rate scalability if required. A stream-selection module calculates a new set of streams for each user.

Chapter 4 discusses the implementation of the stream-selection module. Parameters that define the optimal set of streams are presented. Based on these parameters, cost values are introduced to allow numerical comparison of different sets of streams. Multiple selection algorithms that calculate a new set of streams are considered. Their complexity is discussed.

The results are discussed in Chapter 5. Multiple stream-selection algorithms are compared for one scenario in which the network bandwidth is varied over time. A comparison is made in terms of computational complexity versus the performance of each selection method. Because each heuristic method considers only a limited set of combinations of streams, each method is compared to the full-search method, where all possible combinations of streams are considered.

Chapter 6 concludes.

Actual network transmission is not considered, but the interested reader can find information on this topic in Appendix B.
Chapter 2

Bit-Rate Scalability

2.1 Introduction

In order to send video continuously over a network link with varying bandwidth, the total bit-rate of the streams that are transmitted, should be constantly equal to or below the available network bandwidth. When a compression scheme is used that compresses each frame independently scalability in bit-rate can easily be obtained by simply dropping frames for transmission. Because of the temporal relationship between compressed frames in MPEG compression schemes, however, scalability cannot easily be obtained by dropping frames, since the subsequent frames cannot be decoded correctly anymore then.

In the next section, the basics of MPEG compression schemes are explained. Next, multiple system approaches for scalability in bit-rate are discussed, for systems that apply MPEG video compression. Finally, the selected system approach for our DVR system will be discussed.

2.2 MPEG video compression

As explained in the introduction (see Chapter 1), compression schemes used in DVRs are shifting from still-image based compression like JPEG or Wavelet to techniques that exploit the temporal relation of video. These techniques are often based on one of the MPEG standards. An introduction to the basics of MPEG encoding is given by Le Gall in [10].

MPEG defines a group of pictures (GOP) as one independently decodable frame (I-frame) followed by a number of predicted frames (P-frames). Such a predicted frame is encoded using the difference between the current frame and the motion-compensated previous frame. See Figure 2.1 for a graphical representation. Furthermore, B-frames are defined as bi-directionally coded frames, using the difference between the current frame and a combination of the previous I- or P-frame and the next I- or P-frame. This relation is visualized in Figure 2.2. The arrows in the figure represent the relation between the frames.

As a consequence of exploiting the temporal relationship of video frames, higher compression factors can be achieved. However, due to the dependencies between encoded frames, all preceding data in a GOP needs to be transmitted
in specified order such that all incoming data only depends on already decoded data. Subsequently, when using B-frames, reordering is required for correct displaying of the decoded data. Skipping frames without considering the coding scheme will result in errors in the decoded video. Thus, it is hard to adapt to changing network conditions compared to simply dropping frames when using still-image-based compression techniques. Scalability of inter-coded video is therefore less straightforward.

2.3 System approaches

This section will explain system approaches to obtain scalability in bit-rate to be able to adjust the video transmission rate to the available network bandwidth. Only one video channel is considered. Four systems for scalability in bit-rate are presented and will be discussed in more details:

1. Use multiple video-encoders, each coding with a defined bit-rate (constant or variable bit-rate), see Figure 2.3(a). The encoders create independent bit-streams at different bit-rates. Bit-rate scalability for transmission of video is obtained by switching between the available encoded streams. As a consequence, switching can only occur at GOP granularity.

2. Use one video-encoder that outputs one bit-stream at a pre-defined bit-rate (constant or variable bit-rate). Then use one or multiple scalers to create uncorrelated bit-streams at different bit-rates, see Figure 2.3(b).
Different types of scalers can be used as explained in the next subsection. Bit-rate scalability is obtained by selecting between the available scaled bit-streams.

3. Use one video-encoder that encodes multiple bit-streams, representing multiple hierarchical layers of video quality, see Figure 2.3(c). One base-layer and multiple enhancement layers are available. Each previous layer is required for the decoding of the next layer. The more layers are added to the base-layer, the higher the obtained video quality. Bit-rate scalability is obtained by adding layers to the base-layer.

4. Use one video-encoder and adjust the output bit-rate of the encoder to obtain bit-rate scalability, see Figure 2.3(d). Perfect granular scalability in bit-rate can be obtained at no additional cost, since video data has to be encoded anyway. This system will be less efficient, when multiple users desire different target bit-rates.

![Figure 2.3: Methods to obtain bit-rate scalability.](image)

**2.3.1 Approach 1: multiple independent encoders**

The most simple approach from an implementation point of view is the use of multiple independent encoders, that create independent bit-streams at different bit-rates. Scaling to the available network bandwidth is done by selecting the stream with appropriate bit-rate. Scalability in the surveillance world is commonly done by varying the frame-rate, and low-frame-rate streams are common use. The computational cost of an extra encoder at a low frame-rate is very limited. The scalability is coarse-grained however, limited by the bit-rates of the available streams. Also the switching between bit-streams has limitations. Because of the relation between the frames in a GOP-structure, when switching to a different stream, frames of this new stream cannot correctly be decoded until the availability of the next I-frame. Sun et al. [11] and Chou et al. [12] propose a scheme to achieve seamless switching between multiple streams at predictive frames, based on the use of SP-frames (as accepted as frame-type in H.26L).
2.3.2 Approach 2: scaling methods

The system in Figure 2.3(b) uses one encoder with one or multiple scalers. The input of a scaler is a compressed bit-stream coming from the encoder. The output is one or multiple compressed bit-streams, each with a lower bit-rate than the bit-rate of the input bit-stream, except for the original stream that is always available at the output. Two main categories of scalers can be distinguished:

- Scaling in the compressed domain. One method is re-quantization of DCT-coefficients as done by Amir et al. [13]. The compressed video is not fully decoded to limit the number of computations. Another method that can be used is the selectively dropping of frames, where no decoding of compressed video has to done. Research in this field has been done by Cha et al. [14], Amir et al. [13] and Walpole et al. [15].

- Scaling in the uncompressed domain. The video is fully decoded, filtered and then re-encoded. Examples of filters that can be applied are re-sampling at lower resolution, frame-rate adjustments or re-quantization of the luminance and/or chrominance samples. These methods are computationally intensive, but provide most flexibility and the highest picture quality.

Kang and Zakhor [16] introduce some other scalability methods targeted at transmission of video over wireless connections. One of the methods used is called MTD, where each P-frame is divided in a part with only motion information and texture information. The introduced methods will not be considered because they are targeted at very low bit-rates.

Scaling in the uncompressed domain is very computationally intensive. This computation power can also be used to add another independent encoder to the system. Therefore it will be better to use the system approach as in Figure 2.3(a).

Frame-dropping

An example of scaling in the compressed domain (this can also be done in the uncompressed domain though) is the selectively dropping of frames. A frame is called dropped, when it is not used for transmission and is completely ignored. Instead of sending all the compressed frames to the client, frames can be selectively dropped. Dropping frames causes a reduction in both bit-rate and frame-rate. By selectively dropping only certain frames, reductions in bit-rate can be obtained, against reasonable reductions in frame-rate. This selective dropping method is very computationally inexpensive, but the gain in bit-rate scalability is nihil.

Cha et al. [14] and Walpole et al. [15] use this method to obtain bandwidth-adaptation. Yeadon introduces filters to obtain scalability in [17] and [18], particularly targeted at multicast scenarios. Based on this work, Krasic and Walpole introduce priority-packet-dropping with layered media in [19], and also use frame-dropping to obtain scalability in bit-rate.

Consider a GOP-structure as in Figure 2.4, where only I- and P-frames are encoded. When frames 5 and 6 are dropped, frames 1 to 4 are transmitted and decoded. Only frames 1 and 4 are displayed however, to obtain a constant
2.3. SYSTEM APPROACHES

Figure 2.4: Frame Dropping for MPEG streams with only I- and P-frames.

frame-rate. This means that the reduction in bit-rate is the size of the two P-frames compared to the total GOP-size, whereas the reduction in frame-rate is a factor three. The larger the GOP-length used, the less efficient the method becomes.

Figure 2.5 shows the reduction in bit-rate, compared to the resulting frame-rate, for the case that only I- and P-frames are used. The GOP-length is 15 frames, the frame-rate 30 frames-per-second (fps). The size of I-frames is assumed ten times the size of P-frames. As can be seen from the figure, for obtaining a reduction in bit-rate of 20%, the frame-rate will be dropped from 30 to 5 fps.

Figure 2.5: Frame dropping: bandwidth reduction vs. resulting frame-rate.

Now lets consider a GOP-structure where also B-frames are used, see Figure 2.6. Since the B-frames are not used as a reference for other frames, dropping any B-frame will not have any consequences for the correct decoding of other frames. Therefore, a straightforward reduction in bit-rate can be obtained by dropping all B-frames. This would result in dropping frames 2,3,5 and 6. Only frames 1 and 4 will be transmitted and decoded. This gives a reduction in frame-rate of three, but a reduction in bit-rate corresponding to the size of the B-frames. The resulting GOP-structure contains only I- and P-frames. Further
frame-dropping can be applied as mentioned before.

The use of B-frames is very effective for the frame-dropping process, but causes extra computation power in the encoder and is therefore not used. Because the sizes of B-frames are also very small, the reduction in bit-rate is also limited.

2.3.3 Approach 3: layered encoding

Scalability in bit-rate can also be obtained by using layered-encoding, as visualized in Figure 2.3(c). The level of granularity depends on the number of layers used. However, the size of a hierarchically encoded bit-stream is much higher than a single high-quality video stream at the same video quality. Also the computational complexity of an encoder that produces layered video is much higher compared to the other approaches.

Krasic and Walpole introduce priority-packet-dropping with layered media in [19]. Scalability is added to MPEG. Their so-called SPEG introduces layered video by increasing the quantization by one bit per layer. At run time, the quantization may be adjusted by selection by some or all of the layers.

Sun et al. [11] propose to use a combination of layered encoding and independent bit-streams. A seamless switching scheme is described, to improve the efficiency of scalable video coding over a broad bit-rate range by using two scalable bit-streams.

MPEG defines scalable profiles like the Fine Granularity Scalability (FGS) profile, that include spatial, SNR and/or temporal scalability. Li gives an overview of FGS in [20]. This profile is defined in Amendment 4: Streaming Video Profile [21] of the MPEG-4 Visual specification [22]. Much research in the field of FGS encoding has recently be done. Significant work has been done by van der Schaar and Radha, see e.g. [23]. Kim and Ammar proposed a quality adaptation scheme and tested it over various network connections in [24].

2.3.4 Approach 4: adjusting encoder

System approach 4 as in Figure 2.3(d), proposes the use of an encoder with an dynamically adjustable target bit-rate. The obtained scalability is the best achievable, but only one target bit-rate can be specified. Therefore, this approach becomes less useful when multiple users with different network band-
width conditions request the video stream. Adding extra (adjustable or static) encoders overcomes this problem, but introduces extra costs that are not desired.

2.4 Architectural choice

Four systems were proposed in Section 2.3. For our DVR system, method 1 as in Figure 2.3(a) is used. This approach uses multiple independent encoders as can also be seen in Figure 2.7. The approach has been selected, because its implementation is relatively easy, and the computation power needed for streams of low frame-rates is limited.

Layered encoding as discussed in subsection 2.3.3 was not used, since the availability of layered encoders is limited and the required computation power is relatively high. For video storage in the security domain, only a low frame-rate stream is required when almost no activity is present in the scene (e.g. no motion), but higher frame-rate streams are required for a high level of activity in the scene. Therefore, it is preferred to have both a low- and high-frame-rate stream available. Moreover, a separate high-quality bit-stream is more efficient than a hierarchically encoded bit-stream with the same quality. Both for storage and network transmission, this overhead is not desired.

The use of encoders whose target bit-rates can be changed dynamically to match the network constraints is not applicable. For archival purposes, video of dedicated quality (bit-rate) should be stored. If the encoder is tuned to match network characteristics, the bit-rate (quality) of the compressed stream can vary significantly. This means that for storage purposes an extra encoder needs to be used to create a compressed stream with dedicated quality. When considering streaming of pre-recorded video, this approach also fails, since the stream still needs to be scaled in bit-rate to match the network bandwidth.

Bit-rate scalers as shown in Figure 2.3(b) are currently not used in our sys-
tem, but can easily be integrated. No B-frames are used in the compressed video, because of computational complexity. Furthermore, in the decoding process, delays will be introduced, because B-frames can only be decoded after the arrival of the following I- or P-frame. Especially for very low-frame-rate streams this is not desired. Since we are using only I- and P-frames, the options for frame-dropping are limited as explained in Subsection 2.3.2. Re-quantizers could be implemented, but they degrade the spatial quality of the video, which is not desired.

For the video compression process, an MPEG-4 based compression codec was used. More details on the encoding process are available in Appendix A. For each video sequence (channel) that was used, multiple compressed streams were created, each with a different frame-rate. The original video stream has 25 frames per second. Streams of 1, 5, 12 and 25 frames-per-second (fps) were created.
Chapter 3

Server Architecture

3.1 Introduction

In this chapter the proposed system architecture is described. First, the overall architecture will be discussed. Then, each module will be discussed in more detail.

The overall architecture is presented in Figure 3.1. Note that no encoders are present in the architecture. As already explained, and shown in Figure 1.1, both real-time encoded video and pre-recorded video can be transmitted by the streaming server. The selection between real-time and stored video is assumed to have been made. Therefore, all video streams are assumed available at the source reader module. The source reader module accepts compressed video frames from the source and sends them to the bit-rate scaler module. As proposed approach in Subsection 2.4, no bit-rate scaling module will be used in our implementation. The architecture however, supports this possibility. Especially when only a strictly limited set of compressed streams is available for a video channel (what can be the case with pre-recorded video), the use of a bit-rate scaler improves the bit-rate scalability of this video channel.

In the bit-rate scaler module, each source stream can be scaled to one or multiple extra streams with different bit-rates. Each scaler in the module out-
puts the original stream plus zero or more compressed streams that are scaled in bit-rate. The bit-rate of each of the finally available video streams is measured and is sent to the controller module.

The dispatcher module forwards all compressed frames of the requested video streams to the network module. The controller module tells the dispatcher which streams should be forwarded to the network module, and is responsible for the validity of the selected streams. When selected streams for a user become unusable, a new set of streams is selected in the stream-selection module.

Frames that are selected for transmission are sent to the network module. Here, they are transmitted over a network link to the corresponding user. Furthermore, the network module continuously measures the available network bandwidth of each connected user and informs the controller module.

An implementation of the proposed architecture has been made in software. This system implementation will be discussed in Section 3.3.

3.2 Modules

A more detailed description will now be given for each of the mentioned modules of the proposed server architecture as in Figure 3.1.

3.2.1 Source reader module

This module receives compressed video frames from the source. This source can be a real-time encoder or a storage device that contains pre-recorded video. As already explained in Chapter 2, the system approach with multiple encoders is selected, so for each channel, multiple compressed streams are available. In the case pre-recorded video is transmitted, this total set of streams per channel might be limited, because not all streams have been stored. The set of streams sent out by the source reader module is the total set of streams for all the requested channels, and can be a combination of real-time encoded and pre-recorded video. Frames are forwarded to the bit-rate scaler module at the same rate in which they should be decoded.

3.2.2 Bit-rate scaler module

Compressed frames for all compressed video streams are received from the source reader module. Each compressed stream can be scaled in bit-rate. There are various ways in which this can be done, as discussed in Chapter 2. We assume the settings of the available scalers in the scaler module cannot be dynamically. During the scaling process, the bit-rate, GOP-length and the frame-rate are measured and communicated to the controller module, which uses this information to select the appropriate streams.

3.2.3 Dispatcher module

This module consists of one main dispatcher and a client dispatcher for each connected user. The main dispatcher accepts compressed frames from the bit-rate scaler module. For each arriving frame, the main dispatcher asks each client dispatcher if the frame is requested. If so, the frame is forwarded to the client
dispatcher. The client dispatcher on his turn forwards the frame to the network module, where the frame is transported to the corresponding user.

When the controller tells the a client dispatcher to switch for a channel from one stream to another, the dispatcher waits with forwarding compressed frames until an I-frame has arrived for this new stream, since other frames for this stream received in the meanwhile cannot be correctly decoded.

3.2.4 Controller module

Like the dispatcher module, the controller module contains one main controller and a client controller for each connected user. Each client controller can communicate with its corresponding client dispatcher. When a new user connects to the system, a client controller and dispatcher will be added.

The main controller receives information from the bit-rate scaler module concerning the bit-rate, GOP-length and frame-rate of each available compressed video stream. If any of this information changes, each client controller is informed and checks if the currently transmitted combination of streams is still valid (the constraints for validity will be explained in Chapter 4). If the set of stream is not valid anymore, the corresponding client controller sends a request to the stream-selection module, that calculates a new set of streams for the set of requested video channels.

Information regarding the available bandwidth of network connection for each connected user is received from the network module. Using these measures and a possible maximum transmission bit-rate, the main controller tells each client controller how much network bandwidth can be used.

3.2.5 Stream-selection module

The main emphasis of our work was on the design of this module. This module selects a suitable set of streams satisfying all requirements. A stream-selection request is done by one of the client controllers. The stream-selection process is applied for only one client controller (user). Multiple requests (for multiple client controllers) are handled subsequently.

Required information regarding the bit-rate, GOP-length and frame-rate of each available compressed video stream is received from the controller at each request for a new set of streams. How stream selection can be applied, what methods can be used and what parameters are important will be discussed in detail in Chapter 4.

3.2.6 Network module

This module comprises two tasks. First, it measures the available link capacity for each connected user and communicates this to the controller module. Second, it receives compressed video frames from the dispatcher and transmits them to the connected user over the corresponding network link. If the information on the available network bandwidth sent to the controller module is correct, the frames that arrive can be sent to the connected user without any form of congestion. Since this information is never 100 percent accurate and the system needs some time to process the information, it can happen that sometimes more frames are sent over the network link than the current network status allows.
Therefore, some buffering should be applied. Because the main emphasis of our work was not on network transmission, we assume that frames that arrive at the module can also be transmitted over the link and the size of the buffers used is sufficient to overcome small deviations in the network bandwidth. More aspects of network transmission of streaming video can be found in Appendix B.
3.3 System implementation

To test the performance of the proposed architecture and algorithms for selection, a software system was built. As already discussed in the introduction (Section 3.1), the bit-rate scaler module was not implemented. The architecture as implemented is visualized in Figure 3.2.

![Figure 3.2: Architecture of the system implementation.](image)

3.3.1 Modules

The modules previously discussed in Section 3.2 have all been implemented in software. The network module was not implemented because actual network transmission was not considered. Each module will be shortly addressed and implementation specific issues are discussed. Each module was implemented in a separate thread in software.

**Source reader**

In the source reader module (as introduced in Subsection 3.2.1), video frames are read from disk. Real-time encoders were not used, since they do not improve the performance and only cause computational overhead, which is not desired, while testing the performance of the rest of our system. Video files that are read from harddisk can either be in the Microsoft AVI file format, or specified in a text-file. Each video file represents a video channel and consists of multiple compressed video streams. The reading of multi-stream AVI files causes quite some overhead in the reading process from harddisk. Therefore, when using multiple channels, the data needed for the representation of the video streams in the system, was stored in text files.

All used video streams have a constant GOP-length. For details on the applied encoding process, see Appendix A. When reading is started simultaneously for all channels, all streams with the same frame-rate will produce I-frames simultaneously. This causes peaks in the transmission rate, and will lead to high transmission delays. Another disadvantage is that the switch-delay for stream-selection (as will be introduced in Subsection 4.2.2) will be exactly the same for equal frame-rate streams of the requested channels. This means that no matter which channel will be switched, the switch-delay will be equal. To avoid
these disadvantages, we propose to start reading video streams with a random time-offset per video channel. To maximize the distribution of I-frames over the total set of channels, a random offset between zero and the GOP-duration is used. The GOP-duration is the time between the first and last frame of all frames in a GOP-structure. Note that the GOP-duration is equal for all used video channels.

**Bit-rate scaler**

The bit-rate scaler module has not been implemented, because streams at multiple frame-rates were used as input. The available set of streams has shown to give significant scalability in bit-rate when transmitting multiple channels simultaneously.

**Dispatcher**

One composite dispatcher is available, that accepts video frames from the source reader module. For each connected user, a client dispatcher object is created. At the arrival of a video frame, the composite controller asks all dispatchers if the frame is required. Each client dispatcher asks its connected client controller if the frame is requested. If not, the frame is ignored. Else, the frame should be forwarded to the network module. Because network transmission is not implemented, the dispatcher doesn't have to forward any frame. To see how the system behaves without the actual transmission of video data, video can be decoded in the dispatcher module, and visualized on the screen.

**Controller**

As in the dispatcher module, the controller module contains one composite controller and a client controller for each connected user. The composite controller accepts video stream information from the source reader and updates the video stream information. Each controller handles all tasks for the connected user. The composite controller does not have any user-specific tasks.

Because there is no network module that specifies the available bandwidth, this value can be manually specified, or read from a script file. When multiple users are connected simultaneously, the composite controller currently divides the network bandwidth amongst the users. Each client controller then works independently, based on this target bandwidth. When the client controller wants a new set of streams, because the previous set is not valid anymore, a message is sent to the stream-selection module, that calculates a new stream-combination and forwards it to the requesting client controller.

**Stream selection**

The stream-selection module calculates a new set of streams, for each client controller that requests one. One message queue is available, so multiple requests will be handled subsequently. Once a stream-combination has been calculated, it is returned to the appropriate client controller.
3.3.2 Usage

The system can be used in two ways: either by manually selecting all properties or by automatically doing everything by the use of a script file. The most important parameters can be changed at any time in the system by simply adding a line to the script file. Parameters that can be changed by the script are e.g. the network bandwidth, users decoder power, priorities of the channels and the used stream-selection method. During the operation of the script, the GUI will be disabled. When no script file is used, only the available network bandwidth can be changed in the GUI during the operation of the system. See Figure 3.3 for a graphical representation of the GUI. When the decoding of video is applied in the dispatcher module, all decoded channels are visualized on the screen (because CIF resolution was used, and no spatial scaling support was implemented, only six video channels can be represented).

![Graphical user interface (GUI) of the system implementation.](image)

**Figure 3.3:** Graphical user interface (GUI) of the system implementation.

3.3.3 Software details

The software was built in the C++ programming language for the Microsoft Windows operating system. Documentation of the source code was created using Doxygen [25]. Most code was built using ANSI C++, and the amount of Microsoft-specific code was kept to a minimum. The graphical user interface (GUI) was built using Microsoft Foundation Class (MFC).
Chapter 4

Stream Selection

4.1 Introduction

The considered system supports multiple video channels. For each video channel a limited set of compressed video streams is available, each at a different frame-rate. Each connected user requests a sub-set of video channels. The system has to select one video stream for each requested channel. This set of selected streams is only valid for the appropriate user, so for each connected user, stream-selection is done separately. The total bit-rate of the video streams must remain below the available network bandwidth to make sure the data can be delivered over the network-link. The system should rapidly adjust to changes in the available network bandwidth.

The total solution space can be represented by a tree, where each level in the tree represents a video channel and branches in each level represent the available video streams for that channel. The number of levels in the tree corresponds to the number of video channels requested by the user. Note that the branches for channel two (see Figure 4.1) represent all video streams for channel two, for each video-stream from channel 1 (so four times). A path through the tree (from the root to a leaf) represents a combination of video streams (one stream for each channel), which will be called a stream-combination. The total set of leafs represents all the different stream-combinations. Selecting a stream for each channel means selecting a leaf of the tree and thus a path through the tree.

![Figure 4.1: Tree structure for stream selection.](image)

To get a feeling for the complexity of the selection problem, consider a user requesting 16 video channels, each having three compressed video streams per channel. In case of limited network bandwidth, some channels will have to be turned off. This means no video is transmitted for these channels, which will be
called the off-stream. If this off-stream is considered a separate stream, next to the available real video streams for each channel, the total number of stream-combinations from which one has to be selected is \((3 + 1)^{16}\), which is about 4 billion. A new selection will have to be made regularly, to rapidly adapt to the changing network bandwidth. Therefore, it will be clear that a full-search on the complete set of stream-combinations is not desirable and other selection methods are required.

In the next section, parameters will be introduced, that reflect the level of optimum of a stream-combination. Section 4.3 converts these parameters into numerical cost values, to simplify comparison of multiple stream-combinations. Methods for selection of the most optimal stream-combination are introduced in Section 4.4 and their computational complexity is discussed. Finally, the frequency of stream-selection requests is discussed in Section 4.6.
4.2 Optimal stream-combination: parameters

Because only one stream-combination can be selected at any time, it is very important that this combination is the most optimal one for the situation at that time. However, to understand the definition of 'optimal', all system parameters that have influence on how optimal a stream-combination is, need to be considered.

The following parameters have been identified for each stream-combination, and will be discussed in more detail. In the next section, these parameters will be converted:

- Total bit-rate of a set of streams; the required network-link capacity to transmit the set of streams to the user.
- Required decoding power; the computational power required at the receiving side (user) to decode the total set of streams.
- The perceived video quality of the streams by the user. Higher received video quality is preferred over lower quality.
- Switch delay. The period of time that there will be no video transmitted for a video channel because another stream was selected and the system is waiting for an I-frame of the new stream for correct decoding of the video. Long periods with no received video at the user are considered perceptually annoying.
- Time since last stream-switch; the period of time passed since the last stream-switch for a specified channel occurred. A stream-switch occurs, when another stream is selected for the considered video channel, compared to the currently selected stream. Changing the frame-rate of a video channel is considered unpleasant and therefore negatively influences the perceived video quality.
- Priorities of the requested channels. A user can give priorities to the requested channels. Channels with higher priority should receive higher video quality than channels with lower priority.

4.2.1 Feasibility parameters

Parameters can be feasibility or optimum based. Feasibility parameters are hard parameters that have to meet certain constraints. For example, the total bit-rate of the video streams should always be lower than the available network bandwidth, and is thus a feasibility parameter. Optimum parameters are used to define how optimal a stream-combination is and will be discussed in the next subsubsection. In the following, the identified feasibility parameters will be addressed.

Bit-rate

All streams, part of the selected stream-combination have to be transmitted over the network-link. Therefore, the total bit-rate of the streams should be equal to or lower than the available network bandwidth. The total bit-rate of a stream-combination is defined as the sum of bit-rates of the selected streams.
Decoding power

All the compressed video that is transmitted to the user over the network-link has to be decoded. It is not useful to transmit very high-quality video streams when the streams cannot be decoded. Therefore, we defined a feasibility parameter for the amount of decoder power required to decode a stream at the receiver (the user).

The main bit-rate scalability parameter used is the frame-rate of the stream. A reduction in the number of frames per second will roughly correspond to a decrease in required decoding power with the same factor. Other parameters like the quantization used and the amount of motion in the scene will have less significant influence. Therefore, the frame-rate of the stream will be used as the (only) parameter for the required decoding power. The user has to inform the system how many frames-per-second can be decoded. This number can be changed during the operation of the system.

4.2.2 Optimum parameters

As explained previously, feasibility parameters are parameters that have to meet certain constraints. Optimum parameters on the other hand, are used to define how optimal a stream-combination is. The following parameters have been identified as optimum parameters and will be addressed shortly.

Video quality

The quality of a video stream is considered as the quality perceived by the user. Note that this is highly subjective as will be explained in Subsection 4.3.4.

Switch delay

The switch delay of a stream is defined as the period of time that no video will be transmitted for the considered video channel, after the occurrence of a stream-switch.

Because each video stream is independently compressed, P-frames of one stream cannot be decoded correctly using previously decoded frames from any other stream from the same video channel. Therefore, the system waits for the arrival of an I-frame in the new stream before transmission of this new stream is started, and can be properly decoded at the user. When the new stream has a higher bit-rate than the currently selected stream, the currently selected stream can remain being transmitted until an I-frame for the new stream arrives to avoid temporary unavailability of video at the user. When switching to a lower bit-rate stream however, this cannot be done, since this switch is applied to reduce decoding power or network bandwidth.

Consider for example a stream with a frame-rate of 1 fps and a GOP-length of 5 frames as shown in Figure 4.2. Every 5 seconds an I-frame will be encoded. When a stream-switch to this stream occurs just after the availability of an I-frame, it will take about 4 seconds before the user can start decoding the stream. In this case it will be better to keep the currently selected stream for this channel and switch another channel with a smaller switch delay.
4.2. OPTIMAL STREAM-COMBINATION: PARAMETERS

Figure 4.2: Switch Delay for stream-switch from 5 to 1 frames-per-second, using GOP-lengths of 5 frames.

**Time since last switch**

When the network bandwidth is highly fluctuating, the number of stream-switches per channel can increase, since the system will try to output the highest total video quality, and therefore, the highest total bit-rate. However, a high frequency of switches is very annoying for the connected user. Especially with high GOP-lengths and low frame-rates, high switch delays occur, and cause temporary unavailability of video at the user. Also continuous variations in frame-rate are considered unpleasant, and users prefer a constant frame-rate, even if this does not lead to an optimal bandwidth utilization.

The *time since last switch* represents the amount of time passed since the last stream-switch for the corresponding channel. Only the time of the last switch is interesting, since we can only influence the time between the last and the next stream-switch. After a certain amount of time, the last stream switch for a channel will be 'forgotten' by the user. Hence, the influence of this parameter should decrease over time.

**Priority partitioning**

When the system requests a set of stream-combinations for a connected user, the highest overall video quality possible should be selected, within the network bandwidth and decoder power constraints. To make sure the system makes a nicely balanced selection of video streams, *priorities* are introduced.

The user can specify the priority of each requested video channel. In case all priorities are the same, the system will try to match the selected video quality for all the video channels. In case some channels have higher priority, higher quality streams should be selected for these channels compared to the other channels.
4.3 Cost values

In the previous section, parameters were introduced to identify how optimal stream-combinations are. To be able to compare multiple stream-combinations, these parameters should be translated into cost values. Only optimum parameters will be used for these cost values. Any stream-combination considered in the process of stream-selection must be feasible, as already mentioned in Subsection 4.2.1. This means that all the feasibility parameters must have been met. Unfeasible combinations are ignored during the selection process.

Cost values for all the identified optimum parameters will be introduced. Also a combined cost value is proposed, enabling a straightforward comparison of several stream-combinations.

4.3.1 Bandwidth utilization cost

To give a measure of how optimal a stream-combination is, in terms of video quality, we introduce the bandwidth utilization cost. This cost value represents the amount of network utilization, defined as the ratio between the total bit-rate of the selected steams and the available network bandwidth. Thus, this cost value is defined for a stream-combination. The bandwidth utilization cost does not reflect a perfect measure for the total video quality, but in general a higher bandwidth utilization is assumed to equal higher video quality.

When for all channels the stream with highest possible frame-rate is selected (so the system transmits the highest possible bit-rate), the bandwidth utilization error should be zero, even if more network bandwidth is available. Therefore, the minimum of the network bandwidth and the highest possible bit-rate for the requested video channels is taken.

![Bandwidth utilization cost](image)

Figure 4.3: Bandwidth utilization cost.

If no video is transmitted at all, the cost value will be maximal. Consequently, any other stream-combination is preferred. Furthermore, the bandwidth utilization cost is ignored at very low bit-rates, because of inaccuracy.

The switching of streams is perceptually annoying and is therefore more important than a suboptimal bandwidth utilization. Therefore, we will consider the bandwidth utilization cost for only a percentage of the available network
4.3. COST VALUES

bandwidth. Bit-rates above this utilization percentage are assumed to have optimal utilization and therefore get cost zero. This process is shown in Figure 4.3. The value of 75% has been empirically obtained, and might be dynamically changed, depending on the level of fluctuation of the network bandwidth. For high bandwidth fluctuations, the threshold should be decreased, for a constant network bandwidth, the value should be increased. This is not considered however.

4.3.2 Switch-delay cost

The switch-delay cost is defined for each channel in a stream-combination and represents a cost value for the switch delay as explained in Subsection 4.2.2. For a set of channels (all channels in a stream-combination), the switch-delay cost is defined as the average of the switch-delay costs of the requested channels.

The proposed switch-delay cost has been empirically determined and is listed in Table 4.1 and visualized in Figure 4.4. The proposed values could be improved by doing more perceptual tests, but have proven to give satisfying results. Obviously, the function that translates switch-delay to cost is monotonically increasing, because higher delays are perceptually more annoying. For delays of less than 300 milliseconds, the cost value will be zero. This avoids the influence of processing delays that occur in the system (it takes time from the stream-selection request to the availability of the new combination). Furthermore, very low delays are assumed as not annoying. Delays up to one second are noticeable, but not very annoying (especially for very low frame-rates). Therefore, the cost value should increase only slightly. When the delay increases to two seconds, it becomes more annoying, so the cost increases further. Beyond a delay of five seconds, the cost is assumed maximal. This means that an increase between two and five seconds of delay will cause only a relatively small increase in cost. Notice that the cost value will only be used to compare multiple stream-combinations. It does not mean that delays of two and five seconds are almost equally annoying, but instead, both are very annoying compared to very low delays.

Note that for channels for which no stream is selected (the off-stream), the error will be maximal.

<table>
<thead>
<tr>
<th>Time [milliseconds]</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>t &lt; 300</td>
<td>0</td>
</tr>
<tr>
<td>300 ≤ t &lt; 1000</td>
<td>linear from 0 to 0.2</td>
</tr>
<tr>
<td>1000 ≤ t &lt; 2000</td>
<td>linear from 0.2 to 0.6</td>
</tr>
<tr>
<td>2000 ≤ t &lt; 5000</td>
<td>linear from 0.6 to 1</td>
</tr>
<tr>
<td>t ≥ 5000</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.1: Switch-delay time to switch-delay cost.
4.3.3 Time-since-last-switch cost

The time-since-last-switch cost is calculated for each channel in a stream-combination, and represents a cost value for the time since the last switch for the appropriate channel, as discussed in Subsection 4.2.2. This cost value for a stream-combination is defined as the average cost value over the channels in the combination.

The time since last switch to cost conversion function is defined as shown in Table 4.2. Figure 4.5 gives a graphical representation. When a stream-switch has just occurred for a channel, the cost is maximal. The cost linearly decreases to zero after 15 seconds, which means that after 15 seconds, a stream-switch for this channel is not assumed annoying. When the cost is not zero, other channels with lower cost are preferred to switch. The value of 15 seconds was empirically determined for scenarios where the available network bandwidth was highly fluctuating.

For a more constant network bandwidth, less stream-switches will be required. Decreasing the slope should improve the behavior of the system, since for high time-since-last-switch values, still some accuracy in the cost value is obtained. However, this gives less accuracy for situations with high fluctuations in the network bandwidth.

For the considered system, only the value of 15 seconds is considered. Dynamically adapting the threshold to the variations in network bandwidth is left for future work and may slightly improve the performance of the system.

For channels where the off-stream is selected (no video), the cost will be zero. This is done to make sure that this channel will be switched to a better stream.

<table>
<thead>
<tr>
<th>Time [milliseconds]</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t &lt; 1000$</td>
<td>1</td>
</tr>
<tr>
<td>$1000 \leq t \leq 15000$</td>
<td>linear from 0 to 1</td>
</tr>
<tr>
<td>$t &gt; 15000$</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.2: Time since last switch to cost.
4.3. COST VALUES

4.3.4 Video quality

The perceptual video quality will not be used as a cost value, but will be used for calculation of the priority partitioning cost, as will be defined next (Subsection 4.3.5). A quality value is defined for a compressed video stream. The bandwidth utilization function, as introduced in Section 4.3.1 is used as a measure for the video quality.

As explained in Section 2.4, scalability in bit-rate is obtained by creating multiple independently compressed video streams, each at a different frame-rate. Therefore, we propose to use this parameter as a measure for the perceived video quality. Now let us consider a function that converts frame-rate to video quality. It will be clear that the function has to be monotonically increasing, since more frames-per-second give smoother video and therefore a higher perceived quality. For frame-rates of 0 fps, no video is transmitted, so the quality is assumed zero. For streams of 1 fps, a quality value of 1 was defined. When the frame-rate doubles to 2 fps, the perceived improvement in quality is quite high, so the quality value is also increased. The difference in perceived quality becomes less with increasing frame-rates, so the used function should have a negative second order derivative. The difference between 2 and 5 fps is very well visible, the difference between 5 and 12 fps is already less significant and the difference between 12 and 25 fps is even less. Note that this only holds when only little motion is available in the scene, which is generally the case. The proposed frame-rate to video quality conversion function is visible in Figure 4.6. Since the set of frame-rates is limited, we use a look-up table as defined in Table 4.3 and visualized in Figure 4.6.

Since the perceived video quality is highly subjective, it is not possible to calculate an unambiguous value. We are aware that the proposed quality value will not be the best way to represent the quality of a video stream. Other representations can easily be integrated in the system though.
4.3.5 Priority partitioning cost

Priority values were introduced in Section 4.2.2. Based on these priority values, a proper partitioning of video quality over the set of requested channels should be obtained. The priority partitioning cost value represents a value for the partitioning of video quality in a stream-combination, based on the user-set priorities for each requested channel.

To calculate a value for the partitioning of the video quality over a certain stream-combination, the following parameters are defined:

\[ c \in [0, C] \] as the channel, with \( C \) the number of channels.

\[ s \in [0, S] \] as the stream, with \( S \) the number of streams.

(we assume the number of streams equal for all channels)

\[ p_c \in [0, P] \] as the priority of channel \( c \), with \( P \) the maximum priority value.

\[ q_c \in [0, Q] \] as the video quality of the selected stream for channel \( c \),

in a considered stream-combination, with \( Q \) the maximum quality value.

\[ \hat{q}_c \in [0, Q] \] as the optimal video quality for channel \( c \).

<table>
<thead>
<tr>
<th>Frame-rate [fps]</th>
<th>Video Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( 0 &lt; fps \leq 1 )</td>
<td>1</td>
</tr>
<tr>
<td>( 1 &lt; fps \leq 2 )</td>
<td>2</td>
</tr>
<tr>
<td>( 2 &lt; fps \leq 5 )</td>
<td>3</td>
</tr>
<tr>
<td>( 5 &lt; fps \leq 12 )</td>
<td>4</td>
</tr>
<tr>
<td>( 12 &lt; fps \leq 25 )</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 4.3: Frame-rate to video quality.

Figure 4.6: Video quality.
4.3. COST VALUES

The video qualities for the selected stream of each channel should be proportional to the user-set priority for each channel. This means that:

\[
\bar{q}_1 : \bar{q}_2 : \ldots : \bar{q}_n : \ldots : \bar{q}_C = p_1 : p_2 : \ldots : p_n : \ldots : p_C
\]

\[
\frac{\sum_{m \in [0,C]} \bar{q}_m}{\sum_{m \in [0,C]} p_m} = \frac{p_n}{\sum_{m \in [0,C]} p_m}
\]

\[
\bar{q}_n = p_n \cdot \frac{\sum_{m \in [0,C]} \bar{q}_m}{\sum_{m \in [0,C]} p_m} \quad \forall n \in C
\] (4.1)

Consider a stream-combination. The optimal quality values \( \bar{q}_n \) are calculated for each channel, using the exact same values. Because this is impossible, the optimal quality value for the channels are calculated, using the quality values from the selected streams in the stream-combination:

\[
\bar{q}_n = p_n \cdot \frac{\sum_{m \in [0,C]} q_m}{\sum_{m \in [0,C]} p_m}
\]

(4.2)

The total priority-partitioning error of a stream-combination is calculated as the standard deviation of the quality of the selected stream:

\[
\epsilon_{pp} = \frac{1}{C} \sqrt{\sum_{n \in [0,C]} (\bar{q}_n - q_n)^2}
\]

(4.3)

4.3.6 Combined cost

To distinguish between multiple stream-combinations, in terms of the optimal combination, a combined cost value is introduced, using the introduced cost values. To give more influence to certain values, a weighted sum can be used (see Equation 4.3.6).

The proposed weighting factors have been determined with subjective experiments and are denoted in Table 4.4. The total cost value is calculated as in Equation 4.3.6. The influence of the priority partitioning cost and the switch delay was experienced as more important over the bandwidth-utilization cost. Obtaining a lower bandwidth utilization is preferred over high switch-delays. It is also important that the selected streams are also nicely partitioned given the priorities, which cause improper balance of the quality over the channels. The time-since-last-switch cost was found least important. Because relatively low frame-rates are received for multiple channels at once, a switch in frame-rate for some channels is not experienced as very annoying, especially when the switch-delay is very low.

Typically, the user is watching all video channels simultaneously, which means the users 'scans' over the available channels. When a channel becomes more interesting than other channels, a stream-switch for this channel will be assumed annoying. The user should then simply raise the corresponding channel's priority, so a higher video quality stream will be received and the switching behavior will be less visible (think of the variation in perceived video quality with respect to the frame-rate, as introduced in Subsection 4.2.2).
CHAPTER 4. STREAM SELECTION

<table>
<thead>
<tr>
<th>Weighting Factor</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega_{\text{priority}}$</td>
<td>1.5</td>
</tr>
<tr>
<td>$\omega_{\text{bandwidth}}$</td>
<td>1.0</td>
</tr>
<tr>
<td>$\omega_{\text{switch-delay}}$</td>
<td>1.5</td>
</tr>
<tr>
<td>$\omega_{\text{TSLS}}$</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 4.4: Weighting-factor coefficients for the cost values.

$$c_{\text{total}} = \frac{1}{\sum_{n \in N} w_n} \cdot \sum_{n \in N} (w_n \cdot c_n),$$

with $N$ the number of cost values used, $w_n$ the weighting factor for cost value $n$, and $c_n$ the cost of cost value $n$. 
4.4 Methods for selection

4.4.1 Introduction

In the previous sections, multiple parameters have been identified that will be used for selection of the optimal stream-combination. The feasibility parameters (see Subsection 4.2.1) are parameters that have to be met. Optimum parameters (see Subsection 4.2.2) define the level of optimum of a stream-combination. This section will deal with methods that can be used for selection. Each time a new stream-combination is requested by the controller module, the new optimal stream-combination is calculated and returned.

In many fields of research, similar problems have been addressed. An overview of constrained-based path selection algorithms (for unicast QoS routing over networks) is given by van Mieghem et al. in [26]. When a path is constrained by multiple parameters, selection algorithms are called Multi-Constrained Path selection algorithms (MCP). The solution space are all feasible paths, thus paths of which all feasibility parameters are met. Algorithms that search for the most optimal solution within this feasible set are called Multi-Constrained Optimal Path algorithms (MCOP). The cost of the resulting path is smaller than the cost of any other path in the total set (given a combined cost function). A special case of the MCP problems is the Restricted Shortest Path (RSP) problem, where the shortest path needs to be found, satisfying only one constraint. MCP problems are NP-complete problems, so for large networks algorithms can be considered that improve the computational complexity of the algorithm. These algorithms can be based on heuristics as will be explained in the next subsections.

As mentioned in the introduction of this chapter, the solution space of the stream-selection problem in our system, can be seen by a tree where each level in the tree represents a video channel and branches from one to the next level represent available streams for that level. Selecting the optimal stream-combination is done by finding the shortest path (the path with minimal cost) through the tree. This problem is a multi-constrained optimal path (MCOP) problem, because each video stream has multiple constraint (or cost) parameters that have to be taken into account and only the optimal stream-combination should be selected.

The stream-selection algorithm should basically:

- select a stream for each requested, while
- the total bit-rate of the selected streams remains below the available network bandwidth,
- the total required decoding power of the selected streams remains below the users available decoder limit, and
- the perceived video quality at the connected user is optimal.

We already showed that the total set of stream-combinations grows exponentially with the number of channels requested. Exact calculation of the best stream-combination can only be done by considering each possible combination and is therefore not practical from a certain number of channels. Therefore, heuristic algorithms are considered, that find a near-optimal solution within a limited number of calculations. To do this, certain assumptions have to be
made. For the heuristics introduced in this section, the following assumptions were made.

- Per stream-selection request, stream-switches will only occur for a limited number of channels.
- All video channels are uncorrelated.

Based on these two assumptions, three heuristics are proposed. Next, each heuristic and its computational complexity will be discussed. Then we will explain the use of combinations of heuristics. We conclude with a discussion on the frequency of the stream-selection requests.

4.4.2 Heuristic: switch limit number of channels

We assumed that during a stream-selection request, not all channels are switchable. A channel is defined switchable when the selected stream for this channel can change. If a channel is not switchable, the currently selected stream remains selected and no other stream can be selected.

We divide the total set of channels in sub-sets of a limited number of channels, called channel-combinations. Each channel in a channel-combination can be switched, whereas all other channels are assumed constant. The feasibility parameters are used for the total set of requested channels. An example with three channels (each having three streams), of which one channel is switchable, is shown in Figure 4.7. The solid branches in the left part of the figure represent the currently selected streams and the dashed lines represent the other available streams. There are $3^3 = 27$ possible combinations. The considered stream-combinations are shown in the block in the middle of the figure. In the left, only the first channel is switchable, in the center the second channel and in the right the third channel. The number of considered stream-combinations is $3 + 3 + 3 = 9$, compared with the 27 for a full-search. The most optimal of all considered stream-combinations is returned to the controller module that requested a new stream-combination.

Division in channel-combinations

We assume that maximally $C_{set}$ channels can be switched during a stream-selection request. However, we don’t know which channels will be switched. Therefore, we process multiple channel-combinations, each containing a different sub-set of switchable channels.

Within the limitation of allowing maximally $C_{set}$ channels to be switched, the optimal stream-combination is found by considering all possible channel-combinations of $C_{set}$ channels. Less computationally intensive but also less accurate, is to simply divide the total set of channels in sets of successive channels.

The number of channel-combinations $N$ will now be considered:

- When dividing in all possible combinations of $C_{set}$ channels, the number of channel-combinations $N_{all}$ is:

$$N_{all} = \binom{C_{total}}{C_{set}}$$
4.4. METHODS FOR SELECTION

Figure 4.7: Stream-selection method: limited switch.

- For the remainder of this section, $C_{total}$ is assumed an integer times $C_{set}$.

When dividing in sub-sets of successive channels, the number of channel-combinations $N_{succ}$ is:

$$N_{succ} = \frac{C_{total}}{C_{set}}$$

The number of streams per channel is $S$, and considered equal for all channels, so the number of stream-combinations over a set of $C_{set}$ channels is:

$$S^{C_{set}}$$

Let us consider the number of stream-combinations to be processed:

- When dividing in sub-sets of all possible combinations, the total number of stream-combinations $M_{all}$ is:

$$M_{all} = N_{all} \cdot S^{C_{set}} = \left( \frac{C_{total}}{C_{set}} \right) \cdot S^{C_{set}} \quad (4.4)$$

- When dividing in sub-sets of successive channels, the total number of stream-combinations $M_{succ}$ to process is:

$$M_{succ} = N_{succ} \cdot S^{C_{set}} = \frac{C_{total}}{C_{set}} \cdot S^{C_{set}} \quad (4.5)$$

Figure 4.8 represents the number of stream-combinations (as in Equations 4.5 and 4.4) that are processed as a function the number of channels per sub-set, considering a total set of 16 channels. Both methods of dividing into sub-sets are considered. Note that the vertical axis uses a logarithmic scale.
The optimal solution is the one where we consider all channels at once, so \( C_{\text{set}} = C_{\text{total}} \), and thus perform a full-search. The least optimal solution is the one where only one channel is defined switchable, thus \( C_{\text{set}} = 1 \). As can be seen in the figure, sub-sets of 11 to 16 channels are not attractive when considering all possible channel-combinations, since the number of stream-combinations to process is much higher and the obtained solution is less optimal, compared to the full-search case (where \( C_{\text{set}} = C_{\text{total}} \)).

\[ \text{Computational complexity} \]

The number of calculations related to calculating the cost of one stream-combination is approximately proportional to the number of channels in the combination. Since all channels are considered in each stream-combinations, the cost is proportional to \( C_{\text{total}} \). The total number of calculations for both considered division methods is shown below. A visualization of Equations 4.6 and 4.7 is available in Figure 4.10. Note that the following equations are based on the worst-case scenario, and all stream-combinations are feasible. In case the network bandwidth is limited, not all stream-combinations will be feasible, and the required computation power will be lower.

- When dividing in sub-sets of successive channels, the number of computations \( X_{\text{switch, succ}} \) is:

\[
X_{\text{switch, succ}} = C_{\text{total}} \cdot M_{\text{succ}} = C_{\text{total}} \cdot \frac{C_{\text{total}}}{C_{\text{set}}} \cdot S_{\text{set}}^{C_{\text{set}}} \quad (4.6)
\]
When dividing in sub-sets of all possible combinations, the number of computations $X_{\text{switch,all}}$ is:

$$X_{\text{switch,all}} = C_{\text{total}} \cdot M_{\text{all}} = C_{\text{total}} \cdot \left( \frac{C_{\text{total}}}{C_{\text{set}}} \right) \cdot S_{\text{set}} \quad (4.7)$$

### Number of iterations

It is also possible to apply an iterative process where in each iteration, the optimal stream-combinations is calculated and stored. After a certain number of iterations, the overall optimal stream-combination will be applied. This way, more channels can be switched, but it does not guarantee that the solution found is as optimal as the solution that is found when considering more switchable channels with only one iteration.

#### 4.4.3 Heuristic: independent sub-problems

When all channels are assumed independent, stream-selection can be achieved by independently processing sub-sets of channels. The optimal stream-combination is calculated for each sub-set of channels. The feasibility costs (bit-rate and decoder power) are divided amongst the sub-sets, in ratio with the priorities for the channels. Finally, the calculated optimal stream-combinations for all the sub-sets are merged together to form the resulting stream-combination that will be used. This optimum can be calculated using a full-search, but within each sub-problem, the previously mentioned method can also be used to limit the amount of computations. An example with three channels and sub-sets of one channel is shown in Figure 4.9. As can be seen from this figure, all streams for each channel are independently considered. The best stream for each channel will be combined for the total returned combination. Compare this figure with Figure 4.7, that represents the behavior for the limited-switch method.

![Figure 4.9: Stream-selection method: independent sub-problems.](image-url)
Division in channel-combinations

As in the previously mentioned heuristic, the division in channel-combinations is not defined. We can consider all possible channel-combinations of a limited number of channels, or make a division in sets of successive channels.

Because we assume that all channels are independent, considering all possible combinations is expected to give only little improvement over a successive division. However, the network bandwidth is also divided over the sub-sets of channels. Depending on the bit-rate of the streams, considering all combinations can lead to more improvement over the successive division. Therefore, we will consider both division methods here.

Both methods have already been discussed in Subsection 4.4.2 and will not be discussed in detail here. The number of channels per channel-combination is $C_{set}$.

Computational complexity

The number of calculations for one stream-combination is approximately linear with the number of channels in the combination. Since only the channels from the channel-combination are considered in each stream-combination, the cost is proportional to $C_{set}$. The total number of calculations for both considered division methods is shown below. Figure 4.10 shows both methods from Equations 4.8 and 4.9. Moreover, it shows the methods from the previously discussed heuristic (Equations 4.6 and 4.7 from Subsection 4.4.2). Note that the following equations are based on the worst-case scenario, and all stream-combinations are feasible. In case the network bandwidth is limited, not all stream-combinations will be feasible, and the required computation power will be lower.

- When dividing in sub-sets of successive channels, the number of computations is:

$$X_{\text{sub-problem, succ}} = C_{set} \cdot M_{\text{succ}} = C_{set} \cdot \frac{C_{\text{total}}}{C_{set}} \cdot S_{C_{set}}$$  \hspace{1cm} (4.8)

- When dividing in sub-sets of all possible combinations, the number of computations is:

$$X_{\text{sub-problem, all}} = C_{set} \cdot M_{\text{all}} = C_{set} \cdot \left( \frac{C_{\text{total}}}{C_{set}} \right) \cdot S_{C_{set}}$$  \hspace{1cm} (4.9)

4.4.4 Heuristic: Pruning

The heuristic that is proposed in this subsection is also based on the assumption that all channels are independent. In each stream-selection request, the number of stream-combinations increases exponentially with the number of channels. However, after processing a sub-set of the total set of channels, stream-combinations with high probability of not being selected can be dropped. This decreases the number of computations. The process of pruning is shown in Figure 4.11. Pruning is applied after two channels, and the 16 stream-combinations are reduced to 4 combinations, before adding channel 3.
We will discuss the pruning process in more detail. We take a sub-set of $C_p$ channels from the total set of $C_{\text{total}}$ channels and process all stream-combinations. Pruning is now applied on this set of stream-combinations. The resulting stream-combinations will be stored as streams in a new meta-channel, that encompasses the $C_p$ processed channels. One pruning-pass has been completed. A pruning-pass covers all steps taken until now. Now we consider this meta-channel plus a new sub-set of $(C_p - 1)$ channels, and calculate the stream-combinations. Pruning is again applied, and the remaining stream-combinations are stored in a new meta-channel. This process is repeated until all channels have been processed.

The decision value used for the pruning decision can be any value. In our system, we used the combined cost value as introduced in Section 4.3.6.

How many combinations are pruned in each pruning-pass is not defined. The number of stream-combinations pruned can be traded with the accuracy of the found optimum. The amount of pruning can also be set to obtain a certain limit in processing power. In our system, we implemented pruning based on a percentage of the total number of stream-combinations. Hence, in the following, the amount of pruning will be based on a constant percentage. Using a dynamically calculated percentage will be better. Especially in the case when not many stream-combinations are feasible because the network bandwidth is limited, the pruning percentage should be very low.

---

Figure 4.10: Computation complexity vs. the number of channels per sub-set, for all methods.
CHAPTER 4. STREAM SELECTION

Figure 4.11: Stream-selection method: pruning.

Complexity of pruning

We will now discuss the complexity of a pruning algorithm, assuming that a percentage of the set of stream-combinations will be pruned. When \( C_p \) is the number of channels to process per pruning pass, the number of pruning passes \( X_{\text{total}} \) equals:

\[
X_{\text{total}} = \left\lceil \frac{C_{\text{total}} - C_p}{C_p - 1} \right\rceil + 1
\]

(4.10)

The number of channels to process in the last pruning pass is:

\[
C_{p, \text{last}} = C_{\text{total}} - \left( (X_{\text{total}} - (S - 1)) \cdot (S - 1) + S \right)
\]

(4.11)

The pruning percentage \( P \) is defined as the percentage of the total number of stream-combinations. Combinations with highest cost value are pruned first. When all stream-combinations are feasible, the number of remaining stream-combinations, is given by:

\[
Y = (S^{C_p} \cdot P) \cdot \prod_{(X_{\text{total}}-2)} (S^{C_p-1} \cdot P) \cdot (S^{C_{p,\text{last}}} \cdot P)
\]

(4.12)

\[
= (S^{C_p} + C_{p, \text{last}} \cdot (P^{X_{\text{total}}})) \cdot \prod_{(X_{\text{total}}-2)} (S^{C_p-1})
\]

(4.13)

To give an idea of the reduction of stream-combinations, consider 16 channels, each having 4 streams. When applying a pruning percentage of 50% after sets of 4 channels, the total set of 4.3 billion combinations is reduced to 1.2 million. Increasing this percentage to 80%, reduces the number to roughly 10 thousand.

The total number of stream-combinations decreases enormously. For relatively high number of channels, the remaining set of stream-combinations will still be very high. Therefore, it is preferred not to use pruning as the only method for reducing computational complexity.
4.5 Combinations of heuristics

Three different heuristic algorithms have been discussed. Each heuristic can be used separately, but to reduce the number of computations even more, combinations of heuristics can be used. For example, when the total set of channels is divided in independent sub-problems, pruning can be applied on each sub-set, or the limited switching heuristic can be applied to each sub-set. In the discussion of the results in Chapter 5 we did not consider combinations of heuristics.

4.6 Frequency of stream-selection

So far, only the stream-selection process itself has been discussed. The frequency of selection requests has not been mentioned. When the network bandwidth decreases, the selected stream-combination might exceed the bandwidth budget and a new stream-combination has to be calculated.

The currently selected stream-combination is valid, as long as the feasibility parameters are met. The total required decoder power of the streams should remain below the users decoder limit. The decoder power of each stream and the users decoder power are assumed constant. Therefore, the only constraint that has to be taken into account is the total bit-rate of transmitted video. The available network bandwidth is constantly varying and is therefore an important parameter for the stream-selection process.

We propose to start the stream-selection algorithm maximally once per specified time interval, but only when the total bit-rate of transmitted video exceeds the available network bandwidth, or the bandwidth utilization is very low. We propose a time interval of 500 milliseconds between each stream-selection request, which has been found suitable.

4.7 Conclusions

In this chapter parameters were introduced that influence the level of optimum of a stream-combination. These parameters were converted to numerical cost values by proposed conversion functions. Using these cost functions, the difference in level of optimum for multiple stream-combinations can be expressed. Next methods for stream-selection using these cost values were proposed.

Next, to considering all possible stream-combinations in a full-search, heuristics were introduced, based on certain assumptions. The first heuristic introduced was the limited-switch method (see Section 4.4.2) that allows only a limited number of channels to switch per iteration. The second heuristic is based on the fact that all channels are independent (see Section 4.4.3). Therefore, the total set of channels if divided in sub-sets of channels that will be independently processed. Furthermore, pruning was introduced (see Section 4.4.4). After considering all combinations using a sub-set of channels, the least optimal combinations are pruned and calculation continues with the next sub-set of channels. The computational complexity of each method was discussed.
Chapter 5

Results

5.1 Introduction

In this chapter the performance of the designed system will be discussed. Multiple stream-selection algorithms are compared in terms of performance and computational complexity. Four aspects of the stream-selection methods will be discussed in detail.

- The performance of a number of stream-selection methods, compared to their computational complexity.
- The performance of a number of stream-selection methods, compared to their bit-rates of the considered stream-combinations.
- The influence of the user-set priorities.
- The influence of using video streams with variable bit-rate (VBR) or constant bit-rate (CBR).

System used for simulations

The simulations of the software implementation were run on an HP notebook computer, equipped with a Pentium 4m processor at 1.7 GHz, a front-side-bus (FSB) at 400 MHz, and DDR-RAM memory at 266 MHz. The tests were performed using the software implementation as discussed in Section 3.3, and run under the Microsoft Windows XP operating system.
5.2 Performance vs. computational complexity

When deciding which stream-selection method to use for our DVR system, a trade-off has to be made between performance and computational complexity. This section compares various stream-selection methods with respect to the required computation power and the obtained performance. First, the terms computational complexity and performance will be discussed. Then, the test-scenario will be introduced and the considered selection methods are listed. Finally, the results will be discussed.

Computational complexity

The computational complexity value represents the percentage of CPU-time used by the stream-selection method. Only the processing time of the stream-selection thread was measured, since the rest of the system behaves the same for each selection method. Measurements of the used CPU-time are done every 250 milliseconds. Internally, values are obtained in milliseconds. The presented value is an average over all measurements. All values are in the range \([0, 100]\).

Even if the presented computational complexity values are low, peaks in CPU-usage can have occurred. As can be seen in the results (see Table 5.1), some stream-selection methods seem to use zero CPU-time, which means that the measurements were not accurate enough. It would be more accurate to use the total duration of the test as the measurement interval, and measure the milliseconds of CPU-time over the complete interval. However, the Microsoft Windows Performance Data Helper interface (PDH) was used. PDH averages the CPU-time used over an internally specified interval, and thus reflects a momentary value for the CPU-usage. Therefore, measurements over a very long time-interval seem not possible and are not very accurate. Because this was noticed at the very end of the project, time was limited to look for an alternative measurement method.

Performance

All stream-selection algorithms distinguish between multiple stream-combinations by the combined optimum cost value as introduced in Subsection 4.3.6. Therefore, we will use this combined cost value as a metric for the performance of a selection method. Each stream-selection request returns the cost of the selected stream-combination. The value used to represent the performance of a stream-selection method, is the average over all returned values. Values are in the range of \([0, 10000]\). In practice, these values are below 2000.

Note that the used performance value cannot explain how good each method performs in terms of a specific behavior (for example the number of switches or the total switch-delay). It can only be used to distinguish between multiple selection methods.

Scenario

Six video channels were used, each having four compressed video streams. The frame-rates of the streams are 1, 5, 12 and 25 frames-per-second. Some of the used video channels have high variations in bit-rate (due to motion), others
can be considered constant bit-rate. Because of the high CPU-usage for the full-search method, the number of channels was limited to six for all tests.

As can be seen in the results (see Table 5.1), the average CPU-load for the used test scenario is only 19%. However, the peak CPU-load was approximately 100% (not listed in the table). Due to the exponential character of the full-search, adding another channel would endanger the real-time performance of the system for the full-search method.

Furthermore, because only feasible stream-combinations are considered, the presented computational complexity values do not reflect the worst-case behavior of the methods, when the network bandwidth is unlimited. Tests show however, that the difference in CPU-usage is not much different between the used bandwidth variation and the scenario with unlimited bandwidth.

The network bandwidth will be varied between 0 and 5 MBit/sec. The exact behavior is shown in Figure 5.1. This network behavior was used for the tests, because it contains a slow increase, decrease and various variations. The users decoder power limit was set to maximal, so stream-combinations will be considered not feasible only if the total bit-rate is higher than the network bandwidth. The priorities of the channels are all equal, so streams with equal frame-rates are desired for the total set of channels.

Figure 5.1: Network bandwidth over time.

The random reading offset in the Source Reader module as discussed in Subsection 3.3.1 was adjusted to produce the exact same data for each test (semi-random). The network bandwidth behavior was stored in a script file, so this
behavior was the same for each test. Each test was conducted three times and the obtained performance and complexity values were averaged.

Considered stream-selection methods

Multiple stream-selection methods have been tested, 19 in total. The used methods will be shortly addressed. The measured performance and computational complexity values as measured are shown in Table 5.1 and visualized in Figure 5.2 and Figure 5.3.

- Methods *Sub-Prob* (methods 1 to 3) are the methods of division in independent sub-problems, as discussed in Subsection 4.4.3. The method used for division in sub-sets of channels, is the division in sub-sets of subsequent channels as defined in Subsection 4.4.2. Methods 1, 2 and 3 divide the total set of channels sub-problems of respectively 1, 2 and 3 channels.

- Methods *Lim-Switch* (methods 4 to 9) are the methods where only a limited number of channels is assumed switchable as discussed in Subsection 4.4.2. The method used for division in sub-sets of channels, is the division in all possible sub-sets of channels as defined in Subsection 4.4.2. Methods 4, 5 and 6 apply one iteration pass and define respectively 1, 2 and 3 channels switchable. Methods 7 to 9 apply the same switching behavior, but apply two iteration passes per stream-selection request.

- Method *Full-Search* (method 10) is the method that considers all feasible stream-combinations and can be used as a reference for the other methods.

- Methods *Prune* (methods 11 to 19) are the methods where during the full-search certain stream-combinations are dropped, as discussed in Subsection 4.4.2. Methods 11 to 13 apply pruning at different percentages after every two channels. Methods 14 to 19 are mixed and apply different pruning percentages after every two, three or four channels. The pruning percentages used are 50%, 75% and 85%.

Before discussing the results, we will first address the abbreviations used in Table 5.1. *Iter.* represents the number of iterations used. *Sub-Ch.* represents the number of channels per sub-set, *Perf.* represents the performance and *CPU %* the percentage of CPU-time used.
Results

As can be seen in Table 5.1 or in Figure 5.2 (or zoomed-in in Figure 5.3), there is quite a difference in performance of the used methods. As can be seen in Table 5.1, the measured CPU-time is very inaccurate and the values should therefore only be used as an indication of the required power and should not be numerically compared.

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Method</th>
<th>Iter.</th>
<th>Sub-Ch.</th>
<th>Perf.</th>
<th>CPU %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sub-Prob</td>
<td>1</td>
<td>1</td>
<td>1219</td>
<td>0.000</td>
</tr>
<tr>
<td>2</td>
<td>Sub-Prob</td>
<td>1</td>
<td>2</td>
<td>927</td>
<td>0.000</td>
</tr>
<tr>
<td>3</td>
<td>Sub-Prob</td>
<td>1</td>
<td>3</td>
<td>856</td>
<td>0.000</td>
</tr>
<tr>
<td>4</td>
<td>Lim-Switch</td>
<td>1</td>
<td>1</td>
<td>831</td>
<td>0.000</td>
</tr>
<tr>
<td>5</td>
<td>Lim-Switch</td>
<td>1</td>
<td>2</td>
<td>625</td>
<td>0.000</td>
</tr>
<tr>
<td>6</td>
<td>Lim-Switch</td>
<td>1</td>
<td>3</td>
<td>666</td>
<td>0.004</td>
</tr>
<tr>
<td>7</td>
<td>Lim-Switch</td>
<td>2</td>
<td>1</td>
<td>770</td>
<td>0.000</td>
</tr>
<tr>
<td>8</td>
<td>Lim-Switch</td>
<td>2</td>
<td>2</td>
<td>670</td>
<td>0.002</td>
</tr>
<tr>
<td>9</td>
<td>Lim-Switch</td>
<td>2</td>
<td>3</td>
<td>674</td>
<td>0.880</td>
</tr>
<tr>
<td>10</td>
<td>Full-Search</td>
<td>1</td>
<td>6</td>
<td>641</td>
<td>19.300</td>
</tr>
<tr>
<td>11</td>
<td>Prune 50%</td>
<td>1</td>
<td>2</td>
<td>658</td>
<td>0.430</td>
</tr>
<tr>
<td>12</td>
<td>Prune 50%</td>
<td>1</td>
<td>3</td>
<td>674</td>
<td>2.230</td>
</tr>
<tr>
<td>13</td>
<td>Prune 50%</td>
<td>1</td>
<td>4</td>
<td>706</td>
<td>4.500</td>
</tr>
<tr>
<td>14</td>
<td>Prune 75%</td>
<td>1</td>
<td>4</td>
<td>882</td>
<td>0.130</td>
</tr>
<tr>
<td>15</td>
<td>Prune 85%</td>
<td>1</td>
<td>4</td>
<td>880</td>
<td>0.110</td>
</tr>
<tr>
<td>16</td>
<td>Prune 75%</td>
<td>1</td>
<td>2</td>
<td>764</td>
<td>0.043</td>
</tr>
<tr>
<td>17</td>
<td>Prune 85%</td>
<td>1</td>
<td>2</td>
<td>1039</td>
<td>0.003</td>
</tr>
<tr>
<td>18</td>
<td>Prune 75%</td>
<td>1</td>
<td>3</td>
<td>673</td>
<td>0.450</td>
</tr>
<tr>
<td>19</td>
<td>Prune 85%</td>
<td>1</td>
<td>3</td>
<td>663</td>
<td>0.200</td>
</tr>
</tbody>
</table>

Table 5.1: Performance vs. computational complexity for multiple methods.

First, note that the performance value of the full-search method is not the overall optimal value, would would be expected, since all possible combinations are considered. This can be explained because the optimum stream-combination at each stream-selection request is dependent on the previously selected stream-combination. Considering the global optimal stream-combination for each selection request can therefore lead to an overall less optimal behavior as is the case for the considered scenario. This was verified by doing the same test with other video channels (with other bit-rate behavior).

Methods 1, 2 and 3 use the sub-problem heuristic. Because the feasibility costs are divided amongst the sub-sets of channels, overall bit-rate usage is not optimal and therefore the performance is not very well. When considering a higher number of video channels (e.g. 16 channels), division in a strictly limited number of sub-sets (e.g. two sub-sets of 8 channels)) however, this method will give a descent performance. It will have to be combined with another heuristic algorithm, because a full-search on 8 channels is very expensive.

The limited-switch methods (methods 4 to 9) show that for the used six channels, switching more than two channels is not effective. Defining two channels switchable seems more effective than considering only one channel. Adding a second iteration does not improve the results much. When comparing methods
5 and 7, it can be seen that allowing two iterations of one switchable channel is less optimal than considering one iteration with two switchable channels.

The pruning methods (methods 11 to 19) perform quite well. Comparing the computational complexity with the limited-switch methods however, shows that this method is probably not usefull to use as the only method, to reduce required computation power.

![Performance vs. computational complexity](image)

**Figure 5.2: Computational cost vs. optimum cost.**

Figure 5.4 shows the performance of the system in terms of network utilization and switching behavior. In all sub-figures, the horizontal axis represents the time in milliseconds. The first sub-figure shows the bandwidth utilization. Both the available network bandwidth and the total transmitted bit-rate are displayed. The other sub-figures represent the switching behavior of each requested channel. Values on the vertical axis represent the selected streams. A value of -1 means that no video is transmitted for that channel. Values 0, 1, 2 and 3 represent streams with frame-rates of respectively 1, 5, 12 and 25 fps.

We considered the heuristic where only two channels are allowed to switch simultaneously (method 5 as in Table 5.1). The influence of the bandwidth utilization cost (as discussed in Subsection 4.3.1) can be seen. When the total bit-rate of a stream-combination is above 75% of the available network bandwidth, this cost is zero. Therefore, only the other cost values have influence on the optimum stream-combination. This can be seen in the limited network utilization, but also in the limited number of stream-switches. Between approximately 125 and 140 seconds, the variations in the network bandwidth are high. In this case, especially channel 5 is often switched. Therefore, it might be better to reduce the 75% threshold of the bandwidth utilization cost. It is probably better to decrease the weighting factor of the bandwidth utiliz-
5.2. PERFORMANCE VS. COMPUTATIONAL COMPLEXITY

Figure 5.3: Computational cost vs. optimum cost (zoomed).

The performance cost as introduced in Subsection 4.3.6. The used weighting factor for the time-since-last-switch cost is very low and could be increased to improve the switching performance.
Figure 5.4: Bandwidth usage & selected streams with VBR video (using selection method 5).
5.3 Scalability vs. optimality cost

At each stream-selection request, the used selection method determines the best stream-combination from the total set of considered stream-combinations. Depending on the stream-selection method used, a different set of stream-combinations is considered.

The sets of stream-combinations are compared for the used selection methods at one moment in time. This is done in terms of bit-rate and optimal parameter cost values. The moment considered is at 65 seconds after the simulation start (as in Figure 5.1). Network bandwidth is relatively high (about 650 kbytes/sec), so a high number of combinations is feasible. The decoder limit was set to infinity for this test.

The set of stream-combinations is visualized for each selection method. For each stream-combination in the set, a data point was plotted for each considered stream-combination, with corresponding total bit-rate and optimal cost value (as discussed in Subsection 4.3.6). Results are visualized in Figures 5.5, 5.6, 5.7 and 5.8. Note that since this test only considers one specific moment in time, the results cannot be generalized for the overall performance of the system over a longer period of time.

For the methods that divide in sub-problems (methods 1, 2 and 3), each sub-problem is solved independently. Both the optimal cost value and the bit-rate only represent a sub-set of channels and cannot straightforwardly be compared with the other methods. When sub-sets of only one channel are considered, the bit-rate is also divided over all six channels. Therefore, the bit-rate is approximately one-sixth of the total bit-rate. Because this does tell nothing about the overall performance of the stream-selection method used, the results for the sub-problem are not visualized, nor discussed.

Figure 5.5 shows the behavior of the limited-switch method (methods 4 to 9) for both one and two iterations. When only one channel is defined switchable, the considered stream-combinations contains some combinations with high bit-rate and low optimal cost. When two or three channels are defined switchable, there is no visible improvement in either bit-rate or optimal cost. It seems that increasing the number of switchable channels does not really increase the performance. Increasing the number of iterations also does not seem to improve the performance. The results from Section 5.2 however, show that increasing the number of channels does give a small increase in performance, as does the number of iterations.

Figure 5.6 shows pruning on the total set of stream-combinations, applying pruning after every two channels. As can be seen in the figure, the total set of remaining stream-combinations is quite optimal, since only the stream-combinations with low optimal cost and high bit-rate remain. When applying a pruning percentage of 85%, only two stream-combinations are finally considered. Both however, are quite optimal. When considering the average optimal cost values as introduced in Section 5.2, pruning after two channels shows to be not as optimal.

When applying pruning after every three channels, the stream-combinations as in Figure 5.7 are considered. Compared to pruning after two channels as in Figure 5.6, the remaining stream-combinations are not much more optimal. The same goes for pruning after four channels as in Figure 5.8. This shows that pruning can be applied as desired. The combination of pruning percentage
and the number of channels to consider for pruning can be changed at will to obtain the desired required computation power. The results from Section 5.2 however, show that increasing the number of channels after which to apply pruning does not straightforwardly give an increase in performance. This can be simply explained by the effect of using a limited set of channels. When pruning after every two channels, five pruning passes are made. When pruning after three channels only three passes are made and when pruning after four channels only two. Comparing pruning after different numbers of channels should therefore only be done when using a higher number of total video channels (e.g. 16).

Pruning seems a useful method to reduce the number of computations, and still obtaining decent performance. The pruning percentage however should not be too high, especially when pruning after a small number of channels. When considering limited network bandwidth, pruning with a dedicated pruning percentage is not desired, since quite stream-combinations will be pruned. It might be better to dynamically adjust the pruning percentage to obtain a set with defined total number of stream-combinations, that thus requires a predefined amount of computation power to process. The pruning percentage might by dynamically be adjusted, depending on the available network bandwidth, as already discussed in Section 4.4.4.
5.3. SCALABILITY VS. OPTIMALITY COST

Figure 5.5: Optimal cost vs. bit-rate: limited-switch selection methods.
Figure 5.6: Optimal cost vs. bit-rate: pruning after 2 channels, multiple percentages.
5.3. SCALABILITY VS. OPTIMALITY COST

Figure 5.7: Optimal cost vs. bit-rate: pruning after 3 channels, multiple percentages.
CHAPTER 5. RESULTS

Figure 5.8: Optimal cost vs. bit-rate: pruning after 4 channels, multiple percentages.
5.4 Priorities

When a user wants to receive higher video quality for one video channel, compared to other requested channels, the priority of this channel can be increased. To show that the system automatically adapts to changes in the set priorities, the same test as before was conducted. To see the influence of the priorities, the priorities of channels 0 and 1 were increased from one to three after 85 seconds. The priorities of the other channels remained constant. Stream-selection method 5 was used (see Table 5.1).

As can be seen in Figure 5.9, both channels 0 and 1 receive higher quality video streams, almost immediately after the change in priority. Compare this figure with Figure 5.4.

5.5 Constant bit-rate vs. variable bit-rate video

The switching behavior for variable bit-rate (VBR) video can be different from the behavior for constant bit-rate (CBR) video. Because a stream-combination might become invalid when the bit-rate of a VBR video-stream increases, at least one channel will have to be switched to meet the feasibility constraints.

By the set threshold in the bandwidth utilization cost, the switching behavior is already limited. The system behaves almost the same for both constant and variable bit-rate video streams. Compare the switching behavior using VBR streams in 5.4 with the behavior using CBR streams in Figure 5.10.
Figure 5.9: Bandwidth usage & selected streams with changing priorities (using selection method 5).
Figure 5.10: Bandwidth usage & selected streams with CBR video (using selection method 5).
Chapter 6

Conclusions

An architecture for a streaming server module for a surveillance DVR system is proposed. The architecture allows for multiple users to connect and view video simultaneously. Each user can select multiple video channels. The system automatically adapts the bit-rate of the transmitted video to the available network bandwidth.

To obtain bit-rate scalability, a limited set of compressed video streams is available for each video channel, each on a different frame-rate. These streams can either have a constant or a variable bit-rate. The system automatically selects the optimal set of streams for the requested set of video channels, for all connected users. By giving higher priority to channels that require higher video quality, the system automatically adapts the quality of the transmitted video for all channels.

To determine how optimal the transmission of a set of video streams is, parameters and cost values have been defined. Each cost value represents a specific parameter of the system behavior. To obtain one cost value for a set of streams, a weighted sum is used. By adjusting the weighting factors of the used cost values, the influence of each parameter can be modified.

Multiple heuristic algorithms for stream-selection have been proposed and their performance has been compared. The independent sub-set method divides the total set of channels in sub-sets and processes each sub-set independently. This reduces the computational complexity, but its performance is poor. The limited-switch method allows only a limited set of channels to switch streams. This method seems a good method that gives decent performance for a strictly limited required computation power. Pruning gives good results compared to the full-search, but does not use the assumption that the number of stream-switches should be strictly limited, and therefore requires much more computation power. Overall, the limited-switch method is considered the best method for stream-selection, because the computational load is reduced to a factor of the load for a full-search, while maintaining similar performance.
The results are demonstrated in real-time on a software implementation of the proposed system.
Chapter 7

Recommendations

To make sure the system operates well under totally different network conditions, the values of the weighting factors should be dynamically adjusted. For a very bursty wireless network with high variations in the bandwidth, the time-since-last switch cost should be considered more important than the bandwidth utilization cost.

Subjective tests should give a more detailed overview of what system parameters are assumed perceptually more important. Using these results, the parameter-to-cost conversion functions (as introduced in Section 4.3) can be adjusted to give an overall better performance.

The influence of video channels with a strictly limited set of compressed streams should be considered, since this will be the case when for certain channels, pre-recorded video is requested. The time-since-last-switch cost could be found more important for channels with more compressed streams.

The limited-switch method considers all possible channel-combinations of a limited number of switchable channels. Since this requires quite some computation power for a high number of video channels, the method should be extended to also consider sub-sets of subsequent switchable channels.

The pruning behavior at low network bandwidth is not considered. In this case it will be much better to use a lower pruning percentage since only very few stream-combinations are feasible. Therefore, the pruning percentage should be dynamically adjusted to obtain a constant required computation power.

We showed that all discussed algorithms consider a sub-set of stream-combinations, that still contains the most optimal combinations. These tests, however, are done for a fixed network bandwidth, and don't give a good idea of the scalability in bit-rate over the complete bit-rate spectrum. To obtain a better view of the system behavior for the complete spectrum of bit-rates, more tests should be done, each at a different target network bandwidth.
The purpose of the stream-selection algorithms is to make sure that the selected video can be transmitted over a network connection. The actual network module was never implemented however. This will be the best way to check if the system really works.
**Glossary**

<table>
<thead>
<tr>
<th>Term</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel-combinations</td>
<td>A set of channels. The number of channels in the set is smaller than the total set of requested channels and is used for stream-selection algorithms as in Section 4.4.</td>
</tr>
<tr>
<td>Digital Video Recorder (DVR)</td>
<td>A device that supports recording of multiple video channels. Display of real-time encoded or pre-recorded video is possible on the local or remote via a network connection.</td>
</tr>
<tr>
<td>End-to-end delay</td>
<td>The amount of time between the capture of video data at the camera and the display of video at the connected user.</td>
</tr>
<tr>
<td>Off-stream</td>
<td>The situation in which no compressed video stream is selected for the considered video channel.</td>
</tr>
<tr>
<td>Stream-combination</td>
<td>A stream-combination represents the selected compressed video streams for the total set of video channels.</td>
</tr>
<tr>
<td>Stream selection</td>
<td>The process where for each requested video channel a compressed video stream is selected for transmission.</td>
</tr>
<tr>
<td>Stream-switch</td>
<td>A stream-switch occurs, when another stream is selected for the considered video channel, compared to the currently selected stream.</td>
</tr>
<tr>
<td>Switchable channel</td>
<td>A video channel is defined switchable, when the currently selected stream for this channel can be switched during a stream-selection request.</td>
</tr>
<tr>
<td>Term</td>
<td>Explanation</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>User</td>
<td>A user or connected user is a person or device that connects to the DVR using a network connection and requests video data.</td>
</tr>
<tr>
<td>Video channel</td>
<td>A channel or video channel represents video data from a scene. So for each camera connected to the DVR system, one video channel will be available.</td>
</tr>
<tr>
<td>Video stream</td>
<td>A stream or video stream represents a compressed version of video data.</td>
</tr>
</tbody>
</table>
## List of Figures

1. **Digital video recorder (DVR) with remote video streaming.** 1

2. **Frame relation in MPEG-encoded video,**
   - using only I- and P-frames. 6
2. **Frame relation in MPEG-encoded video,**
   - using only I-, P-, and B-frames. 6
2. **Methods to obtain bit-rate scalability.** 7
2. **Frame Dropping for MPEG streams with only I- and P-frames.** 9
2. **Frame dropping: bandwidth reduction vs. resulting frame-rate.** 9
2. **Frame Dropping for MPEG streams with I-, P- and B-frames.** 10
2. **Adopted bit-rate scalability approach.** 11

3. **Proposed server architecture.** 13
3. **Architecture of the system implementation.** 17
3. **Graphical user interface (GUI) of the system implementation.** 19

4. **Tree structure for stream selection.** 21
4. **Switch delay for stream-switch from 5 to 1 fps.** 25
4. **Bandwidth utilization cost.** 26
4. **Switch delay cost.** 28
4. **Time since last switch cost.** 29
4. **Video quality.** 30
4. **Stream-selection method: limited switch.** 35
4. **Number of stream-combinations vs channels per sub-set.** 36
4. **Stream-selection method: independent sub-problems.** 37
4. **Computation complexity for all selection methods.** 39
4. **Stream-selection method: pruning.** 40

5. **Network bandwidth over time.** 45
5. **Computational cost vs. optimum cost.** 48
5. **Computational cost vs. optimum cost (zoomed).** 49
5. **Bandwidth usage & selected streams with VBR video.** 50
5. **Optimal cost vs. bit-rate: limited-switch methods.** 53
5. **Optimal cost vs. bit-rate: pruning after 2 channels.** 54
5. **Optimal cost vs. bit-rate: pruning after 3 channels.** 55
5. **Optimal cost vs. bit-rate: pruning after 4 channels.** 56
5. **Bandwidth usage & selected streams with changing priorities.** 58
5. **Bandwidth usage & selected streams with CBR video.** 59
List of Tables

4.1 Switch-delay time to switch-delay cost.......................... 27
4.2 Time since last switch to cost........................................ 28
4.3 Frame-rate to video quality............................................. 30
4.4 Weighting-factor coefficients for the cost values................. 32

5.1 Performance vs. computational complexity for multiple methods. 47
Acknowledgements

This thesis presents the results of a year of research, done as the graduation project for my Master of Science title at the Department of Electrical Engineering of the Technische Universiteit Eindhoven (TU/e). The graduation project was carried out at Bosch Security Systems B.V. in Eindhoven. I hope the presented results can have a positive influence on implementation of a streaming server module in future surveillance systems. There are quite some people that I'd like to thank, who helped me during the project.

First of all, my supervisor, professor dr. ir. P.H.N. de With (Peter) for expanding my interest in the world of video compression and related architectures. This started with my external Internship that was carried out at Bosch Security Systems B.V. in the United States, where I gained interest for both the field of image processing and the world of video surveillance.

Paul Merkus for giving me the possibility of doing my research at Bosch in Eindhoven. As an academic student, it was a great experience to get an inside look at a company that works on future products for the surveillance market. I've enjoyed the combination of the industrial product-related view, together with the theoretical background and support from the university.

Both my supervisors at Bosch, ir. M.C. Jacobs (Marco) and dr. ir. E.G.T. Jaspers (Egbert), who pushed me in the right direction when needed, and were always very critical. All my colleagues at Bosch, with whom I had many interesting discussions that often resulted in new ideas and solutions.

My house-mates for all the interesting social movie nights, in which everyone enjoyed the techniques of video compression.

During the overall duration of my study, but especially in the period of graduation, I have had much support from all my friends and family. Many social gatherings have helped me relax when needed. You have always been motivating and inspiring, and understanding, in times when my study had priority.
References


Appendix A

Video Compression

In our system, we used an MPEG-4 based encoder to create compressed streams for each original video sequence. The following settings were used to create the compressed sequences:

- Advanced Simple Profile @ Level 5
- CIF video resolution (352x288 pixels)
- Frame-rates of 1, 5, 12 and 25 frames per second
- Constant GOP-length of 15 frames
- Target bit-rate of 1 MBit
- H.263 quantization, quantization factor set manually, obtaining an average bit-rate of 1 MBit
- Square pixel aspect ratio
- Motion: high search area, chroma motion, mode decision VHQ mode
- MPEG-4 Advanced Simple Profile options:
  - No B-VOPs
  - No Global Motion Compensation
  - No adaptive quantization (thus same q-factor for all blocks in frame)
  - No quarter-pixel quantization

Constant GOP-length

We chose for a constant GOP-length for streams of both high and low frame-rates, to obtain a constant gain in compression ratio compared to still-image based compression techniques. For network-streaming purposes, it would be better to have a constant GOP-duration, since for each stream the switch-delay (as introduced in Subsection 4.2.2) would be the same. This however, would require large GOP-lengths for streams with high frame-rates or GOP-lengths of one frame (only I-frames) for streams of low frame-rates. Since this contradicts the gain in compression ratio, we chose to use a constant GOP-length.
Video Stream Duration

Because the length of the used sequences is limited (between 30 and 120 seconds), the sequence will regularly be restarted during the operation of the system, to simulate a continuous video stream. Therefore, each sequence has to end with a complete GOP, for all used frame-rates. The duration of a stream is:

\[
\text{GOP-length} \cdot \frac{\text{decimation factor}}{\text{original frame-rate}} \quad (A.1)
\]

The duration of a GOP for the frame-rates 1, 5, 12 and 25 are respectively 15, 3, 1.2 and 0.6 seconds. The greatest common divisor of these values is 30, which means that each sequence must have a duration of multiple times 30 seconds.
Appendix B

Network Transmission

B.1 Introduction

When using best-effort IP networks like the Internet, the available network bandwidth of a link between two systems will be constantly varying. When considering transmission of compressed video from a server to a client, the sending rate of the video streams has to be continuously adjusted in order to achieve a smooth video playback at the client.

When QoS is provided on the network, some parameters like the network bandwidth or delay are guaranteed. In this case, the sending rate does not have to be adjusted since the available network bandwidth remains constant. We restricted our work to the use of best-effort IP networks, where the behavior of the network is not known in advance.

In this chapter we will introduce several issues that should be taken into account when transmitting video over a best-effort network connection. First Congestion Control mechanisms will be discussed. Then packet scheduling and the multiplexing of several variable bit-rate MPEG streams is discussed.

An extended overview of transmission of MPEG-4 compressed video data over a best-effort IP network is given by Feamster in [27].

B.2 Congestion control

Several mechanisms have been defined, that adjust the sending rate to achieve the highest utilization of the available bandwidth. These mechanisms are called Congestion Control mechanisms. Congestion Control mechanisms are only concerned with the actual transmission of data and do not do any form of quality-adaptation.

The most well known is probably the one used by TCP, called Additive Increase Multiplicative Decrease (AIMD). Rate-control using AIMD goes as follows. When a connection is started, the congestion window is doubled every roundtrip time. This window is used to set the transmission rate. When packet-drop is detected, the congestion window size is decreased by 1/2, which means the transmission rate is halved. If no packet loss is detected, the congestion window is increased by one packet per round-trip-time (RTT). When heavy congestion is detected (retransmitted packets are dropped), the window-size is
decreased exponentially. This process is described in more detail by ten Kate in [28].

The AIMD algorithm results in a heavily oscillating transmission rate and a slow convergence to the actual available bandwidth. Therefore, AIMD is clearly not designed for use in applications with a real-time behavior like video-streaming. Other mechanisms have been proposed that have a more smooth transmission rate. These algorithms should be TCP-friendly, which means that they will not influence other connections that use the same network link.

To improve the performance of congestion control algorithms, especially for real-time applications like streaming video, significant research has been done. Multiple protocols have been proposed. The Rate Adaptation Protocol (RAP) in [5], the Streaming Control Protocol (SCP) in [29], the TCP-Friendly Rate Control (TFRC) congestion control is discussed in [30] and [31]. A survey on some of some congestion control protocols is discussed by the authors of [32].

An integrated congestion control architecture is proposed by Balakrishnan et al. in [33]. The proposed Congestion Manager allows applications to easily adapt to congestion in the network. This work has been adopted by the IETF in an RFC [34].

B.2.1 Binomial congestion control

Another class congestion control mechanisms is Binomial Congestion Control, as proposed in [35]. The binomial congestion control class generalizes the increase and decrease rules in AIMD. Feamster [27] proposes the SQRT algorithm. The magnitude of oscillations is much smaller compared to AIMD, because $\sqrt{\omega}$ ($\omega$ is the congestion window size) is used, instead of adjusting with $\omega$ itself.

B.3 Receiver buffering

Perhaps the most simple way to achieve smooth playback at the client with the highest possible video quality is to use an unlimited amount of receiver buffering. Start playing video, when the complete video clip has been received. This is how 'video streaming' worked in the past. The higher amount of buffering used, the smoother the playback, but the higher end-to-end delay. For real-time applications, the amount of buffering should be minimized.

In the field of video surveillance, with controlling of the pan-tilt-zoom (PTZ) functions of a camera, delays below 200 milliseconds are required. This includes en- and decoding of the video and the network transmission.

B.4 Multiplexing of multiple MPEG streams

Multiple methods can be used to transmit MPEG-encoded video streams over a network link. Krunz et al. [36] describe a video-on-demand system that transmits MPEG video. For example, temporal averaging (or smoothing) can be used for both pre-recorded and real-time video. For pre-recorded video, it is also possible to apply buffering and video frames are transmitted prior to their playback times. However, for the optimal transmission schedule exact knowledge of the end-to-end delay is required, to make sure no buffer overflow or underflow occurs. The effectiveness of the algorithm depends on the used
buffer-size. The larger the used buffer (and thus the buffer build-up delay), the more effective the transmission.

Statistical multiplexing can also be applied. The sum of peak rates of input streams is allowed to exceed the output rate of the multiplexer, assuming that the peaks of multiple streams do not occur simultaneously. However, this can result in delay and buffer overflow. Krunz et al., consider transmission of VBR-encoded video over a network link with constant bandwidth. The approach proposed, is based on exploiting the periodic structure of the group of pictures (GOP) pattern in MPEG video to provide appropriate traffic envelopes. Based on such envelopes, a framework has been formulated in [37] for stream scheduling, multiplexing and bandwidth allocation. In [38] the statistical characteristics of VBR MPEG streams were studied for transmission over an ATM link, using statistical multiplexing.

Krunz et al., consider transmission of VBR-encoded video over a network link with constant bandwidth. The approach proposed, is based on exploiting the periodic structure of the group of pictures (GOP) pattern in MPEG video to provide appropriate traffic envelopes. Based on such envelopes, a framework has been formulated in [37] for stream scheduling, multiplexing and bandwidth allocation. In [38] the statistical characteristics of VBR MPEG streams were studied for transmission over an ATM link, using statistical multiplexing.

Zhao and Tripathi [39] describe an optimal multiplexing scheme for single and multiple streams over one network connection. This work is based on earlier work of Salehi et al. in [40] and [41].

Tong et al. [42] introduce a GOP-aware aggregation traffic model, taking into account the arrival times of frames from multiple streams.

One of the core components of a QoS network is the packet scheduling algorithm that determines the transmission order of packets at the output buffers of switches (or the sending server). Most packet-scheduling algorithms send packets in priority order. Lieber and Wrege [43] give an overview of available techniques for scheduling and introduces an approximation of the sorted priority queue.

B.5 Network protocols

Various network protocols can be used to transmit video. The one used for real-time applications is the Real-Time Transport Protocol (RTP) [44] that works on top of both the User Datagram Protocol (UDP) [45] and the Transmission Control Protocol (TCP) [46]. UDP and TCP operate on top of the Internet Protocol (IP) [47].

Together with RTP, the Real-Time Transport Control Protocol (RTCP) protocol is defined [44]. RTCP is used simultaneously with RTP and can be used to send feedback from the receiver to the sender, such as packet-loss and round-trip time information. Control of media, like starting, stopping and fast-forwarding can be done using the Real Time Streaming Protocol (RTSP) [48].