Dipl.-Ing. Michael Grafl, Bakk.techn.

Scalable Media Delivery Chain with Distributed Adaptation

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

Alpen-Adria-Universität Klagenfurt

Fakultät für Technische Wissenschaften

1. Begutachter: Univ.-Prof. DI Dr. Hermann Hellwagner
   Institut für Informationstechnologie
   Alpen-Adria-Universität Klagenfurt

2. Begutachter: Dr. Cyril Concolato
   Département Traitement du Signal et des Images
   Ecole Nationale Supérieure des Télécommunications, Paris (Télécom ParisTech)

Juni 2013

Ehrenwörtliche Erklärung

Ich erkläre ehrenwörtlich, dass ich die vorliegende wissenschaftliche Arbeit selbstständig angefertigt und die mit ihr unmittelbar verbundenen Tätigkeiten selbst erbracht habe. Ich erkläre weiters, dass ich keine anderen als die angegebenen Hilfsmittel benutzt habe. Alle ausgedruckten, ungedruckten oder dem Internet im Wortlaut oder im wesentlichen Inhalt übernommenen Formulierungen und Konzepte sind gemäß den Regeln für wissenschaftliche Arbeiten zitiert und durch Fußnoten bzw. durch andere genaue Quellenangaben gekennzeichnet.

Die während des Arbeitsvorganges gewährte Unterstützung einschließlich signifikanter Betreuungshinweise ist vollständig angegeben.

Die wissenschaftliche Arbeit ist noch keiner anderen Prüfungsbehörde vorgelegt worden. Diese Arbeit wurde in gedruckter und elektronischer Form abgegeben. Ich bestätige, dass der Inhalt der digitalen Version vollständig mit dem der gedruckten Version übereinstimmt.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Michael Grafl Klagenfurt, 28.06.2013

Declaration of Honour

I hereby confirm on my honour that I personally prepared the present academic work and carried out myself the activities directly involved with it. I also confirm that I have used no resources other than those declared. All formulations and concepts adopted literally or in their essential content from printed, unprinted or Internet sources have been cited according to the rules for academic work and identified by means of footnotes or other precise indications of source.

The support provided during the work, including significant assistance from my supervisor has been indicated in full.

The academic work has not been submitted to any other examination authority. The work is submitted in printed and electronic form. I confirm that the content of the digital version is completely identical to that of the printed version.

I am aware that a false declaration will have legal consequences.

Michael Grafl Klagenfurt, 28.06.2013

Acknowledgements

I would like to thank Univ.-Prof. Dr. Hermann Hellwagner, Dr. Christian Timmerer, and Dr. Cyril Concolato for their guidance, support, and very valuable feedback on my work.

Several co-authors have contributed to the papers comprised in this thesis, to whom I am very grateful: Christian Timmerer, Hermann Hellwagner, Daniele Renzi, Daniel Negru, Stefano Battista, Eugen Borcoci, Alex Chernilov, Wael Cherif, Anne-Lore Mevel, Markus Waltl, George Xilouris, Angelos-Christos G. Anadiotis, Jaime Delgado, Angelo Difino, Sam Dutton, Michael Eberhard, Georgios Gardikis, Adlen Ksentini, Panos Kudumakis, Stefan Lederer, Silvia Llorente, Keith Mitchell, Christopher Müller, Benjamin Rainer, Víctor Rodríguez-Doncel, Mark Sandler, Giuseppe Tropea, Iakovos S. Venieris, Xin Wang, and Nikolaos Zotos.

Furthermore, I would like to thank my colleagues for helping and enduring me, especially Dr. Markus Waltl for implementation and integration efforts on countless occasions, DI Stefan Lederer for his help with simulation results, DI Christopher Müller for his support on DASH, DI Benjamin Rainer for his lessons in statistics, and DI Daniela Pohl. I am also grateful to many partners in the ALICANTE project for their support and assistance, especially Daniele Renzi, MSc, for his assistance with the bSoft SVC encoder, Wael Cherif, MSc, for his help on SVC encoding evaluations, and Stefano Battista, ME, for his efforts in maintaining and improving the bSoft encoder.

Most of all, I would like to thank Nina Winkler, Bakk.techn., for all her support, encouragement, feedback, advice, and care. Heartfelt thanks also go to my whole family for their support during my education, especially to my mother, Andrea Grafl.

This work was supported in part by the EC in the context of the ALICANTE project (FP7-ICT-248652).

Kurzfassung

Auf Fernsehern, PCs, Tablets und Mobiltelefonen ist Videostreaming ein ständiger Begleiter unseres täglichen Lebens geworden. Für jedes Video erwarten wir hohe visuelle Qualität, frei von Unterbrechungen oder Verzerrungen, die an das jeweilige Gerät angepasst ist. Aber wie können Streaming-Systeme mit steigendem Datenverkehr, daraus resultierenden Netzwerküberlastungen, sowie den verschiedenen Charakteristika der Ausgabegeräte umgehen?

Diese Dissertation behandelt Ansätze zur verteilten Adaptierung skalierbarer Videoströme für Medienübertragungen. Skalierbare Videoströme bestehen aus mehreren Schichten, die verschiedene Auflösungen, Bildwiederholraten oder Qualitätsstufen des Inhalts ermöglichen. Durch das Weglassen einiger dieser Schichten kann das Video an die verfügbare Bandbreite oder ein bestimmtes Ausgabegerät angepasst werden. Die Adaptierung kann auf der Senderseite, auf der Empfängerseite, sowie auf einem oder mehreren Netzwerkknoten durchgeführt werden. Skalierbare Videocodierung kann auch helfen, Bandbreitenanforderungen in Multicast-Szenarios (z.B. für IPTV) zu reduzieren. Eine berühmte Realisierung skalierbarer Videocodierung ist der Scalable Video Coding (SVC) Standard. Diese Dissertation besteht aus drei Hauptteilen, die sich mit verschiedensten Herausforderungen für effiziente SVC Adaptierung befassen.

Der erste Teil dieser Dissertation widmet sich der Codierung von SVC. Um effiziente Adaptierung zu ermöglichen, muss zum Zeitpunkt der Codierung die Konfiguration der Schichten sorgfältig gewählt werden. Daher wird die Performanz verschiedenster Codierungskonfigurationen und Encoder-Implementierungen evaluiert. Außerdem werden Codierungsrichtlinien für SVC entwickelt, die im Einklang mit den Empfehlungen industrieller Streaming-Lösungen stehen. Die Evaluierungsresultate der entwickelten Codierungsrichtlinien legen nahe, dass Qualitätsskalierung gegenüber Auflösungsskalierung bevorzugt werden sollte. Unterschiedliche Auflösungen zur Unterstützung von Ausgabegeräte-Klassen sollten stattdessen als separate SVC-Ströme bereitgestellt werden.

Der zweite Teil dieser Dissertation beschäftigt sich mit der Tatsache, dass skalierbare Medienformate, wie beispielsweise SVC, nach wie vor weder auf der Senderseite noch auf Ausgabegeräten weit verbreitet sind. Um die Verwendung von SVC für die Netzwerkübertragung zu ermöglichen und um die Streaming-Unterstützung zu verschiedenartigen Ausgabegeräten zu verbessern, wird in dieser Dissertation das Konzept des *SVC Tunneling* eingeführt. Das Video wird auf der Senderseite in SVC transcodiert und später auf der Empfängerseite auf einem erweiterten Home-Gateway wieder zurück in ein anderes Videoformat transcodiert. Das Transcodieren zwischen Videoformaten hat jedoch einen negativen Einfluss auf die Videoqualität. Der Trade-Off zwischen dem Qualitätsverlust und der Bandbreiteneffizienz wird evaluiert. SVC Tunneling mit Qualitätsschichten ermöglicht Bandbreiteneinsparungen bei moderatem Qualitätsverlust (ca. 2,5 dB) im Vergleich zum Streaming separater nicht-skalierbarer Repräsentationen der gleichen Qualitäten.

Im dritten Teil dieser Dissertation werden Adaptierungstechniken für sogenannte Content-Aware Networks untersucht. In Content-Aware Networks sind manche Netzwerkknoten fähig, Videoströme in Reaktion auf schwankende Netzwerklasten dynamisch zu adaptieren. Mit der steigenden Verbreitung von HTTP Streaming wird client-seitige Adaptierung zu einem Hauptfaktor des Betrachtungserlebnisses. Das Umschalten zwischen zwei Repräsentationen (z.B. unterschiedlichen Bitraten) eines Videos kann dieses Betrachtungserlebnis stören. Um den Effekt eines abrupten Qualitätswechsels zu reduzieren, wird das Konzept eines weichen Übergangs zwischen den Repräsentationen entwickelt und evaluiert. Eine subjektive Benutzerstudie deutet darauf hin, dass durch diesen Ansatz die gesamte Betrachtungsqualität tatsächlich gesteigert werden kann. Abschließend werden die Erkenntnisse der vorherigen Teile in einem adaptiven Ende-zu-Ende-SVC-Streaming-System integriert. Evaluierungen dieses Streaming-Systems zeigen, dass das entwickelte Adaptierungsframework die Videoqualität unter Paketverlust im Vergleich zu nicht-adaptivem Streaming maßgeblich (um bis zu 6 dB) verbessert.

Abstract

On TV screens, PCs, tablets, and mobile phones, video streaming has become a constant companion in our daily lives. For every video, we expect high visual quality, free from distortions, that is adjusted to the device at hand. But how can streaming systems cope with the increasing network traffic, the subsequent network congestions, and the different characteristics of end-user terminals?

This thesis covers approaches for distributed adaptation of scalable video resources in media delivery. Scalable video resources consist of several layers that enable various spatial resolutions, frame rates, or qualities of a content. By dropping some of these layers, the video can be adjusted to the available bandwidth or to a specific end-user terminal. The adaptation can be performed on the sender side, on the receiver side, and on one or more network nodes. Scalable media coding can also help to reduce bandwidth requirements in multicast scenarios (e.g., for IPTV). One popular realization of scalable media coding is the Scalable Video Coding (SVC) standard. This thesis consists of three main parts, addressing various challenges towards efficient SVC adaptation.

The first part of this thesis focuses on the encoding of SVC. In order to enable efficient adaptation, the configuration of layers has to be carefully chosen at encoding time. Thus, the performances of various encoding configurations and encoder implementations are evaluated. Furthermore, encoding guidelines for SVC are developed, which are aligned with recommendations of industry streaming solutions. The evaluation results of the developed SVC encoding guidelines suggest that quality scalability should be preferred over spatial scalability for adaptive streaming scenarios. Different resolutions for supporting device classes should rather be provided as separate SVC streams.

The second part of this thesis deals with the fact that scalable media formats, such as SVC, are still not widely adopted neither on the sender side nor on the end-user terminal. In order to enable the deployment of SVC for network transmission and to improve the support for streaming to heterogeneous devices, the concept of SVC tunneling is introduced in this thesis. The video is transcoded to SVC at the sender side and then transcoded back to another video format at the receiver side at an advanced home-gateway. However, the transcoding between video formats has a negative impact on the video quality. The trade-off between quality loss and bandwidth efficiency of SVC tunneling is evaluated. SVC tunneling with quality layers enables bandwidth savings at moderate quality loss (approx. 2.5 dB) compared to streaming separate non-scalable representations of the same qualities.

In the third part of this thesis, adaptation techniques for content-aware networks are investigated. In content-aware networks, some network nodes are capable to dynamically adapt video streams in reaction to varying network loads. With the increasing adoption of HTTP streaming, adaptation at the client side becomes a main factor for the viewing experience. The switch between two representations (e.g., different bitrates) of a video can disrupt that viewing experience. To reduce the effect of an abrupt quality change, the approach of a smooth transition between representations is developed and evaluated. A subjective user study indicates that this approach can indeed improve the overall viewing quality. Finally, the findings of the previous parts are integrated in an adaptive end-to-end SVC streaming system. Evaluations of this streaming system show that the developed adaptation framework significantly improves the video quality under packet loss (by up to 6 dB) compared to non-adaptive streaming.

Table of Contents

"Je n'ai fait celle-ci plus longue que parce que je n'ai pas eu le loisir de la faire plus courte."

Blaise Pascal (1623-1662), Lettres provinciales

1 Introduction

1.1 Motivation

When you think about it, digital video streaming is quite an impressive technological achievement. A digitized sequence of pictures is compressed with such efficiency that it can be sent as a stream of 0s and 1s over a packet-switched network such as the Internet to a computer that is capable of reconstructing and displaying the pictures in real-time. This requires first, efficient video coding formats, second, high-bandwidth networks, and third, computers powerful enough to perform real-time video decoding and playback. Since all these three aspects improve continuously, higher resolutions, better video qualities, and higher frame rates become possible and increasingly common.

As this technology is even available on our mobile phones, we also tend to utilize it more and more often. Be it on TV screens, PCs, tablets, or mobile phones, video streaming has become a constant companion in our daily lives. Our demands for high-definition media streaming often increase faster than the necessary network bandwidths. Thus, we have to cope with the increasing network traffic, the subsequent network congestions, and many different characteristics of end-user terminals. Those terminals have a plethora of different display resolutions and processing capabilities. The particular configuration and encoding of the streamed media does not necessarily match those capabilities. Network congestion results in lower throughput, retransmission delay, or packet loss. Consequently, the end user does not experience the desired quality. Adaptation helps to improve the quality of experience in two respects. First, the streamed media can be adjusted to the terminal in terms of spatial resolution, bitrate, coding format, etc. Second, the media bitstream can be adjusted on the fly by reducing the bitrate to accommodate network congestions. Special media coding techniques can aid the adaptation by making the media representations scalable in terms of spatial resolution, bitrate, and frame rate. Traditionally, adaptation is performed at the server side or at the client side. The deployment of adaptation at network nodes can increase the flexibility of adaptation operations, especially in case of network congestion. In the scope of a media-driven Future Internet, adaptation at the network edges and even within the network becomes increasingly relevant. That is, adaptation can be distributed between the server, the client, and one or more network nodes. However, many challenges remain as how to deploy, configure, and distribute adaptation operations. This thesis will address some of the key research questions towards realizing a scalable media delivery chain featuring distributed adaptation.

1.2 Research Objectives

This thesis investigates mechanisms for distributed adaptation in scalable media streaming systems. Distributed adaptation enables media streaming to heterogeneous devices under varying network conditions. While previous works have focused either on adaptation at network edges or on in-network adaptation, this thesis will combine these approaches, enabling adaptation towards device capabilities at the network edge as well as dynamic adaptation based on network conditions during media delivery. The content-aware media delivery relies on scalable media coding formats such as the Scalable Video Coding (SVC) extension of H.264/AVC.

The research objectives of this thesis are:

(1) to evaluate the performance of *SVC encoding configurations* and scalability features;

(2) to develop *guidelines for SVC encoding* in the context of adaptive media streaming;

(3) to investigate the feasibility of *SVC tunneling* for device-independent access;

(4) to analyze the effects of *scalability features and adaptation configurations* on *content- and context-aware* media delivery;

(5) to investigate the applicability of *distributed adaptation* in *content-aware networks* for different *transport mechanisms*;

(6) to evaluate the performance of *distributed media adaptation* in an end-to-end streaming system.

The thesis will inquire reasonable encoding configurations for adaptive media streaming and evaluate their rate-distortion performance **(1)**. Encoding configurations are guided by display characteristics of typical end-user devices and by realistic network bandwidth estimations. Furthermore, the scalability features of the encoding configurations depend on the anticipated adaptation operations.

In conjunction with the performance evaluations of SVC encoding configurations, guidelines will be developed for the encoding and deployment of SVC for adaptive media streaming **(2)**. They shall comprise suitable resolutions and bitrates, as well as recommendations for the use and configuration of spatial and quality scalability in SVC. In particular, the thesis will investigate whether a single SVC bitstream is always the most suitable choice for streaming of heterogeneous devices.

The concept of *SVC tunneling* allows for device-independent media access in a scalable media streaming system. Inspired by IPv6-over-IPv4 tunneling, content can be converted to and from SVC on the network edges, enabling both delivery of content originally encoded in a non-scalable media format and consumption by

devices without built-in support for scalable media formats. The feasibility and performance of this concept will be evaluated **(3)**.

If network nodes are aware of the transported content and its scalability features (i.e., its spatial, quality, and temporal layers), they can intelligently adapt the content in order to preserve a satisfactory Quality of Experience (QoE) for the end user in case of network congestion. Furthermore, the awareness of an end user's context and the capabilities of end-user terminals allow for advanced adaptation at the network edges. These potentials pose many challenges for the configuration of scalability features and adaptation algorithms. Suitable choices of adaptation configurations will result in guidelines on what, where, when, and how often to adapt **(4)**.

SVC is traditionally transported over the Real-time Transport Protocol (RTP), but with the advance of Dynamic Adaptive Streaming over HTTP (DASH), transport of SVC over HTTP becomes increasingly popular. Another transport mechanism is peer-to-peer (P2P) streaming that allows the retrieval of SVC layers from multiple peers. Media-Aware Network Elements (MANEs) may process and cache SVC layers to improve the network resource utilization and, ultimately, the QoE for the end user. *Media-* or *content-awareness* refers to a network node's ability to intelligently handle the forwarded data based on a limited knowledge about the nature of that data. This thesis will analyze the impact of different transport mechanisms on adaptive SVC streaming in the context of content-aware networking **(5)**.

Finally, the thesis will evaluate the performance of distributed adaptation in an end-to-end streaming system **(6)**. This end-to-end streaming system will demonstrate the integration of SVC encoding guidelines and SVC tunneling over a content-aware network with distributed adaptation. The performance will be evaluated in terms of end-to-end delay and impact on the video quality.

While this thesis covers a wide range of topics in the domain of scalable media delivery, its main focus will be the deployment of SVC in a context-aware media streaming system featuring distributed adaptation in the course of the EU FP7 project ALICANTE.

1.3 Contributions

This thesis comprises multiple scientific contributions in the field of distributed adaptation that have been published in the proceedings of international conferences and workshops, in international standards, project deliverables, book chapters, and journals.

The research on best practices of *SVC encoding for adaptive media streaming* was published in **[1]** and **[2]**. While previous studies of SVC encoding performance typically did not consider realistic encoding configurations as those used by actual industry streaming solutions, our work has investigated encoding recommendations of popular streaming solutions in order to devise guidelines for encoding scalable media resources at multiple representations. Based on those guidelines, we

performed extensive SVC encoding performance studies of multiple SVC encoder implementations at various configurations for high-definition (1080p) content. Furthermore, we proposed and evaluated the concept of hybrid SVC-DASH to optimize the usage of SVC in adaptive HTTP streaming for heterogeneous devices.

The concept of *SVC tunneling* was introduced and evaluated in **[3]**, **[4]**, and **[5]**. Media resources are transcoded at the network edges to allow the use of SVC in the network for fast adaptation and network resource optimization on the one hand, and media access from heterogeneous devices on the other hand. We have evaluated the trade-off between quality loss and bandwidth efficiency in order to enable advanced control of transcoding configurations.

An overview of the *ALICANTE architecture*, along with *research challenges for scalable media adaptation in content-aware networks* was published in **[6]** and subsequently in **[7]**. The research challenges and innovation areas were further evaluated in **[5]**. Additional information on the ALICANTE adaptation framework was provided in the project deliverables **[8]** and **[9]**.

Use cases and challenges for *scalable media coding in content-aware networks* were published in **[10]** and **[11]**. The work has identified the advantages and open issues of deploying scalable media coding for different forms of media transport (i.e., RTP unicast and multicast, HTTP streaming, and P2P streaming) and explored the potential of distributed adaptation in such settings. The distributed adaptation framework of the ALICANTE architecture was demonstrated in an integrated end-to-end streaming system prototype. Some auxiliary tools for this streaming system were made available as open-source software **[12][13]**.

For the particular issue of flickering experienced when adaptation is performed in HTTP streaming, we have introduced the concept of smooth transitions between representations, also referred to as *representation switch smoothing* **[14]**. Instead of a single, noticeable switch between two (bitrate) representations, the video quality is continuously adjusted to result in a smooth transition between those representations.

In the broader scope of *interoperable media delivery* towards the Future Internet, contributions to International Standards for *Multimedia Content Description* (MPEG-7) **[15]** and *Multimedia Service Platform Technologies* (MPEG-M) **[16][17]** were made as documented in **[18]**, **[19]**, **[20]**, and **[21]**.

1.4 Structure

The remainder of this thesis is structured as follows. Chapter 2 will give an overview of the technical background on Scalable Video Coding and on the ALICANTE project.

Chapter 3 will target encoding guidelines for SVC. In addition to recommendations derived from industry solutions, we will also provide performance evaluations of major encoder implementations at a wide range of encoding configurations.

The concept of SVC tunneling will be introduced and evaluated in Chapter 4. SVC tunneling allows the use of scalable media resources in the network, regardless of the media coding formats deployed at the server or client. The goal is to facilitate adaptation and to reduce network resource utilization in multicast streaming scenarios. The necessary transcoding steps to and from SVC impact the video quality. We will evaluate the trade-off between bandwidth savings and quality degradation.

Distributed adaptation of SVC and various aspects of media transport will be discussed in Chapter 5. In particular, use cases for scalable media coding in content-aware networks will be analyzed and an in-depth discussion of SVC adaptation in the context of the ALICANTE project will be given. An adaptive end-to-end streaming demonstrator will be described and evaluated in that chapter as well.

Chapter 6 will conclude the thesis with a wrap-up of the research objectives and an outlook on future work in this field.

2 Technical Background

Before we dive into the research carried out for this thesis, the following sections provide some background on the involved technologies. We will explain how scalability in video coding is achieved and how it can be utilized. Before that, a brief introduction of video coding tools in general is provided. Towards the development of a media delivery chain, an overview of the FP7 ALICANTE project is presented. The key innovations and research challenges related to a distributed adaptation framework within that project are also discussed.

The purpose of this chapter is for the reader to become familiar with the concepts of SVC and content-aware media delivery. Special discussions of related work on particular topics will be provided within the respective chapters.

Parts of the work presented in this chapter are published in [7], [6], [8], [19], [20], and [21].

2.1 Video Coding

Digital video is nowadays omnipresent in many different forms (e.g., entertainment, surveillance, communication) and on an increasing variety of devices. For this to work, an efficient digital representation of moving pictures is necessary.

2.1.1 Encoding Tools

In uncompressed, unencoded form, digital video simply comprises the color information of each pixel, frame after frame. One possible representation of such raw digital video is *YUV*, which contains the luminance component (*Y*), and the chrominance components (*U* and *V*) of each pixel. As the human visual system is more sensitive to luminance than to chrominance, luminance values are often sampled at double the resolution of the corresponding chrominance values [22].

Video coding drastically reduces the size of a digital video through specialized compression techniques. Video encoders partition a raw input picture into so-called *macroblocks*, typically consisting of 16x16 samples for their processing. The main processing tools of a typical video encoder are [23][24][25]:

- *Image compression* tools are deployed to compress each frame:
    - The picture data is *transformed* from the spatial domain into the frequency domain via a discrete Fourier-related transformation, such as a discrete cosine transform (DCT) or Hadamard transform (HT). The transformation removes spatial redundancy and, thus, allows for better compression.

  - The transform coefficients are *quantized*. That is, the transform coefficients are divided by values specified in a quantization matrix. As a result, the less significant transform coefficients become small or even zero. This step loses image information as the precision of less significant transform coefficients is reduced. The quantization is controlled by a *quantization parameter* (QP), based on which the quantization matrix is produced.
  - Then, *entropy coding* is performed on the quantized transform coefficients to compress the data.

- To reduce inter-frame redundancy, *motion estimation* (also known as motion prediction) is deployed. For each block of luminance samples of the current frame, a block-matching algorithm finds the best match in a reference frame. The displacement is represented as a motion vector. Some video coding formats also allow *intra-prediction*, where macroblocks are predicted from the current frame (instead of a past or future reference frame).
- *Motion compensation* accounts for the mismatches between the motion prediction model and the current frame itself. A motion-compensated residual frame is produced and stored along with the motion vectors.
- Some video coding formats comprise also a *deblocking filter* that reduces the visual artifacts from block-based transform and motion compensation. While deblocking can also be performed as post-filtering after the decoder, integration within the encoder loop improves the video quality [25].

Figure 1 shows a generalized block diagram of a typical video encoder.

For lossy video encoding, the QP determines the compression efficiency of the coded video (i.e., the bitrate) as well as its visual quality, more precisely, the distortion of the reconstructed frames. The relationship between bitrate and distortion is called rate-distortion (RD) performance. The rate control of the encoding process is either achieved by statically setting the QP or by applying a rate control algorithm that dynamically adjusts the QP during encoding to achieve a certain bitrate.

2.1.2 Advanced Video Coding

Video coding standards often specify only the decoder, leaving the encoder counterpart open for competition and innovation. Throughout this thesis, we focus on the deployment of the MPEG-4 Advanced Video Coding (AVC) International Standard [23] and its extension for SVC. The standard was jointly developed by the Moving Picture Experts Group (MPEG) and the International Telecommunication Standardization Sector (ITU-T). The standard is formally known as ISO/IEC 14496-10 as well as ITU-T Rec. H.264. It is commonly referred to as H.264/AVC or simply AVC.

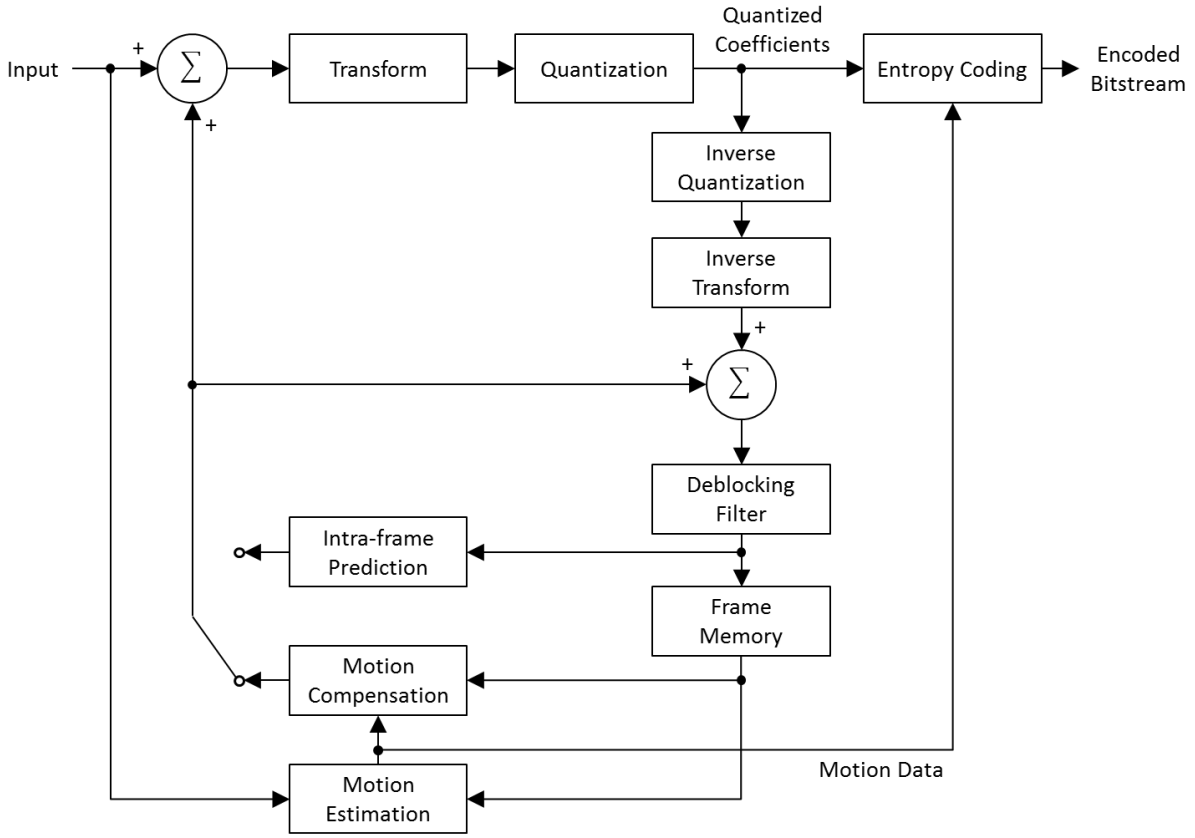


Figure 1: Generalized block diagram of an example video encoder, adopted from [25].

H.264/AVC uses the low-complexity integer-based HT for the transformation into the frequency domain. Entropy coding is performed either via context-adaptive variable length coding (CAVLC) or context-adaptive binary arithmetic coding (CABAC). The format supports inter-frame motion prediction and intra-frame prediction modes. The decoding loop also includes a deblocking filter [23][25].

A coded frame can either be self-contained by relying only on intra-prediction (*I frame*), or its macroblocks are predicted (*P frame*) from a (past or future) reference frame, or its macroblocks are bi-predicted (*B frame*) from two reference frames. (Technically, a frame is divided into one or more *slices*, to which the prediction modes apply, and H.264/AVC specifies two additional types of slices as explained in [25]. For the sake of this discussion, the notions of I, P, and B frames are sufficient.)

A self-contained set of consecutive frames is called a group of pictures (GOP). A GOP limits error propagation in time and can provide entry points for a decoder as the frames inside a GOP only reference each other but no frame from outside the GOP. To facilitate random entry into a bitstream, *instantaneous decoding refresh* (IDR) frames allow the decoder to initialize and to start decoding at a certain GOP. This is especially useful for video transmission if the receiver wants to consume a video stream that has already started.

The design of H.264/AVC covers a video coding layer (VCL), handling the actual coded video data, and a network abstraction layer (NAL) that encapsulates the VCL

data into so-called NAL units (NALUs) and equips them with header information. The NALU is designed to enable flexible media transport and storage. NALUs contain video data (typically one frame or slice per NALU) as well as control information [24].

A coded video bitstream can be further encapsulated in a container format, such as the MP4 file format [26]. The container also provides media access, synchronization, and control information as well as other video, audio, and metadata streams. In differentiation from the container format, a mere video (or audio) bitstream is often called *Elementary Stream* (ES).

2.2 Scalable Video Coding

SVC follows a layered coding scheme comprising a *base layer* (BL) and one or more *enhancement layers* (ELs) with various scalability dimensions [27]:

- *Spatial scalability*

  A video is encoded at multiple spatial resolutions. By exploiting the correlation between different representations of the same content with different spatial resolutions, the data and decoded samples of lower resolutions can be used to predict data or samples of higher resolutions in order to reduce the bitrate to code the higher resolutions.

- *Quality scalability*

  A spatial resolution can be encoded at different qualities. The data and decoded samples of lower qualities can be used to predict data or samples of higher qualities in order to reduce the bitrate to code the higher qualities. Quality scalability is also known as signal-to-noise ratio (SNR) or bitrate scalability.

- *Temporal scalability*

  The motion compensation dependencies are structured so that complete pictures (i.e., their associated packets) can be discarded from the bitstream, thus, reducing the frame rate of the video. Note that temporal scalability is already enabled by AVC and that SVC only provides supplemental enhancement information (SEI) to improve its usage. A hierarchical prediction structure between frames has to be used to allow temporal scalability [24].

To identify an enhancement layer, the NALU header provides information about the comprised data. The NALU header specifies the *dependency identifier* (DID) for the spatial layer, the *quality identifier* (QID) for the quality layer, and the *temporal identifier* (TID) for the temporal layer.

SVC specifies two different modes for spatial scalability. With *dyadic spatial scalability*, the resolution of a video is doubled (in both width and height) from one layer to the next. With *extended spatial scalability* (ESS), the ratio between

resolutions can be arbitrary, involving even changes in aspect ratio and cropping [28]. Temporal scalability also supports dyadic and nondyadic modes [27].

The concept of scalable media coding has been around for several decades, early spatial scalability techniques for video are attributed to Jones [29] back in 1979 [30] and all three scalability dimensions (spatial, temporal, and quality) were supported by MPEG-2 [31], approved in 1994, via the scalable profile. However, scalability features of MPEG-2 were never adopted by industry, mainly due to their high compression overhead. In 2007, the SVC extension for H.264/MPEG-4 AVC was standardized [32], providing a promising scalability coding scheme that reduced compression overhead down to approx. 10% compared to AVC [33].

For audio coding, bitrate scalability is discussed, e.g., in [34], [35], and [36]. Note that for multimedia streaming scenarios, video scalability is generally preferred over audio scalability because video coding requires far higher bitrates than audio coding.

Note that throughout this work, the term *SVC* denotes the H.264/AVC extension, while *scalable media coding* stands for the general concept.

SVC is not the only scalable media coding scheme available today. Wavelet-based scalable video coding (WSVC) deploys discrete wavelet transform (DWT), an operator that decomposes the original signal into a set of so-called subbands, to obtain scalability [37]. Another technique of scalable media coding is Multiple Description Coding (MDC) [38]. While SVC has a cumulative layered scheme, with enhancement layers depending on lower layers, MDC encodes the content into independent layers, called *descriptions*. The content can be reconstructed from any subset of these descriptions. More descriptions yield better quality of the reconstructed content. The independence of descriptions makes MDC well-suited for application areas where a video is transported through multiple disjoint unreliable channels.

SVC enables fast, low-complexity video adaptation, even on devices with restricted computing resources, by avoiding computationally expensive transcoding operations. Lower resolutions, frame rates, and bitrates can be extracted from the video bitstream by removing unnecessary NALUs. A prerequisite for this adaptation is that these *extraction points* have been foreseen at encoding time, i.e., that the encoder has considered all adaptation operations that may subsequently be performed on the video.

The main advantages of SVC (or scalable media coding in general) can be summarized as follows:

- *low-complexity adaptation* that allows dynamic adjustment of video streaming to the network conditions and device characteristics;
- *media storage savings*, i.e., a server only has to store a single SVC bitstream instead of multiple representations for the different resolutions or bitrates it offers;

- ensuing *bandwidth utilization savings* in certain scenarios such as multicast streaming;
- selective treatment of SVC layers to enable:
    - *SVC-specific encryption* to protect specific layers [39] – e.g., encryption of some enhancement layers to provide a free low-quality version of a video and to charge for higher quality in an entertainment media streaming use case;
    - *unequal error protection* [40] and *differentiated routing/forwarding* of SVC layers, i.e., lower SVC layers receive higher error protection.

Detailed information on SVC and its coding tools can be found in [27].

2.3 ALICANTE Project

The demand for access to advanced, distributed media resources is nowadays omnipresent due to the availability of Internet connectivity almost anywhere and anytime, and of a variety of different devices. This calls for rethinking of the current Internet architecture by making the network aware of which content is actually transported. This section introduces the European FP7 Integrated Project "*Medi**a** Ecosystem Dep**l**oyment through Ubiqui**t**ous **C**ontent-**A**ware **N**e**t**work **E**nvironments*" (*ALICANTE*) [41] that researches, among other topics, the deployment of SVC as a tool for Content-Aware Networks (CANs). As this thesis focuses on the distributed adaptation of SVC and on SVC-based media delivery in general, a description of the project this research originated from contributes to a better comprehension of the conveyed ideas. The architecture of ALICANTE with respect to SVC and CAN is presented, use cases are described, and research challenges and open issues are discussed.

2.3.1 ALICANTE Architecture

In recent years the number of contents, devices, users, and means to communicate over the Internet has grown rapidly and with that the heterogeneity of all the involved entities. Many issues can be associated with that, which are generally referred to as ongoing research in the area of the Future Internet (FI) [42]. One project in this area is the FP7 project ALICANTE which proposes a novel concept towards the deployment of a new networked *Media Ecosystem*. The proposed solution is based on a flexible cooperation between providers, operators, and end users, finally enabling every user (1) to access the offered multimedia services in various contexts, and (2) to share and deliver her/his own audiovisual content dynamically, seamlessly, and transparently to other users [43].

2.3.1.1 Overview

The ALICANTE architecture promotes advanced concepts such as *content-awareness* to the network environment, network/user *context-awareness* to the service environment, and *adapted services/content* to the end user for her/his best service experience. The end user can take the role of a consumer and/or producer. The term *environment* denotes a grouping of functions defined around the same functional goal and possibly spanning, vertically, one or more architectural (sub-)layers. This term is used to characterize a broader scope than the term *layer*.

Two novel virtual layers are proposed on top of the traditional network layer as depicted in Figure 2: the *CAN layer* for network packet processing and a *Home-Box (HB) layer* for the actual content adaptation and delivery. Furthermore, SVC is heavily employed for the efficient, bandwidth-saving delivery of media resources across heterogeneous environments. The ALICANTE project also contributed to standardization actions in MPEG in order to foster interoperability of media service platforms.

Innovative components instantiating the CAN are called *MANEs*. They are CAN-enabled routers and offer content-aware and context-aware Quality of Service/Experience (QoS/QoE), content-aware security, and monitoring features, in cooperation with the other elements of the ecosystem.

At the upper layers, the *Service Environment* uses information delivered by the CAN layer and enforces network-aware application procedures, in addition to user context-aware ones. The Service Environment comprises Service Providers and Content Providers (SP/CP) which offer high-level media services (e.g., video streaming, video on demand, live TV) to the end users.

The novel proposed *Home-Box* entity is a physical and logical entity located at end users' premises which is gathering context, content, and network information essential for realizing the big picture. Associated with the architecture there exists an open, metadata-driven, interoperable middleware for the adaptation of advanced, distributed media resources to the users' preferences and heterogeneous contexts enabling an improved Quality of Experience. The adaptation is deployed at both the HB and CAN layers making use of scalable media resources as outlined below.

For more detailed information about the ALICANTE architecture, the interested reader is referred to [43].

2.3.1.2 Scalable Video Coding and Content-Aware Networks

In the ALICANTE architecture, adaptation relies on SVC. The adaptation deployed at the CAN layer is performed in a MANE [44]. MANEs, which receive feedback messages about the terminal capabilities and delivery channel conditions, can remove the non-required parts from a scalable bitstream before forwarding it. Thus, the loss of important transmission units due to congestion can be avoided and the

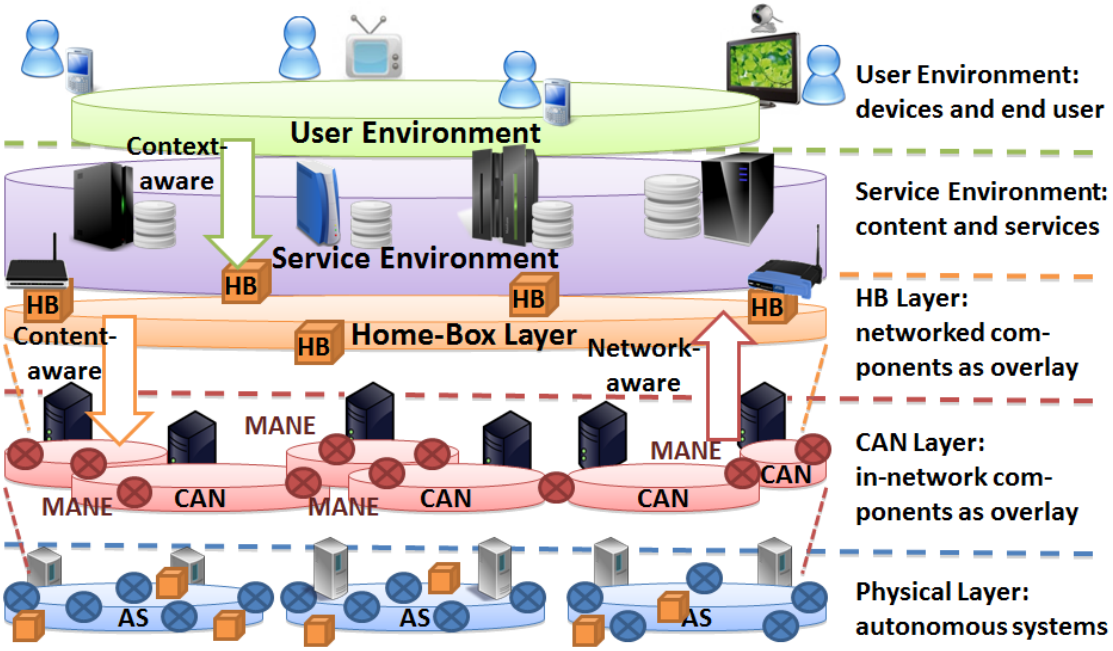


Figure 2: ALICANTE concept and system architecture, adopted from [7].

overall error resilience of the video transmission service can be substantially improved.

Design options for in-network adaptation of SVC have been described in previous work [45] and first measurements of SVC-based adaptation in an off-the-shelf WiFi router have been reported in [46]. More complex adaptation operations that are required to create scalable media resources, such as transcoding [47] of media resources which have increased memory or CPU requirements, are performed at the edge nodes only, i.e., in the Home-Boxes. Therefore, the ALICANTE project has developed an SVC (layered-multicast) tunnel, as detailed later on in Chapter 4, inspired by IPv6-over-IPv4 tunnels. That is, within the CAN layer only scalable media resources – such as SVC – are delivered adopting a layered-multicast approach [48] which allows the adaptation of scalable media resources by the MANEs implementing the concept of distributed adaptation. At the border to the user, i.e., the Home-Box, adaptation modules are deployed enabling device-independent access to the SVC-encoded content by providing X-to-SVC and SVC-to-X transcoding/rewriting functions, where X={MPEG-2, MPEG-4 Visual, MPEG-4 AVC, etc.}. An advantage of this approach is the reduction of the load on the network (i.e., no duplicates), making it free for (other) data (e.g., more enhancement layers). However, multiple adaptations may introduce challenges that have not been addressed in their full complexity (cf. Section 2.3.3).

Due to multiple locations within the delivery network where content may be subject to adaptation, we propose a distributed Adaptation Decision-Taking Framework (ADTF) that coordinates the local adaptation decisions of modules at the content source, the border to the user (Home-Box), and within the network at MANEs.

The key innovations of the ALICANTE project with respect to service/content adaptation are as follows [8][5]:

- Better network *resource utilization* based on adaptation and *maintaining a satisfactory QoS/QoE*: Content is encoded in or transcoded to scalable media formats such as SVC for efficient layered multicast distribution enabling in-network adaptation. End users and network devices provide QoS/QoE feedback to the ADTF, to adjust the service in a distributed and dynamic way.
- Context information from multiple receivers is aggregated at MANEs and used for *local* adaptation decision-taking. Additionally, adaptation decisions are propagated within the media delivery network enabling *distributed* adaptation decision-taking.
- *Distributed coordination* for optimal adaptation and improved bandwidth usage involves the active participation of multiple entities across the media delivery network such as adaptation decision-taking, actual adaptation, and QoS/QoE probes.

2.3.1.3 Media Streaming Advances

While media streaming is traditionally performed via RTP [49], DASH [50] has recently become popular both in the research community and industry solutions. In DASH, a client realizes continuous streaming via the sequential download of temporal media segments. The segments are listed in a Media Presentation Description (MPD). The MPD also describes multiple *representations* of the same content (e.g., at different resolutions or bitrates) between which the client can dynamically switch. The ALICANTE and Social Sensor [51] projects have jointly contributed to the implementation of tools for DASH [52].

In addition to RTP- and HTTP-based streaming, the ALICANTE architecture also deploys P2P-streaming tools developed by the P2P-Next project [53].

2.3.1.4 Towards Media Service Platform Technologies

An advanced media ecosystem as envisaged by the ALICANTE project comprises not only adaptive media delivery, but also innovation and interoperability along the entire media-handling value chain. The media-handling value chain spans from content creation and registration over editing, processing, and publication to delivery and ultimately consumption. Therefore, the ALICANTE project has contributed to the standardization of MPEG-M Elementary Services [16] and Service Aggregation [17] in order to provide interoperable media services [54].

MPEG-M, also referred to as *Multimedia Service Platform Technologies* (MSPT), is a suite of standards that has been developed for the purpose of enabling the easy design and implementation of media services via devices that interoperate seamlessly because they are all based on the same set of technologies, exposed

through standard APIs. MPEG-M specifies a set of *Elementary Services* and respective protocols enabling distributed applications to exchange information related to content items and parts thereof, including rights and protection information. *Service Aggregation* specifies mechanisms for enabling the combination of Elementary Services and other Services to build Aggregated Services. For example, Elementary Services provide interfaces for processing (i.e., adaptation, transcoding, etc.) or delivery of content [19].

For a detailed description of MPEG-M, the interested reader is referred to [55], [20], and [21].

The ALICANTE and P2P-Next [53] projects have also jointly contributed to the standardization of an amendment to MPEG-7 Multimedia Description Schemes (MDS), targeting social metadata [15]. The amendment provides means for describing a person in the context of social networks [56], fostering the integration of social networking in interoperable future media ecosystems [18].

2.3.2 Use Cases

In order to demonstrate the concept of SVC in the context of CANs/HBs, several use cases have been defined, a selection of which is briefly introduced in the subsequent sections.

2.3.2.1 Multicast/Broadcast

In this scenario, multiple users are consuming the same content from a single provider (e.g., live transmission of sport events). The users may have different terminals with certain capabilities as depicted in Figure 3. The ALICANTE infrastructure is simplified in Figure 3 to highlight the interesting parts for this scenario (i.e., the Home-Boxes and the MANEs). Note that the SVC layers depicted in the figure are only examples and that SVC streams in ALICANTE may comprise temporal, spatial, and quality (SNR) scalability with multiple layers. The properties and numbers of SVC layers will be determined by the Home-Box at the Service/Content Provider side based on several parameters (e.g., diversity of terminal types, expected network fluctuations, size overhead for additional layers, available resources for SVC encoding/transcoding, etc.) which are known a priori or dynamically collected through a monitoring system operating across all network layers.

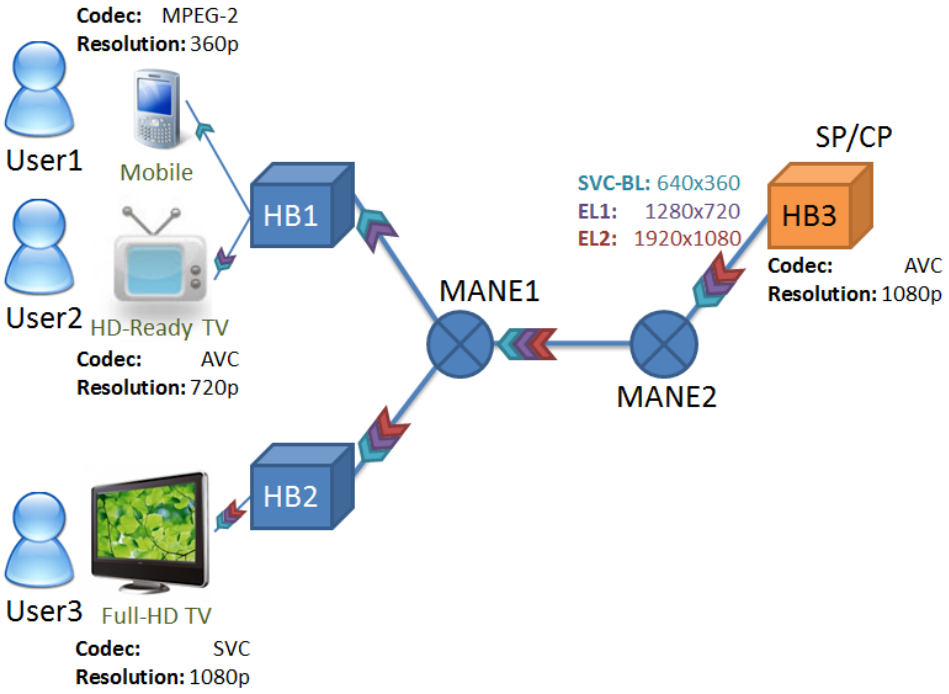


Figure 3: Multicast/broadcast use case with SVC adaptation, adopted from [7].

2.3.2.2 Home-Box Sharing

In this scenario, a user consumes content through a foreign (shared) Home-Box, e.g., the user accesses the content/service to which she/he has subscribed while being abroad (e.g., business trip, vacation). Figure 4 depicts a user consuming content at two different locations on two different terminals, connected to different Home-Boxes. Note that the user might as well use a mobile phone to consume content through *HB2*.

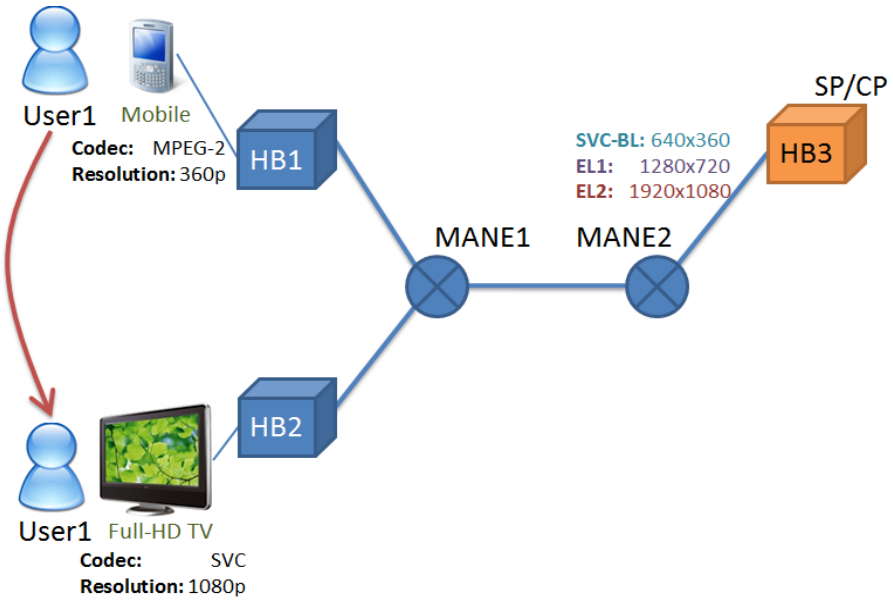


Figure 4: Home-Box sharing use case, adopted from [7].

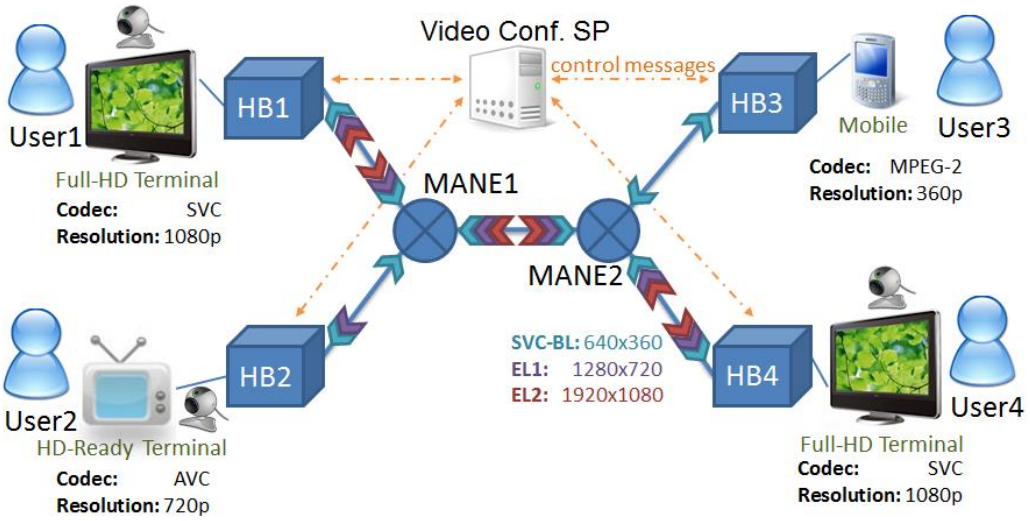


Figure 5: Video conferencing use case, adopted from [7].

2.3.2.3 Video Conferencing

This scenario consists of a video conferencing session (e.g., in family meetings, office meetings, etc.) as depicted in Figure 5. The media distribution is handled over a multicast shared bi-directional non-homogeneous tree in the ALICANTE network. In such a way only the minimum amount of network resources are spent, while assuring maximum quality to the end user. Assymetric connections (e.g., between *HB2* and *MANE1* are also considered.

2.3.2.4 Peer-to-Peer Media Streaming

The Home-Boxes operate in P2P mode within the ALICANTE ecosystem as illustrated in Figure 6. The MANEs, through which the P2P traffic flows, act as proxy caches which intercept requests for content pieces issued by Home-Boxes and aggregate them respecting the capabilities of requesting terminals. Furthermore, content pieces are only forwarded if the requesting terminals can decode them. Therefore, unnecessary traffic is reduced to a minimum freeing up the network resources for other data (e.g., additional enhancement layers).

2.3.3 Research Challenges and Open Issues

The heterogeneity of devices, platforms, and networks is and most likely will be a constant companion within future media (Internet) ecosystems. Thus, we need to provide tools to cope with that heterogeneity in order to support a maximum of use cases while optimizing (network) resource utilization and improving QoE. One such tool is the SVC tunneling approach featuring edge and in-network media adaptation.

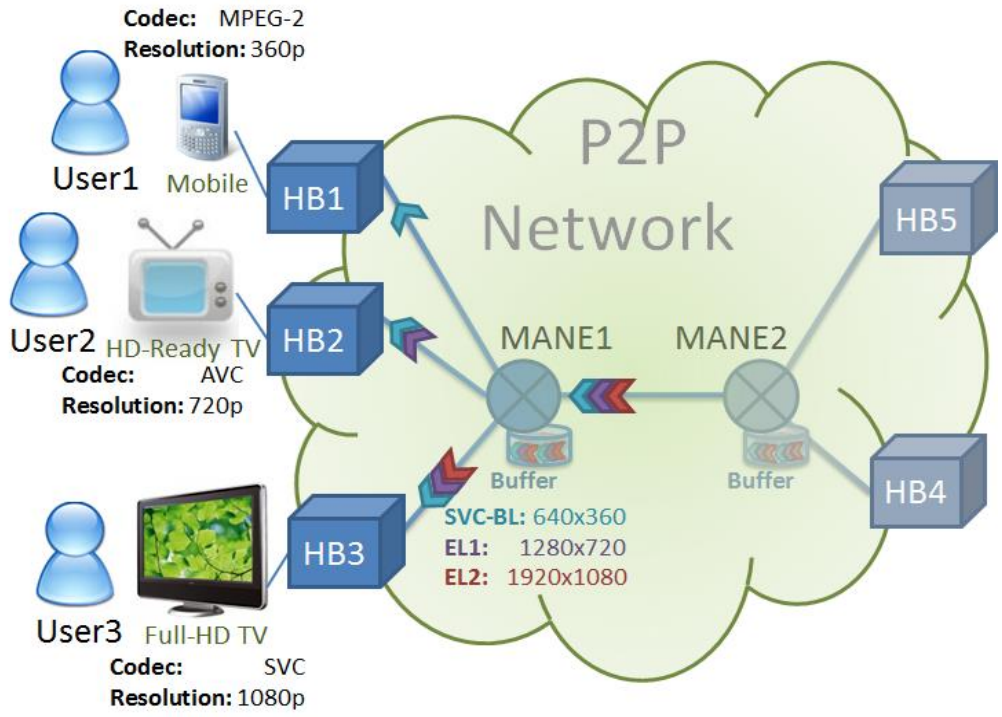


Figure 6: P2P media streaming use case, adopted from [7].

In this section we point out some research challenges and open issues with respect to utilizing Scalable Video Coding within Content-Aware Networks.

2.3.3.1 Distributed Adaptation Decision-Taking Framework

Due to the fact that many, possibly heterogeneous entities are involved – in the production, ingestion, distribution, and consumption stages – there is a need to develop a framework for distributed adaptation decision-taking; that is, finding the optimal decision regarding the adaptation of the content for a single entity (i.e., Home-Box, MANE) within a network of various entities in the delivery system. Note that decision-taking is needed at the request stage and during the delivery of the multimedia content as (network) conditions might change.

While the *adaptation framework* operates mainly at flow level – a *flow* denotes the media stream data transmitted over an individual transport-layer session between two network sockets – whereas the *CAN management* deals with control information at an aggregated level (i.e., it is not aware of individual media streams). Appropriate *cooperation* between them and mappings for monitoring and control information have to be defined in order to ensure efficient use of transport resources.

The actual *adaptation* at both layers needs to be done efficiently, based on several criteria, in order to obtain low (end-to-end) delay, minimum quality degradation, and assuring scalability in terms of the number of sessions that can be handled in parallel.

The following research questions arise:

- *Where to adapt?* At the content source, within the network (with multiple options), at the receiving device, and combinations thereof.
- *When to adapt?* At request and during the delivery enabling dynamic, adaptive streaming based on the user's context.
- *How often to adapt?* Too often may increase the risk of flickering, whereas too seldom may result in stalling, both having a considerable impact on the QoE.
- *How to adapt?* The optimization towards bitrate, resolution, frame rate, SNR, modality, accessibility, region-of-interest (ROI), etc. results in (too) many possibilities and often depends on the actual content, genre, and application.

2.3.3.2 Efficient, Scalable SVC Tunneling

The approach of tunneling the content within SVC streams in the (core) network opens up a number of issues due to SVC adaptation within the MANEs, SVC transcoding/rewriting within the Home-Boxes, and the associated signaling requirements. The issues range from efficiency and scalability to quality degradations and latency:

- Minimum *quality degradation* and *scalability* w.r.t. the number of parallel sessions and acceptable (end-to-end) latency.
- How can transcoding and adaptation steps be organized to minimize impact on QoS and video quality?
- How many parallel sessions can be supported on network and client equipment?

2.3.3.3 Impact on the Quality of Service/Experience

As there may be many adaptations happening during the delivery of the content, the impact on QoS and QoE needs to be studied in order to find the best trade-off for the use cases in questions. While for the QoS many objective measures are available, the QoE is highly subjective and requires tests involving end users. These tests are time consuming and costly. In any case, a good test-bed is needed for both objective and subjective tests for the evaluation of the QoS and QoE, respectively. The corresponding research challenges are:

- The *QoS/QoE trade-off* for the use cases and applications developed in ALICANTE. One example is the trade-off between quality degradation due to transcoding against the QoE gain of dynamic bitrate adaptation.
- Possible *mappings* of QoS to QoE. Established network QoS parameters (such as packet loss, delay, and jitter) as well as objective video quality are taken into account for estimating the viewing experience.

2.4 Conclusions

In this chapter, we have provided a brief overview of video coding principles and the Scalable Video Coding technology. We have also introduced the usage of SVC in Content-Aware Networks for various use cases. In particular, SVC is a promising tool for making the network aware of the actual content being delivered, i.e., when it comes to technical properties such as bitrate, frame rate, and spatial resolution. Furthermore, it allows for efficient and easy-to-use in-network adaptation due to the inherent structure of SVC.

The goal of the ALICANTE project is to provide an advanced Media Ecosystem that enables the management of media services with respect to QoS and QoE on the one hand, while delivering the media content at dynamically adaptable bitrates to heterogeneous terminals on the other hand. The outlined use cases of the ALICANTE architecture indicate the advantages of using SVC and in-network adaptation. We have highlighted research challenges and open issues, some of which will be tackled in the following chapters.

3 Scalable Video Coding Framework

3.1 Introduction

The need for scalability (e.g., spatial, temporal, signal-to-noise ratio) in video coding is often motivated to address heterogeneous environments in terms of terminal characteristics (e.g., different resolutions) and network conditions (e.g., varying available bandwidth). Recently, the development of a scalable extension for High Efficiency Video Coding (HEVC) has started [57]. Today's state of the art solution is SVC, an extension to the AVC standard which employs a cumulative layered coding approach [27]. In addition to temporal scalability of AVC, SVC supports spatial and quality scalability. Quality scalability can be achieved through coarse-grain scalability (CGS), which uses the same mechanisms as spatial scalability but at a single resolution, or through medium-grain scalability (MGS), which enables a finer granularity for adaptation per video frame. For the MGS mode, most encoders, such as the reference software Joint Scalable Video Model (JSVM) [58], perform requantization, the QP for which is configured manually.

The deployment of SVC has an important role in adaptive media streaming. In particular, it allows the adaptation to the users' contexts and enables in-network adaptation in emerging content-aware networks [7]. MANEs can adapt SVC streams on the fly during the delivery to accommodate changing network conditions (e.g., congestion) [45]. For this technique to work, the content has to be encoded appropriately, taking expected terminal capabilities (such as resolution) and characteristics of the codec into account.

SVC offers significantly more encoding configuration parameters than non-scalable video formats due to the configurations of its layers. Suitable configurations depend on the expected adaptations in a given use case. Possible use cases include Video on Demand (VoD) streaming, multicast of live or non-live content, streaming of user-generated content, video conferencing, and video surveillance. Each of these use cases poses different requirements on adaptation and subsequently on the SVC encoding configuration. In use cases where network transmission is involved, coding efficiency is an important aspect of the encoding process. Constellation and number of layers can have considerable impact on the coding efficiency.

This chapter devises encoding recommendations for SVC for adaptive media streaming applications based on a survey of media streaming industry solutions. The RD performance of these recommendations is validated for various encoders and several further encoding configurations for adaptive media streaming are evaluated for high-definition (HD) content. In our tests, we investigate appropriate SVC layer configurations for selected streaming-related use cases and study the trade-off between bandwidth requirements and video quality. We further extend our studies to focus on SVC-based HTTP streaming.

The majority of devised guidelines and performed evaluations on SVC apply to streaming scenarios regardless of the media transport (i.e., RTP, P2P streaming, or DASH). The implications of deploying SVC-based streaming in Content-Aware Networks (CANs) with various media transports will be discussed in Chapter 5. Since the delivery of SVC over DASH (*SVC-DASH*) has recently gained attention by the research community [59][60][61][62], we also place special focus on SVC-DASH for selected evaluations. In this context, we introduce the concept of Hybrid SVC-DASH, which is based on the hypothesis that SVC encoding with one stream (including several quality layers) per resolution is better suited for DASH than a single stream that combines spatial and quality scalability. We validate our hypothesis in terms of RD performance, and present further quality evaluations of SVC configurations related to SVC-DASH.

The work presented in this chapter is published in [1] and [2].

In the following sections we first discuss related work and streaming recommendations of prominent industry solutions, devise guidelines for SVC, present the tested SVC encoders, explain the selection of test sequences, and then detail the SVC configurations of our test scenarios. Thereafter we present and discuss test results.

3.2 Related Work

3.2.1 SVC Performance

A considerable amount of SVC performance tests is available in the technical literature. Wien et al. [33] and Schwarz et al. [27] provide performance evaluations of several encoding configurations, targeting spatial scalability (dyadic spatial scalability and ESS) as well as quality scalability (CGS and MGS). Their evaluations indicate a 10% bitrate overhead of SVC compared to H.264/AVC, presuming no coding penalty at the AVC-compatible base layer of the SVC bitstream. They also discuss low-level encoding configurations, such as hierarchical prediction structures, inter-layer prediction methods, and drift control. Spatial scalability of SVC is discussed and evaluated in [28]. Guidelines for testing conditions of the JSVM for the development of SVC by the Joint Video Team (JVT) are documented in [63]. An improved encoder control is proposed in [64].

Based on the proclaimed 10% bitrate overhead of SVC, a subjective performance evaluation [65] investigated the subjective quality of SVC for three different application areas (mobile broadcast, video conferencing, HD broadcast). For mobile broadcast and video conferencing, the study deployed two-layer SVC configurations featuring spatial scalability and quality scalability separately and compared them to H.264/AVC. The subjective quality ratings confirm the 10% bitrate overhead of SVC. HD broadcasting material with quality scalability was compared to H.264/AVC

encoding at the same bitrate, which also yielded overlapping subjective quality ratings. An important aspect of the study is that it targets different application areas, selecting typical video sequences and appropriate resolutions and bitrates for each application area. However, the study is limited to configurations with two layers, the base layer and one enhancement layer. A survey of subjective SVC evaluations is given in [66]. A broader range of SVC settings is assessed in [67], including a test for the best extraction path (i.e., whether to adapt in spatial, temporal, or quality direction). Each test sequence was encoded with different resolutions and frame rates, covering all spatio-temporal combinations, each layer at the same bitrate. Subjective quality ratings indicated a higher preference of the highest frame rate (60 fps) rather than the highest resolution (4CIF – 704x576), in contrast to objective methods, which had yielded better results for the highest resolution. Both subjective and objective results show that lowest resolutions (QCIF – 176x144) and lowest frame rates (7.5 fps) should be avoided. For further background on these assessments, the interested reader is referred to [68] and [69].

An extensive study of SVC-based adaptation techniques was conducted in [70] and [71], which classifies content based on motion intensity and structural features in order to develop a utility function for adaptation decision-taking and a NALU prioritization scheme. The study also examines viewing preferences w.r.t. spatial resolutions and frame rates at various bitrates. However, it does not directly address SVC layer configurations and their impact on video quality.

The SVC performance of full HD (1080p – i.e., 1920x1080, *p* indicates progressive scan) video sequences was evaluated in a recent study [72]. The RD performance of a fine-granular packet-dropping scheme that consecutively discards MGS enhancement layer NALUs of lower temporal layers was analyzed. The evaluations were conducted on two video sequences with a single SVC encoding configuration using 1 CGS enhancement layer and 3 MGS enhancement layers. The results indicate over 50% bitrate overhead compared to AVC. The authors have attributed the overhead to the bad coding efficiency of the JSVM reference software implementation. As we will show later in this chapter, the overhead was rather caused by the selected encoding configuration (in particular by the number of layers – the proclaimed 10% bitrate overhead apply per layer, not for the entire bitstream). Our tests show that the JSVM exhibits the best coding efficiency of all tested SVC encoders.

The quality and rate variability of CGS and MGS was evaluated in [73] for long-running video sequences (about 54,000 frames per sequence) with various EL configurations. The study concludes that the extraction mechanism for MGS is an important aspect for the video quality and that an extraction mechanism based on priority IDs performs considerably better than extracting MGS layers. Furthermore, the results show 10-30% coding overhead for CGS with two ELs over single-layer encoding and up to 83% for five ELs. Yang and Tang [74] performed an evaluation of SVC configurations featuring 4 to 8 quality layers on CIF (352x288) and 4CIF test sequences. The tests showed unexplained encoder anomalies and unstable RD performance behavior at 8 layers.

Scenarios for the use of SVC in IPTV services are presented in [75] along with an evaluation of the scalability types of SVC. The evaluations comprise objective and subjective test results of CIF and 4CIF test sequences. The work discusses the application of SVC for IPTV and the benefits in terms of content portability, optimized content management and distribution, smart management of access network throughput, and improved QoS/QoE. Studies [76] and [77] investigate the application of SVC for IPTV. A thorough comparison of AVC simulcast and SVC in terms of required capacity is given in [76] by modeling user behavior of IPTV consumption and channel switching. The findings indicate that SVC can reduce required network capacity by around 18% compared to AVC simulcast in some scenarios. Further evaluations of SVC for IPTV with content encoded at resolutions from QVGA (320x240) up to 720p (1280x720) with variable bitrate (VBR) mode are given in [77]. The study investigates several parameters that influence the comparison of SVC vs. AVC simulcast. The study developed a user behavior model for IPTV channel consumption, switching, and selection of representations (i.e., quality versions). It concludes that SVC is best suited for scenarios where most IPTV channels are being requested at most of their quality versions at any given time. Lambert et al. [78] discuss the deployment of SVC with spatial scalability for IPTV. Among others, they point out the options for dealing with different aspect ratios at individual resolutions.

Most performance evaluations use different test sequences and do not provide exact encoding configurations. Those circumstances make comparisons of results between studies very hard. Nevertheless, the studies sketch an overall picture of a coding scheme with around 10% coding overhead per EL, a significant quality drop towards very low resolutions and frame rates, and the advantage of MGS over CGS for quality scalability. The studies mainly focus on content with CIF resolutions, a few go up to 4CIF and 720p resolutions. Although [65] tries to use realistic bitrates, none of the studies performed evaluations based on bitrates used in actual industry solutions. To the best of our knowledge, no considerable research has been conducted to evaluate different SVC encoding configurations for adaptive media streaming of full HD (1080p) content.

One major deployment of SVC is Google+ Hangout [79], a video conferencing tool within the social networking website Google+. The tool uses SVC to enable video delivery to heterogeneous devices and to adapt to the client's network conditions [80]. A measurement study of Google+ Hangout and other video conferencing systems is conducted in [81].

3.2.2 Multi-Bitrate Streaming of Single-Layer Formats

Despite the academic activity and performance studies of SVC, scalable media coding has only recently gained some attention by the industry [82]. In order to establish recommendations for SVC-based video streaming, we take a look at existing industry recommendations for multi-bitrate streaming of single-layer video formats. (Note that in this context a single-layer format is characterized by the lack of

spatial and quality scalability. In other words, we denote AVC as a single-layer format, despite its support of temporal scalability.) Among the most prominent streaming solutions and streaming platforms are: Apple HTTP Live Streaming (HLS) [83], Adobe Dynamic Streaming [84][85], Microsoft Smooth Streaming [86], YouTube [87], Netflix [88], Hulu [89], and MTV [90]. Furthermore, Google+ Hangout [79] and Facebook Video Calling [91] based on a Skype [92] plugin are popular web-based video conferencing tools. Several of these technologies provide recommendations for content encoding: Apple HLS [93], Apple QuickTime [94], Adobe HTTP Dynamic Streaming [95], Adobe Flash Media Server [96], Microsoft Smooth Streaming [97][98], YouTube [99][100], and MTV [97].

In this section, we analyze those recommendations and later on deduce suggestions for SVC streaming.

The spatial resolutions listed in those recommendations range from QCIF (176x144) at bitrates around 50 kbps (even 112x64 for thumbnail display, to be precise) up to 1920x1080 at maximum bitrates around 8 Mbps. The most comprehensive recommendations are for Apple HLS and Adobe Dynamic Streaming.

Apple HLS recommends resolutions around 416x234 for streaming in 16:9 aspect ratio to cellular networks and 640x480 up to 1280x720 for WiFi networks, along with bitrate, frame rate, and profile suggestions [93]. The recommendation contains typically 2 or 3 bitrates per resolution. Furthermore, Apple provides encoding recommendations for QuickTime [101], specifying resolutions, frame rates, and bitrates for different use scenarios [94]. The bitrates suggested for QuickTime (up to 6 Mbps for 720p) are higher than those for HLS (up to 4,500 kbps for 720p). Also, the QuickTime recommendations provide an additional use scenario with a resolution of 1080p.

Adobe provides encoding recommendations for multiple bitrate delivery in Flash Media Server with resolutions from 176x144 up to 1280x720, targeting different categories of connection speeds [96]. Its recommendation lists suggested bitrates (2 bitrates per resolution) and the percentage of US broadband consumers with sufficient bandwidth to support the respective bitrate in 2008. For example, 69% of US consumers were capable of receiving a resolution of 1280x720 at 2,400 kbps. While Adobe Flash Media Server realizes streaming via the proprietary Real-Time Messaging Protocol (RTMP) [102], Adobe also provides comprehensive encoding recommendations for its HTTP Dynamic Streaming solution [95]. The recommendations feature seven different variants of multi-bitrate configurations with resolutions ranging from 256x144 up to 1920x1080. The variants have up to 14 streams with between three and five different resolutions. Note that the recommendations focus on an aspect ratio of 16:9, although for resolutions of 768x432 and below they also present alternatives in 4:3 aspect ratio.

Most encoding recommendations rely on a single frame rate (around 24 to 30 fps) for all content representations. Adobe Flash Media Server recommendations [96] and Apple HLS [93][94] suggest lower frame rates (10-15 fps) for cellular connections at

the lowest resolution. Still, the frame rate shall not be changed during a streaming session.

The investigated bitrate recommendations are compiled in Table 1. For each resolution, the table lists bitrates stated in the recommendations of the investigated industry solutions. The bitrates are represented as follows: If a recommendation states multiple bitrate points, they are written separated by comma (e.g., "8000, 6000, 5500, 5000, 4000"), starting at the highest bitrate. If a recommendation provides a bitrate range, that range is written in the table (e.g., "7000-8000"). All investigated recommendations suggest progressive scan. For Adobe Flash Media Server, only recommendations for 16:9 aspect ratio are included in order not to overload the table. Note that YouTube recommendations for content uploads (represented in gray) are significantly higher than bitrate configurations from other streaming solutions since YouTube prefers to collect content of highest video quality and to transrate it on the server side. At the time of writing no reliable information on streaming bitrates of YouTube VoD content was available. However, YouTube provides recommendations for live streaming. It should also be noted that some recommendations list multiple configurations for multi-bitrate streaming, sometimes with different bitrates for the same resolution. Thus, the number of bitrates in the table does not necessarily reflect the number of streams recommended by a streaming solution. In general, around one to four streams per resolution are suggested.

Table 1: Combined bitrate suggestions for multi-rate streaming of industry solutions [1].

Resolution	Bitrate [kbps]	Streaming solution	Resolution divisibility	Dyadic spatial scalability
1920x1080	6000, 5000	Microsoft Smooth Streaming	mod-8	down
	8000, 6000, 5500, 5000, 4000	Adobe HTTP Dynamic Streaming		
	7000-8000	Apple QuickTime		
	8000-50000 (upload)	YouTube		
1280x960	4500	Apple HLS	mod-16	down
1280x720	3450, 2272, 1672	Adobe Flash Media Server	mod-16	down
	4000, 3500, 3000, 2500, 2000, 1500	Adobe HTTP Dynamic Streaming		
	4500, 2500, 1800	Apple HLS		
	5000-6000	Apple QuickTime		
	3450, 3000, 2100, 1400	Microsoft Smooth Streaming		
	5000-30000 (upload); 2400 (live)	YouTube		
	3500	MTV		
960x540	2250	Microsoft Smooth Streaming	mod-4	up
	1800	Apple HLS		
	2200	MTV		

Resolution	Bitrate [kbps]	Streaming solution	Resolution divisibility	Dyadic spatial scalability
720x486	1072, 672	Adobe Flash Media Server	mod-2	
854x480	2500-15000 (upload); 1000 (live)	YouTube	mod-2	
848x480	1950	Microsoft Smooth Streaming	mod-16	
640x480	1200, 600	Apple HLS	mod-16	up
	1000-2000	Apple QuickTime		
848x440	1950	Microsoft Smooth Streaming	mod-8	
768x432	1740, 1140	Adobe Flash Media Server	mod-16	down
	1700, 1500, 1200, 1000	Adobe HTTP Dynamic Streaming		
	1700	MTV		
736x416	1600	Microsoft Smooth Streaming	mod-16	
720x404	1500	Microsoft Smooth Streaming	mod-4	
640x360	1250	Microsoft Smooth Streaming	mod-8	up
	1200, 600	Apple HLS		
	1000-5000 (upload); 600 (live)	YouTube		
	1200	MTV		
554x304	950	Microsoft Smooth Streaming	mod-2	
400x300	400, 200, 110	Apple HLS	mod-4	
512x288	900	Microsoft Smooth Streaming	mod-16	down
	650, 450, 300	Adobe Flash Media Server		
	1700, 1500, 1200, 900, 600, 450, 300	Adobe HTTP Dynamic Streaming		
	750	MTV		
352x288	372, 268	Adobe Flash Media Server	mod-16	down
448x252	450, 150	MTV	mod-4	
426x240	300 (live)	YouTube	mod-2	
416x234	400, 200, 110	Apple HLS	mod-2	
384x216	400	MTV	mod-8	up
312x176	400	Microsoft Smooth Streaming	mod-8	
288x160	350	Microsoft Smooth Streaming	mod-16	
256x144	300, 250, 150	Adobe HTTP Dynamic Streaming	mod-16	up
176x144	80, 32	Adobe Flash Media Server	mod-16	up
	50-60	Apple QuickTime		
112x64	50	Microsoft Smooth Streaming	mod-16	

Furthermore, Table 1 indicates several characteristics of the listed resolutions. *Resolution divisibility* denotes whether both horizontal and vertical resolution are divisible by 16 (*mod-16*) or any lower power of two (i.e., *mod-8*, *mod-4*, *mod-2*). Since AVC and other common video codecs use macroblock sizes of 16x16 block luma samples [23], resolutions adhering to the mod-16 rule are better suited for optimizing coding performance. Resolutions with lower divisibility typically require the encoder to pad the last macroblocks. Note that some encoders, e.g., the bSoft SVC encoder [103], try to optimize coding performance by removing those incomplete macroblocks, thus cropping a small part of the video. The column labeled *dyadic spatial scalability* marks those resolutions for which another resolution in the table either has half the horizontal and half the vertical resolution of the first (dyadic downscaling) or has double the horizontal and double the vertical resolution of the first (dyadic upscaling).

Less than half of the 26 resolutions in Table 1 adhere to the mod-16 rule. Only six resolutions meet the dyadic downscaling criterion, the same goes for dyadic upscaling, but none meets both criteria. This means that the listed resolutions would not support SVC encoding with three dyadic spatial resolutions. It can also be observed that the CIF resolution (352x288), which is commonly used in research literature, is only used in the encoding recommendations for Adobe Flash Media Server; most other streaming solutions prefer 512x288, which has a wider aspect ratio of 16:9. None of the recommendations lists the 4CIF resolution (704x576), which is also often used in research literature. While most of the listed resolutions have aspect ratios around 16:9, some lower resolutions have narrower aspect ratios, CIF and QCIF (176x144) resolutions having the narrowest aspect ratio of 11:9.

As a final remark, Table 1 shows that major industry streaming solutions use lots of different resolutions, often with slight discrepancies across these systems. Resolutions 1280x720 and 1920x1080 are common to most platforms, but at lower resolutions, both the exact resolution and aspect ratio are different across platforms. Since all recommendations target single-layer formats, the support of dyadic spatial scalability is irrelevant in their scenarios, which is reflected by the choice of recommended resolutions.

3.3 Test-bed Setup

3.3.1 Deduced Bitrate Suggestions

Based on the encoding recommendations of industry solutions for multi-bitrate streaming (Section 3.2.2), this section devises guidelines for AVC-based multi-rate streaming and for SVC streaming. In order to create guidelines viable for both research and industrial deployment, the criteria for the selection of resolutions and bitrates are as follows. First, the plethora of resolutions shall be boiled down to at most 7 resolutions. Each resolution should allow dyadic upscaling or downscaling.

The bitrates shall be distributed properly to ensure that video quality increases in constant steps. We argue that 4 bitrates per resolution are sufficient for most use cases. Thus, we devise bitrate suggestions for 2 and 4 bitrates per resolution. This provides a baseline from which other numbers of bitrates can easily be interpolated. Note that a higher number of bitrates should typically span a slightly wider bitrate range as well.

Table 2 comprises a list of suitable bitrates for typical resolutions. These guidelines take the popularity of resolutions among streaming solutions, top and bottom bitrates, as well as bitrate steps into account. We placed special emphasis on assembling meaningful resolutions, reducing the number of different resolutions from Table 1. Although none of the industrial solutions lists the 4CIF resolution in their recommendations, we included it to enable dyadic spatial resolutions from QCIF to CIF up to 4CIF. The bitrate suggestions for 4CIF are interpolated from other suggestions. With the exception of CIF and 4CIF resolutions, each of the listed resolutions is mentioned in at least two streaming solution recommendations. The resolution 512x288 was removed in favor of CIF (352x288).

The bitrate recommendations of industry solutions and the deduced bitrate suggestions for 2 bitrates are shown in Figure 7. Resolutions are sorted by the number of pixels. Our suggestions for resolution 352x288 are based on the respective industry recommendations for 512x288. From the figure, we can observe how the highest bitrate of resolution $n-1$ compares to the lowest bitrate of resolution n for the different recommendations. Only the recommendations for Apple QuickTime and YouTube live streaming consequently increase the bitrates for such resolution changes. For all other recommendations, the bitrates overlap from one resolution to the next at least to some extent.

Table 2 focuses on two and four bitrates per resolution. For those resolutions, for which Table 1 does not list sufficient different bitrates, the column for *4 bitrates* in Table 2 was left blank. Suggested bitrates at 4 streams for resolutions 960x540 and 640x360 were also interpolated from other resolutions. It should be mentioned that the table does not take characteristics and requirements of SVC into account. In

Table 2: Derived guidelines for bitrates in AVC-based multi-rate streaming.

Resolution	Bitrate suggestions	
	4 bitrates [kbps]	2 bitrates [kbps]
1920x1080	8000, 6000, 5000, 4000	8000, 5500
1280x720	6000, 4000, 2500, 1500	4500, 2500
704x576		2000, 1225
960x540	2700, 2250, 1800, 1200	2250, 1800
640x360	1600, 1250, 900, 600	1600, 600
352x288	1500, 900, 450, 270	1200, 300
176x144		100, 50

particular, it does neither consider SVC coding overhead nor is it optimized for dyadic spatial scalability. When adjusting the bitrates for coding overhead of SVC, one must consider both targeted quality and connection speed for streaming. We assume that the recommendations of industry streaming solutions are based on real-life scenarios that take typical network conditions into account. Simply increasing the bitrate to maintain the same quality as AVC might potentially exceed the network bandwidth in some cases if the coding overhead is too high. On the other hand, simply keeping the same bitrate decreases the video quality. Scientific literature typically assumes a coding overhead of 10% [27][33] compared to AVC. Note that this overhead is based on a single enhancement layer. Each additional enhancement layer requires another 10% overhead. We reckon that the bitrate can be increased accordingly (i.e., by around 10% for the first enhancement layer, by around 20% for the second, and so on), considering three relevant aspects. First, the use of SVC enables dynamic bitrate adaptation, alleviating the risk of stalling. Second, we assume that the recommendations of industry streaming solutions include a safety factor towards higher bitrates. That is, we expect that their bitrate recommendations are designed so that streaming does not have to operate on the limit of network resources. Third, network traffic forecasts [104] and broadband surveillance reports [105][106][107] shown a continuous increase of video network traffic and connection speeds that would easily accommodate the proposed bitrate increase.

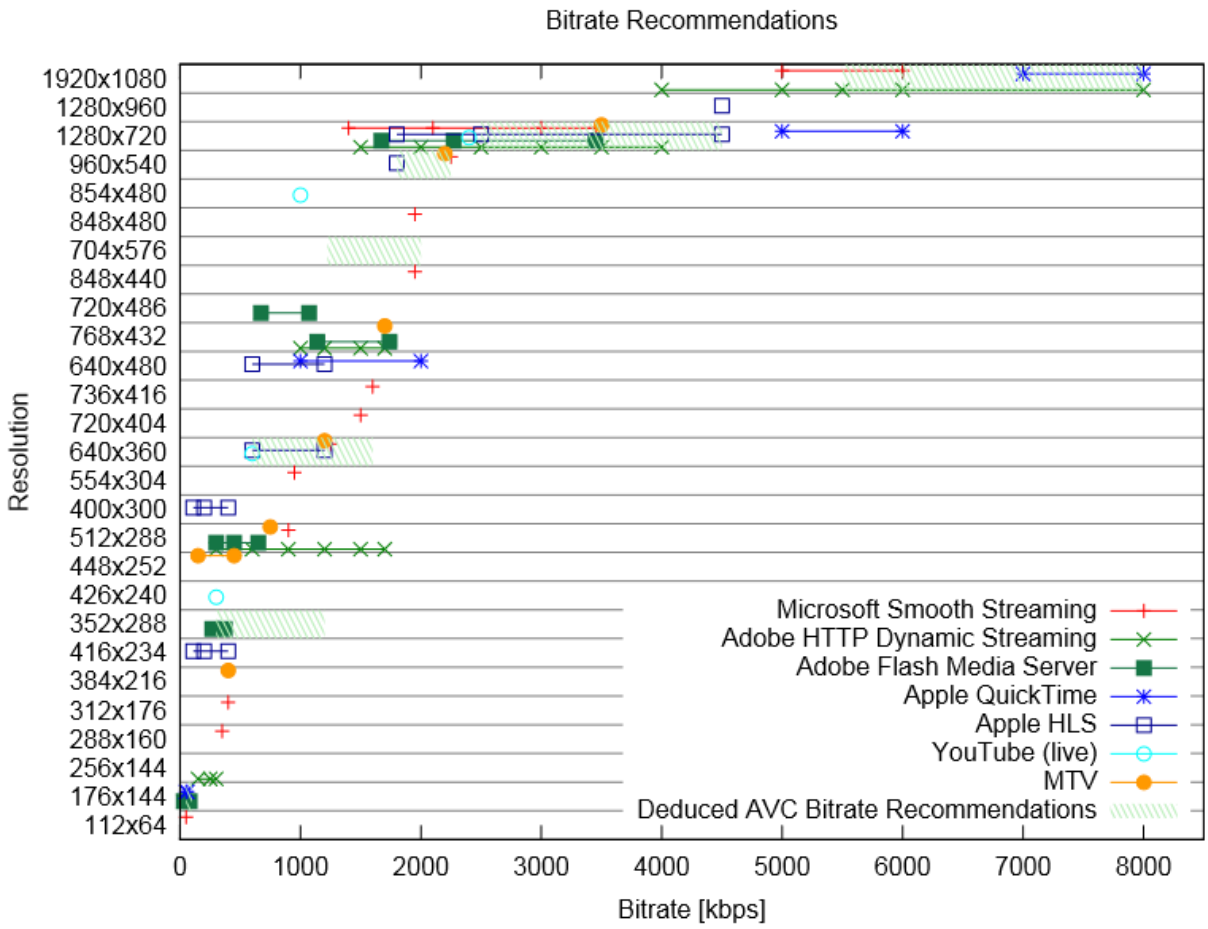


Figure 7: Bitrate recommendations of AVC-based streaming solutions and deduced suggestions.

We further adjust the bitrate recommendations for SVC in Table 3. For streams with 2 layers, we propose to add 10% overhead for both bitrates compared to Table 2 (with some rounding where appropriate). However, with 4 layers, we keep the original bitrate for the base layer in order to support low bandwidths, increase the bitrate for the first enhancement layer by 10%, for the second by 20% and for the third by a total of 30%. As mentioned before, we added the 4CIF resolution (704x576) in order to better support dyadic spatial scalability. The table also indicates whether dyadic spatial scalability (up- or downscaling) is supported by the listed resolutions.

Depending on the scenario and targeted client devices, we suggest streams with a total of six to twelve extraction points out of the possible combinations in Table 3, ranging over 2 to 4 resolutions. We also suggest allocating proportionally more bitrates per resolution for higher resolutions. For example, a configuration may contain the four bitrates indicated for 1920x1080 and two bitrates for 960x540.

3.3.2 SVC Encoders and Evaluation Metrics

Besides the reference software, *JSVM*, several proprietary SVC encoders exist. To our knowledge, the most prominent ones are *MainConcept* [108], *VSS* [109], and *bSoft* [103].

Note that the encoders exhibit many different encoding configuration options and yield individual bitstream characteristics. Occasionally, they have differing interpretations of various coding concepts, and are of varying stability. While the MainConcept and VSS encoders use requantization for MGS layers, the bSoft encoder distributes transform coefficients automatically across layers (also known as MGS vectors). The JSVM encoder supports both behaviors (i.e., requantization and manual distribution of transform coefficients) [110]. The MainConcept, VSS, and bSoft encoders provide configuration options for constant bitrate (CBR) mode. However, the tested version of the MainConcept encoder was only able to encode in CBR mode at a few specific configurations. The bSoft encoder requires an initial QP value even for CBR encoding. We noticed that fixed QP settings always yielded better RD performance than any CBR setting with that initial QP. Thus, only fixed QP rate control was used for the bSoft encoder. Only the VSS encoder supported CBR at all resolutions and applied configurations. In contrast to the other encoders, VSS has a different approach for extracting layers from an SVC bitstream as further detailed in Section 3.5.2.2. Furthermore, bitstreams encoded by one encoder (e.g., VSS or bSoft) are not necessarily decodable by another decoder (e.g., JSVM). Also, the JSVM tool set has one tool (called *BitStreamExtractor*) for adapting SVC bitstreams and another tool for decoding, while the other encoders couple adaptation

Table 3: Adjusted bitrate recommendations for SVC streaming [1].

Resolution	Bitrate suggestions		Dyadic spatial scalability
	4 bitrates [kbps]	2 bitrates [kbps]	
1920x1080	10400, 7200, 5500, 4000	8800, 6050	down
1280x720	7800, 4800, 2750, 1500	5000, 2750	down
704x576		2200, 1350	down
960x540	3500, 2700, 1975, 1200	2475, 1980	up
640x360	2075, 1500, 990, 600	1760, 660	up
352x288	1950, 1080, 500, 270	1320, 330	up & down
176x144		110, 55	up

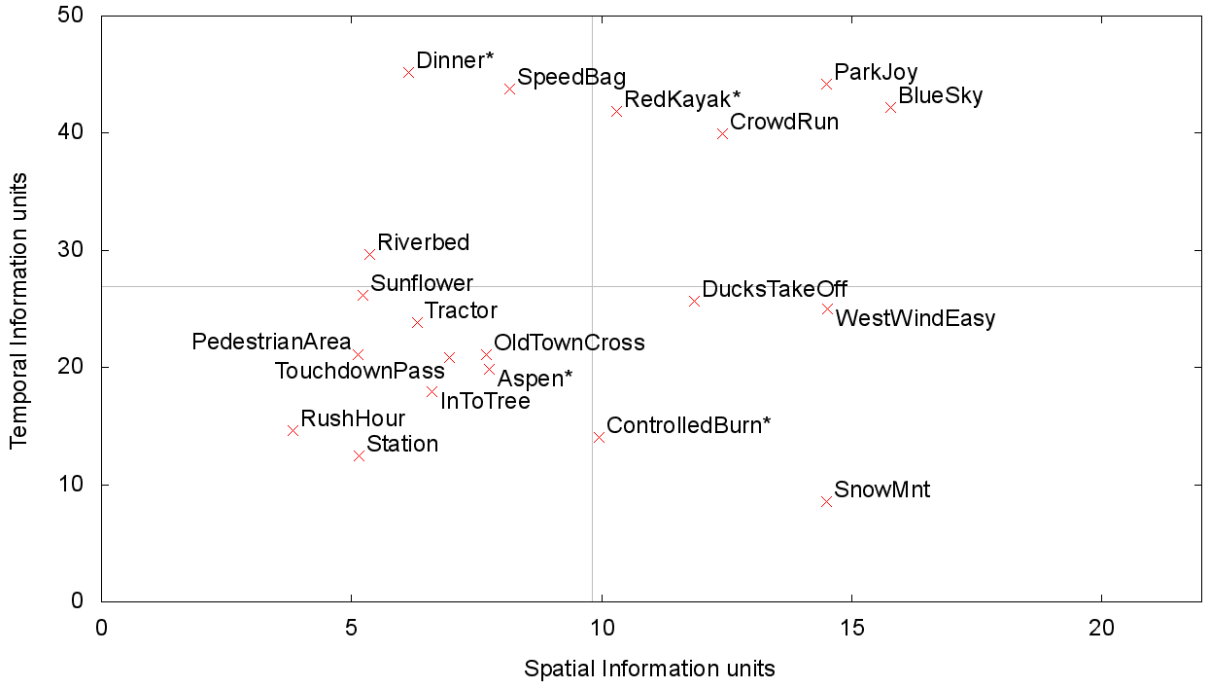


Figure 8: Spatial-Temporal plot for test sequences.

and decoding into a single tool. Performance tests of all these encoders will be presented throughout this chapter.

Unless noted otherwise, the resolution was set to 1920x1080 and the deltaQP (dQP) for requantization between MGS layers was set to 2. For example, the dQP of 2 denotes QPs of MGS layers (from highest to lowest layer) of 28, 30, 32, and 34. The entropy coding mode was set to CABAC. We used a fixed interval of 32 frames for IDR frames. While MainConcept and VSS encoders support scene change detection, where IDR frames are inserted dynamically, we used a fixed IDR frame interval to ensure consistency with other encoders.

Peak Signal-to-Noise Ratio (PSNR) is one of the most widely used full reference metrics for objective video quality assessment due to its simplicity and its low computational requirements. A possible PSNR to Mean Opinion Score (MOS) conversion was given in [111] and subsequently used in [112], [113], [114], [115] and others. But to the best of our knowledge, no evaluation on the actual correlation of that particular mapping is available. Another mapping table with different PSNR values was proposed in [116] based on correlation evaluations on still images in [117].

The NTIA Video Quality Metric (VQM) [118][119] is a standardized full-reference objective method. VQM compares an original video sequence to a distorted sequence in order to estimate the video quality by combining perceptual effects of several video impairments such as blurring, jerky/unnatural motion, global noise, block distortion, and color distortion. The VQM output describes the distortion of a video on a scale from 1 (high distortion) to 0 (no distortion). The VQM results can be mapped to the MOS scale as shown in Table 4. VQM was specifically designed to

correlate better with the human visual system than PSNR [120][121][122][123][124], therefore, we also use VQM results in addition to PSNR in our performance tests.

3.3.3 Selection of Test Sequences

We selected four different video sequences for our evaluations. Video content can be characterized by its Spatial Information (SI) – i.e., amount of structural features – and Temporal Information (TI) – i.e., amount of motion – as defined in [125]. Both aspects have impact on the encoding process (e.g., encoding duration and RD performance). The Xiph.Org Foundation provides a collection of test sequences at various resolutions [126]. The collection comprises 20 sequences with resolutions of 1920x1080 or above (not counting three full movies). We analyzed those sequences in order to select appropriate sequences that represent different SI and TI characteristics.

The SI and TI of all sequences are shown in Figure 8. The first 250 frames of each sequence were used in order to have uniform durations and because longer sequences caused the VQM software to crash during quality evaluations. The *BlueSky* sequence only has 217 frames. The *Dinner* sequence is a computer-generated video. Sequences with resolutions above 1920x1080 were downsampled to 1920x1080, sequences with frame rates of 50 fps were downsampled to 25 fps. Note that TI computes the differences between frames and selects the maximum value. Some sequences contain multiple shots. TI values for scene cuts were removed in accordance with [125] before computing the maximum value (indicated via *). Fade-overs were not removed.

Based on the results, we selected the following test sequences: *PedestrianArea* (low SI, low TI), *Dinner* (low SI, high TI), *DucksTakeOff* (high SI, low TI), and *CrowdRun* (high SI, high TI). Snapshots of the four test sequences are shown in Figure 9. The *Dinner* sequence has a frame rate of 30 fps, the other sequences have 25 fps. *DucksTakeOff* and *CrowdRun* have original resolutions of 3840x2160 at 50 fps. One factor of the test sequence selection was the depiction of different sceneries, such as people and faces in medium shot (*PedestrianArea*), people in wide shot (*CrowdRun*), animals (*DucksTakeOff*), and synthetic scenes (*Dinner*). As our goal was to assess SVC configurations for typical media streaming purposes, we decided to avoid

Table 4: Mapping of VQM results to MOS.

VQM	MOS
0.0 – 0.2	5 (Excellent)
0.2 – 0.4	4 (Good)
0.4 – 0.6	3 (Fair)
0.6 – 0.8	2 (Poor)
0.8 – 1.0	1 (Bad)

(a) (b)

(c) (d)

Figure 9: Snapshots of (a) *PedestrianArea*, (b) *Dinner*, (c) *DucksTakeOff*, and (d) *CrowdRun* sequences.

extreme cases (e.g., *BlueSky* or *SnowMnt* sequences) in favor of having different sceneries.

Based on the deduced coding suggestions, the following sections provide performance evaluations of several SVC configurations. It is important to note that the following performance results are implementation-dependent and provide a volatile snapshot of current SVC encoder performances. Nevertheless, we strive to highlight performance characteristics of SVC as a coding scheme that we expect to remain valid beyond the mere comparison of encoder implementations.

3.4 High-Definition SVC Encoding Performance for Adaptive Media Streaming

In this section, extensive performance evaluations of SVC with a focus on 1080p resolutions are presented, including various SVC configurations and different encoders (JSVM, MainConcept, VSS, and bSoft). The goals of these evaluations are (1) to provide RD performance results in terms of PSNR and VQM, (2) to investigate various encoding configurations, (3) to highlight the characteristics of different encoders, and (4) to validate the encoding recommendations devised in Section 3.3.1.

We first evaluate rate control modes (i.e., constant bitrate vs. fixed QP) for different encoders in order to compare their RD performance and to validate whether the devised bitrate recommendations yield consistent qualities at all resolutions. Then, we test the combination of spatial and quality scalability to decide whether to encode one stream per resolution or all resolutions in one stream for media streaming scenarios. Another factor to adaptive streaming configurations is the number of quality layers for a given resolution, which affects the flexibility of adaptations at the cost of coding overhead. This aspect is evaluated with 1 to 4 quality layers for various encoders. Finally, we investigate the impact of requantization on the RD performance, which controls the bitrate distances between quality layers.

3.4.1 Rate Control Modes

As a first evaluation in our set of tests, we validate the bitrate recommendations discussed in Section 3.3.1, comparing RD performance of rate control modes (CBR and fixed QP) of several encoders.

The configurations for this test are as follows. For each resolution (from 1920x1080 down to 176x144), bitstreams were encoded with 2 MGS layers. In CBR mode, target bitrates were set to the values stated in Table 3 (for two bitrates). For encoding with fixed QP, we selected for each sequence the two QPs that resulted in bitrates just above and just below the target bitrate of the enhancement layer for the respective resolution in Table 3 (for two bitrates). As mentioned in the base test configuration description in Section 3.3.2, the dQP between MGS layers was set to 2.

We note that this static dQP of 2 can cause the bitrate of the base layer to deviate from the suggested target bitrate in some cases, but we argue that this static dQP makes the results better comparable to the bSoft encoder and to the results of our other tests. Our results also show that the chosen dQP fits surprisingly well for most resolutions and respective target bitrate suggestions.

The tested version of the MainConcept encoder has some limitations concerning supported bitrates for CBR mode. Therefore, we were only able to obtain results for 1920x1080 with target bitrates of 4,400 kbps for the base layer and 8,800 kbps for the enhancement layer.

The JSVM encoder was only evaluated for a resolution of 1920x1080 at fixed QP mode. Although the tested version of the JSVM provides basic CBR support, it only supports CBR mode at the base layer, making it unsuitable for our tests.

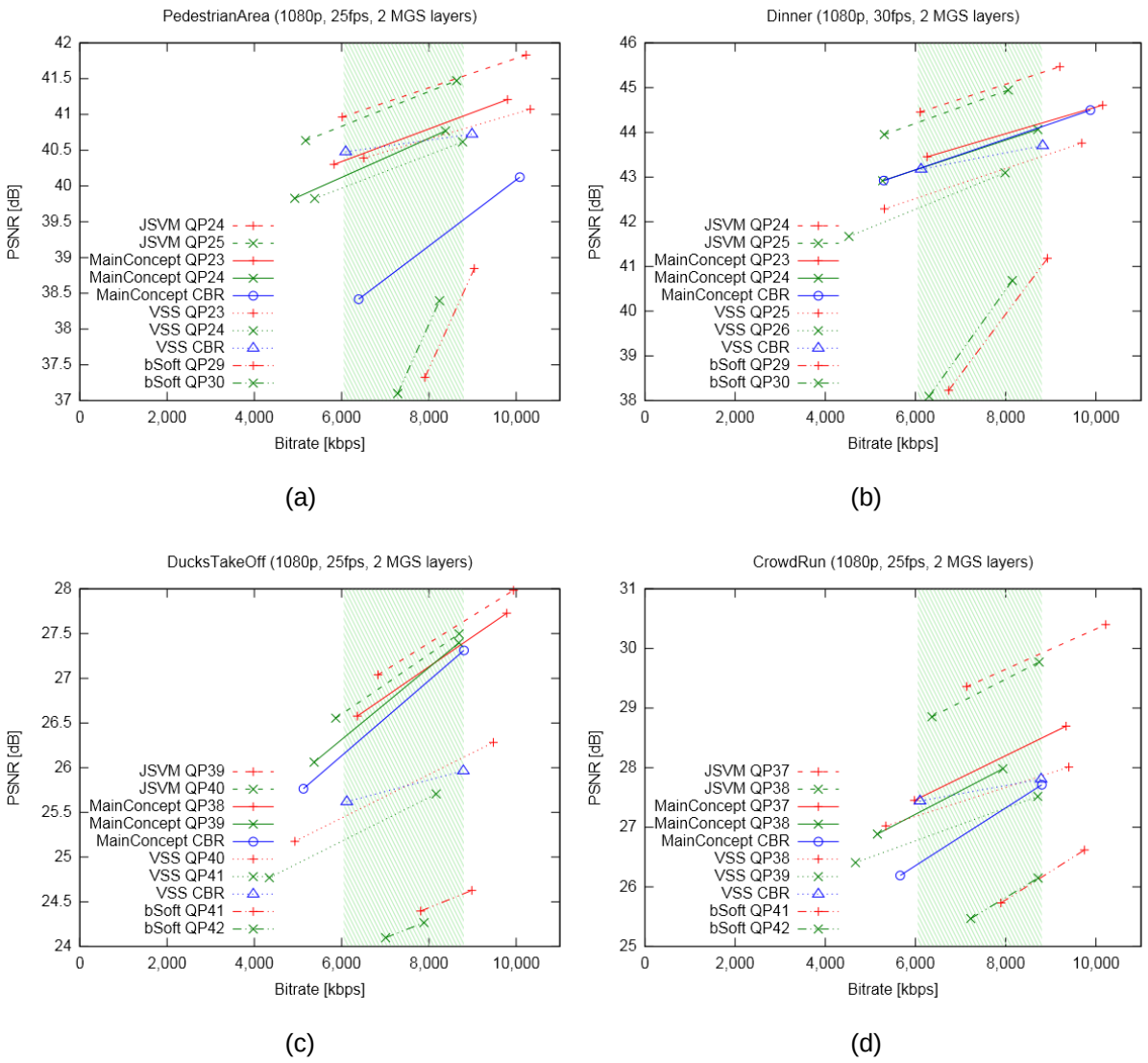


Figure 10: PSNR results of rate control modes for different encoders for (a) *PedestrianArea*, (b) *Dinner*, (c) *DucksTakeOff*, and (d) *CrowdRun* sequences [1].

Configurations for all encoders in fixed QP mode are provided in Annex B.

The PSNR results at a resolution of 1920x1080 are shown in Figure 10. The bitrate ranges from the suggestions in Table 3 (for 2 bitrates) are indicated as green background. The corresponding VQM results are given in Figure 11. Note that the y-axis of VQM results is an impairment scale from 1 (high distortion) to 0 (no distortion), indicating the expected quality of a sequence. In contrast to PSNR results, where the range of the y-axis is dynamically adjusted to the results, graphs for VQM are always shown for the entire y-axis range from 1 to 0 in order to better indicate the overall expected quality instead of quality changes. Different line types are used for the encoders for fixed QP mode.

In terms of encoder comparison, the JSVM outperforms the other encoders with respect to RD performance, followed by MainConcept and VSS. The bSoft encoder has somewhat lower PSNR results and shows a sharp decrease of PSNR for the base layer, but the VQM results show that the quality decrease of the bSoft encoder

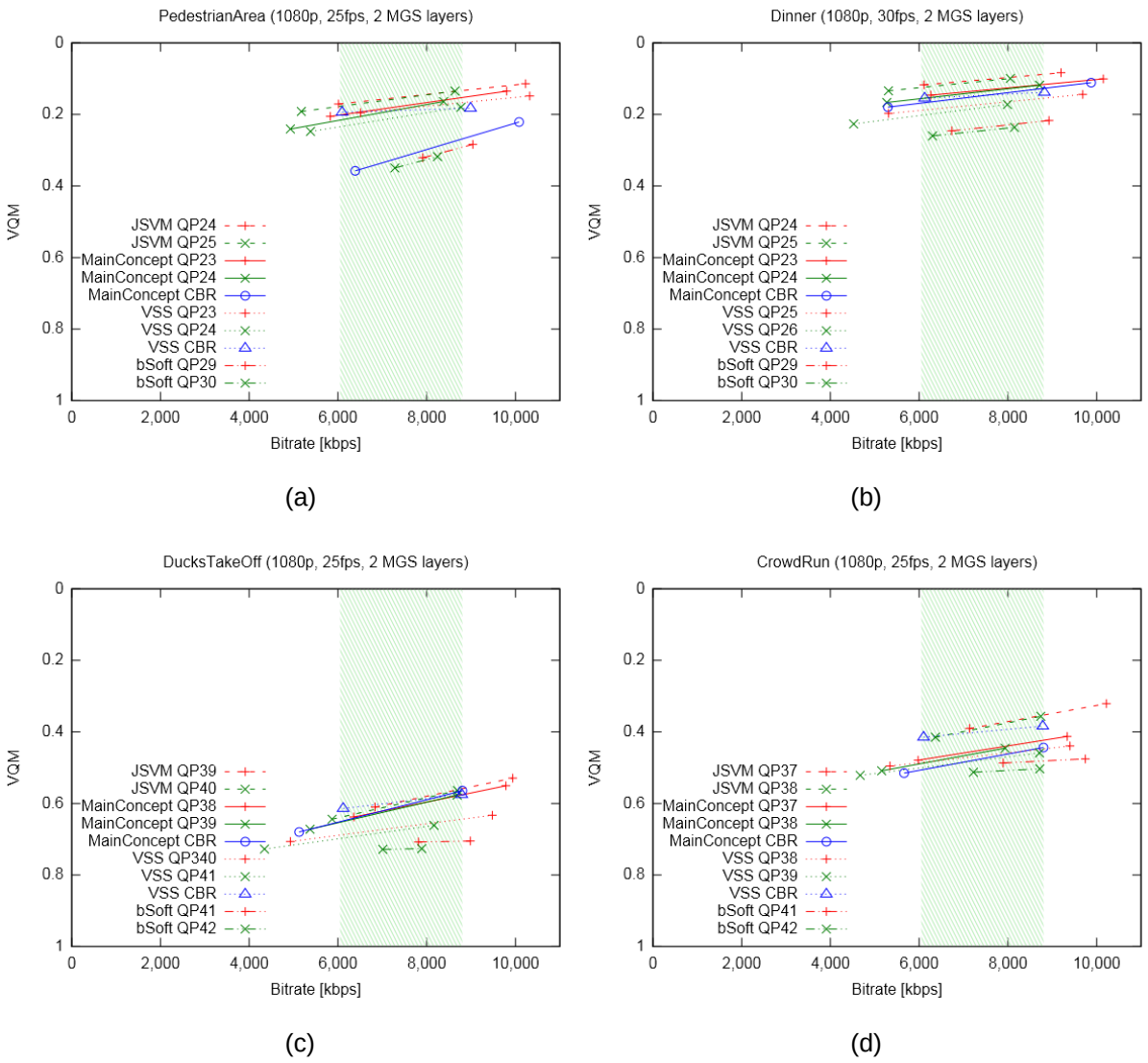


Figure 11: VQM results of rate control modes for different encoders for (a) *PedestrianArea*, (b) *Dinner*, (c) *DucksTakeOff*, and (d) *CrowdRun* sequences [1].

towards the base layer is comparable to – if not lower than – the decrease of other encoders. Especially for sequences with high SI, the VQM results are on par with the other encoders. Since VQM correlates better with the human visual system, these results suggest that the actual visual quality of the bSoft encoder is significantly higher than indicated by the corresponding PSNR values. Similar to the behavior of the bSoft encoder, CBR modes of the MainConcept and VSS encoders tend to have better VQM than corresponding PSNR results.

From the tested sequences we conjecture that SI has a higher coding efficiency than TI. In particular, the *DucksTakeOff* and *CrowdRun* sequences have low PSNR results. However, the number of test sequences does not allow conclusive inference.

When comparing rate control modes, we see that the MainConcept encoder achieves higher quality in fixed QP mode than in CBR mode. In contrast, the VSS encoder yields equal or slightly lower quality in fixed QP mode compared to CBR mode.

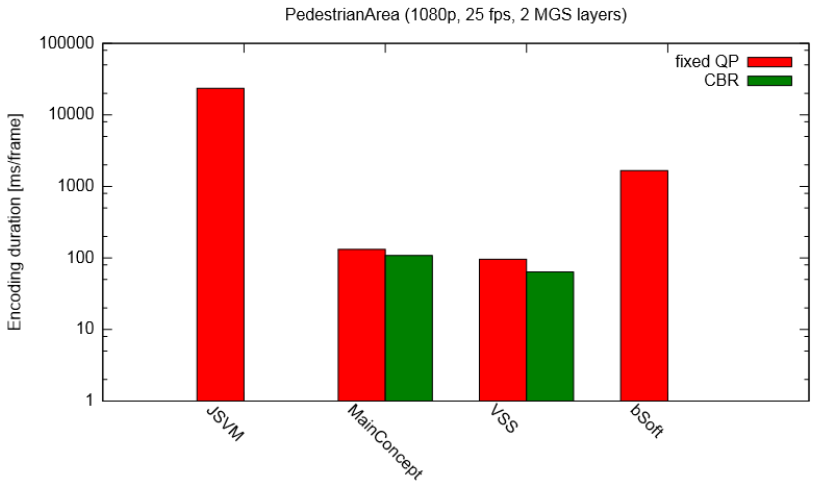


Figure 12: Encoding durations of rate control modes for different encoders for the *PedestrianArea* sequence.

Among the tested encoders and rate control modes, the VSS encoder in CBR mode shows the lowest decrease of RD performance towards the base layer.

Whether the MainConcept or VSS encoder gives better RD performances in CBR mode highly depends on the content. Similar to the already observed behavior of the bSoft encoder, CBR modes of both encoders tend to have better VQM than corresponding PSNR results. For example, PSNR results of the *CrowdRun* sequence in Figure 10 (d) show that the RD performance of the VSS encoder in CBR mode (labeled *VSS CBR*) is clearly below that of the MainConcept encoder at QP=37 (labeled *MainConcept QP37*), whereas the VQM results in Figure 11 (d) show the opposite. We can thus conclude that (for a given sequence) PSNR and VQM results correlate to some extent for JSVM, MainConcept and VSS encoders in fixed QP mode, while the bSoft encoder in fixed QP mode and MainConcept and VSS encoders in CBR mode yield better VQM results compared to PSNR.

Encoding durations per frame of the different encoders are exemplarily shown in Figure 12 for the *PedestrianArea* sequence. Due to the very low encoding speed of the JSVM, results are depicted on a logarithmic scale. The MainConcept and VSS encoders are about two orders of magnitude faster than the JSVM encoder, while the bSoft encoder is roughly one order of magnitude faster than the JSVM. Interestingly, CBR mode is slightly faster for both MainConcept and VSS encoders. A more detailed breakdown of encoding durations is provided in Section 3.4.6.

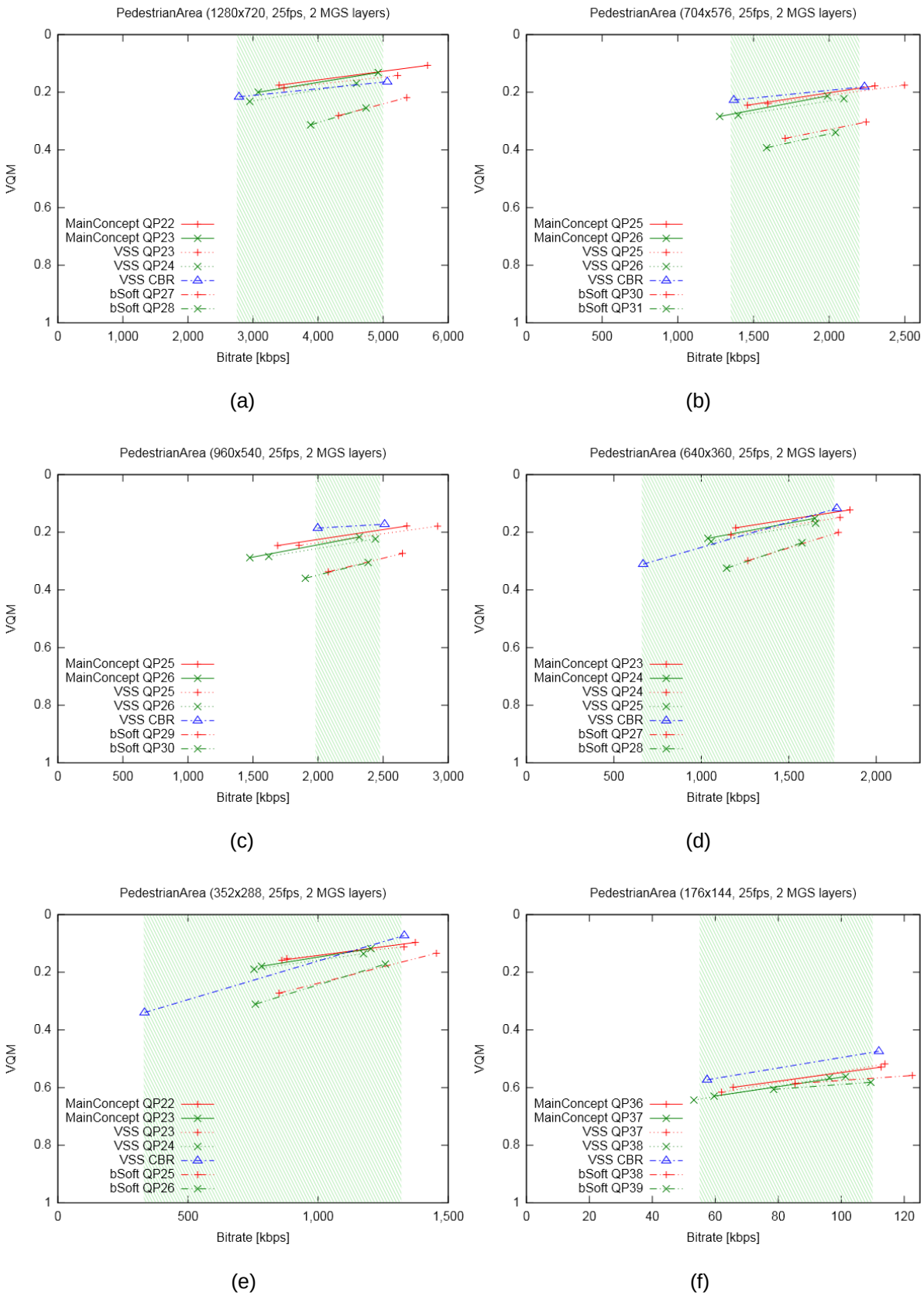


Figure 13: VQM results of rate control modes for different encoders for *PedestrianArea* sequence at (a) 1280x720, (b) 704x576, (c) 960x540, (d) 640x360, (e) 352x288, and (f) 176x144 resolutions [1].

VQM results for lower resolutions are presented for the *PedestrianArea* sequence in Figure 13. Further test results for lower resolutions are shown for the *PedestrianArea* sequence in Figure 77 (PSNR), as well as for the *CrowdRun* sequence in Figure 78 (PSNR) and Figure 79 (VQM) in Annex C. Due to the high number of resolutions and resulting figures, we selected only the sequences with lowest and highest spatio-temporal complexities (i.e., *PedestrianArea* and *CrowdRun*). Again, the suggested bitrate ranges from Table 3 (for 2 bitrates) are indicated as green background.

The suggested bitrates yield quite constant and good qualities across all resolutions, except for the lowest resolution 176x144 (QCIF). The rationale behind the low target bitrate suggestions for QCIF is to enable video transmission even for very low bandwidths. QCIF is intended for miniature preview or as a fallback solution in case of bad connectivity, i.e., cases in which the end user prefers low quality over no video playback at all.

As already observed for 1920x1080, the VSS encoder in CBR mode and the bSoft encoder (in fixed QP mode) perform consistently better in terms of VQM results compared to PSNR results. For the *CrowdRun* sequence the VSS encoder in CBR mode has best VQM results for all resolutions and is almost on par with JSVM performance in Figure 11 (d).

It can also be noticed that the differences between encoders in terms of RD performance decrease for more complex sequences (such as *CrowdRun*).

3.4.2 Combination of Spatial Scalability and MGS

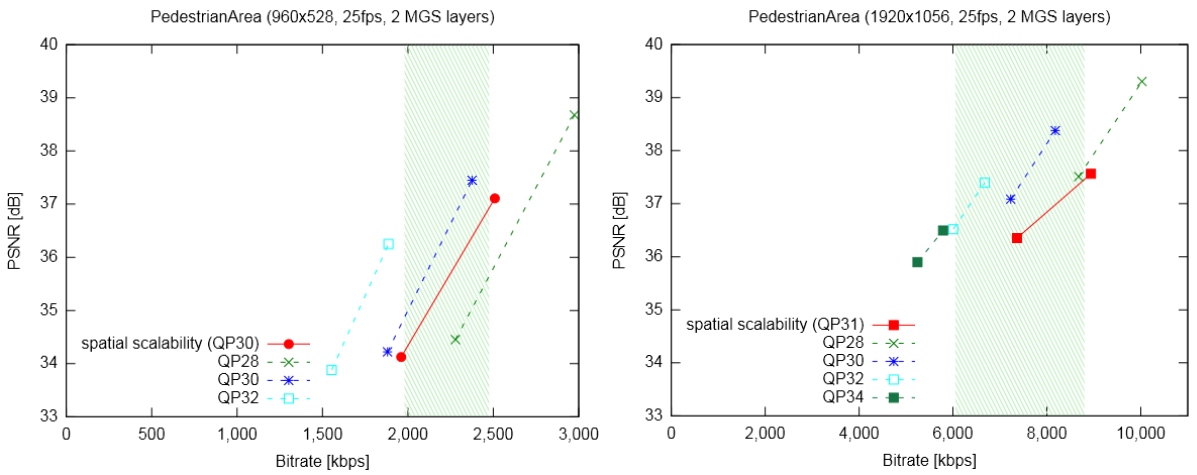
In the following test we investigate the RD performance of spatial scalability at two resolutions combined with two MGS layers. We compare the RD performance to bitstreams without spatial scalability with two MGS layers at either resolution.

This configuration is also relevant for determining whether to use one SVC bitstream for multiple resolutions or to use separate SVC bitstreams featuring quality scalability for each resolution in SVC streaming scenarios.

The tested version of the bSoft encoder supports spatial scalability only for dyadic resolutions, i.e., extended spatial scalability (ESS) is not supported. Thus, we use the following configurations in our tests:

- Resolution 1: 960x528;
- Resolution 2: 1920x1056, each resolution with 2 MGS layers.

Note that the bSoft encoder requires that the vertical resolution be divisible by 16 (known as "mod-16"), which is the reason for the slightly cropped vertical resolutions in this test. For this test, we aimed for bitrates conforming to the recommendations of Table 3. The PSNR results are shown in Figure 14 for the *PedestrianArea* sequence and in Figure 15 for the *CrowdRun* sequence. Note that Figure 14 (a) and Figure 15 (a) show extraction points for resolution 960x528, while Figure 14 (b) and Figure 15



(a) (b)

Figure 14: PSNR results for spatial scalability of the bSoft encoder for the *PedestrianArea* sequence. The line labeled *spatial scalability* represents a single bitstream ranging over both resolutions (a) 960x528 and (b) 1920x1056 [1].

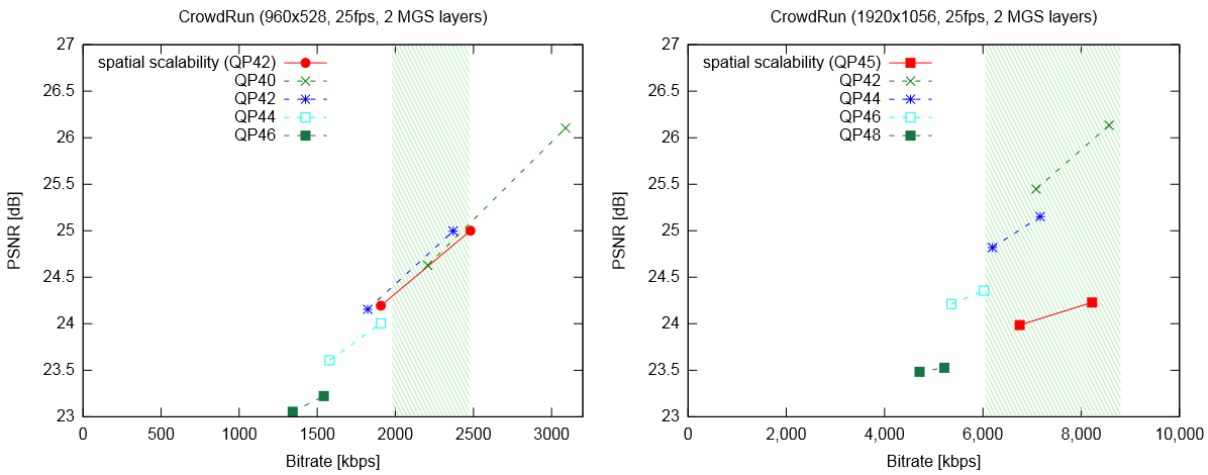
(b) show extraction points for resolution 1920x1056. Note that the lines labelled *spatial scalability* range over both resolutions, i.e., the lines represent the two resolutions of a single bitstream.

The bitstream with spatial scalability has only a small overhead for the lower resolution (i.e., extra bits that enable the upscaling prediction). Since the layers of the higher resolution depend on the lower resolution ones, the RD performance at 1920x1056 is worse than for bitstreams without spatial scalability. To achieve the same quality, the single-resolution bitstreams need around 18% to 26% less bitrate for *PedestrianArea* and 25% to 35% less bitrate for *CrowdRun*. Conversely, a single-resolution bitstream of the same bitrate as the spatial scalability bitstream achieves roughly 1-1.5 dB higher PSNR at 1920x1056. It requires almost the same disk space to store the spatial scalability bitstream or two separate bitstreams for the two respective resolutions. Due to the high quantization used for the *CrowdRun* sequence to achieve the recommended bitrates, the PSNR results are very low.

The bitstream with spatial scalability loses slightly less quality between layers as indicated by the lower slopes of the lines representing spatial scalability compared to single-resolution bitstreams in Figure 14 (b) and Figure 15 (b), resulting in smoother in-network adaptation. However, we consider the bitrate overhead to be a more relevant factor in favor of using separate SVC bitstreams for each resolution.

3.4.3 Number of MGS Layers

The following test investigates the impact of the number of SVC layers in MGS mode on the RD performance. Intuitively, higher numbers of layers come with some bitrate penalties. We tested the JSVM, MainConcept, and bSoft encoders with the following

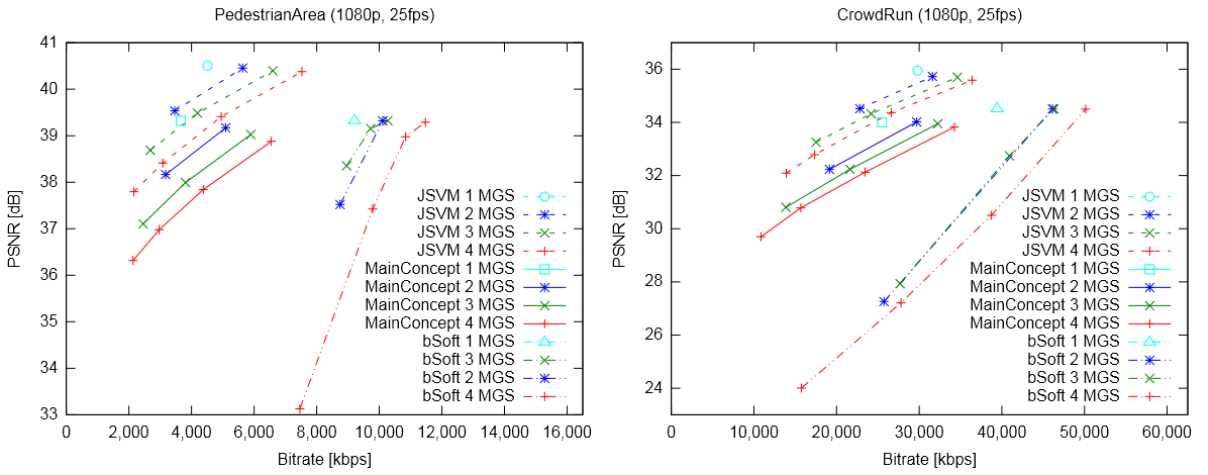


(a) (b)

Figure 15: PSNR results for spatial scalability of the bSoft encoder for the *CrowdRun* sequence. The line labeled *spatial scalability* represents a single bitstream ranging over both resolutions (a) 960x528 and (b) 1920x1056.

configuration: the QP of the highest layer was set to 28. For the MainConcept encoder, dQP was set to 2.

Figure 16 shows the PSNR results of 1 to 4 MGS layers for (a) *PedestrianArea* and (b) *CrowdRun* sequences. Results for the VSS encoder are similar to the results for the MainConcept encoder (although at slightly higher bitrates) but are not included in order not to overload the figures. The JSVM and MainConcept encoders exhibit rather constant decrease in RD performance for higher number of layers. The results for the bSoft encoder show that the bitstreams with 2 and 3 MGS layers (labeled *bSoft 2MGS* and *bSoft 3MGS* respectively) have almost the same RD performance. For the *PedestrianArea* sequence, the base layer of *bSoft 2MGS* even has lower bitrate and PSNR than the base layer of *bSoft 3MGS*.



(a) **(b)**

Figure 16: PSNR results for varying number of MGS layers for different encoders, for (a) *PedestrianArea* and (b) *CrowdRun* sequences [1].

Table 5: Relative bitrate penalties for additional MGS layers.

PedestrianArea				
Bitrate penalty	**JSVM**	**MainConcept**	**VSS**	**bSoft**
1MGS to 2MGS	25.0%	39.3%	30.1%	9.8%
2MGS to 3MGS	17.1%	15.7%	21.7%	1.6%
3MGS to 4MGS	14.0%	11.1%	16.4%	11.7%
Average	**18.7%**	**22.0%**	**22.7%**	**7.7%**
Dinner				
Bitrate penalty	**JSVM**	**MainConcept**	**VSS**	**bSoft**
1MGS to 2MGS	21.7%	31.9%	43.4%	10.6%
2MGS to 3MGS	17.0%	14.8%	22.7%	4.7%
3MGS to 4MGS	13.9%	10.8%	15.5%	9.9%
Average	**17.5%**	**19.1%**	**27.2%**	**8.4%**
DucksTakeOff				
Bitrate penalty	**JSVM**	**MainConcept**	**VSS**	**bSoft**
1MGS to 2MGS	0.4%	15.2%	14.9%	19.3%
2MGS to 3MGS	9.2%	6.9%	12.6%	0.0%
3MGS to 4MGS	1.4%	5.0%	10.6%	5.4%
Average	**3.7%**	**9.0%**	**12.7%**	**8.2%**
CrowdRun				
Bitrate penalty	**JSVM**	**MainConcept**	**VSS**	**bSoft**
1MGS to 2MGS	6.1%	16.4%	23.7%	17.0%
2MGS to 3MGS	9.5%	8.6%	14.6%	0.3%
3MGS to 4MGS	5.1%	6.2%	10.7%	8.3%
Average	**6.9%**	**10.4%**	**16.3%**	**8.6%**
Total Average	**11.7%**	**15.2%**	**19.7%**	**8.2%**

The PSNR results of the highest layers remain relatively static across the number of MGS layers for all encoders, even though they slightly decrease for MainConcept. Instead, encoders allocate less quality to the base layers for each additional MGS layer due to the applied configuration.

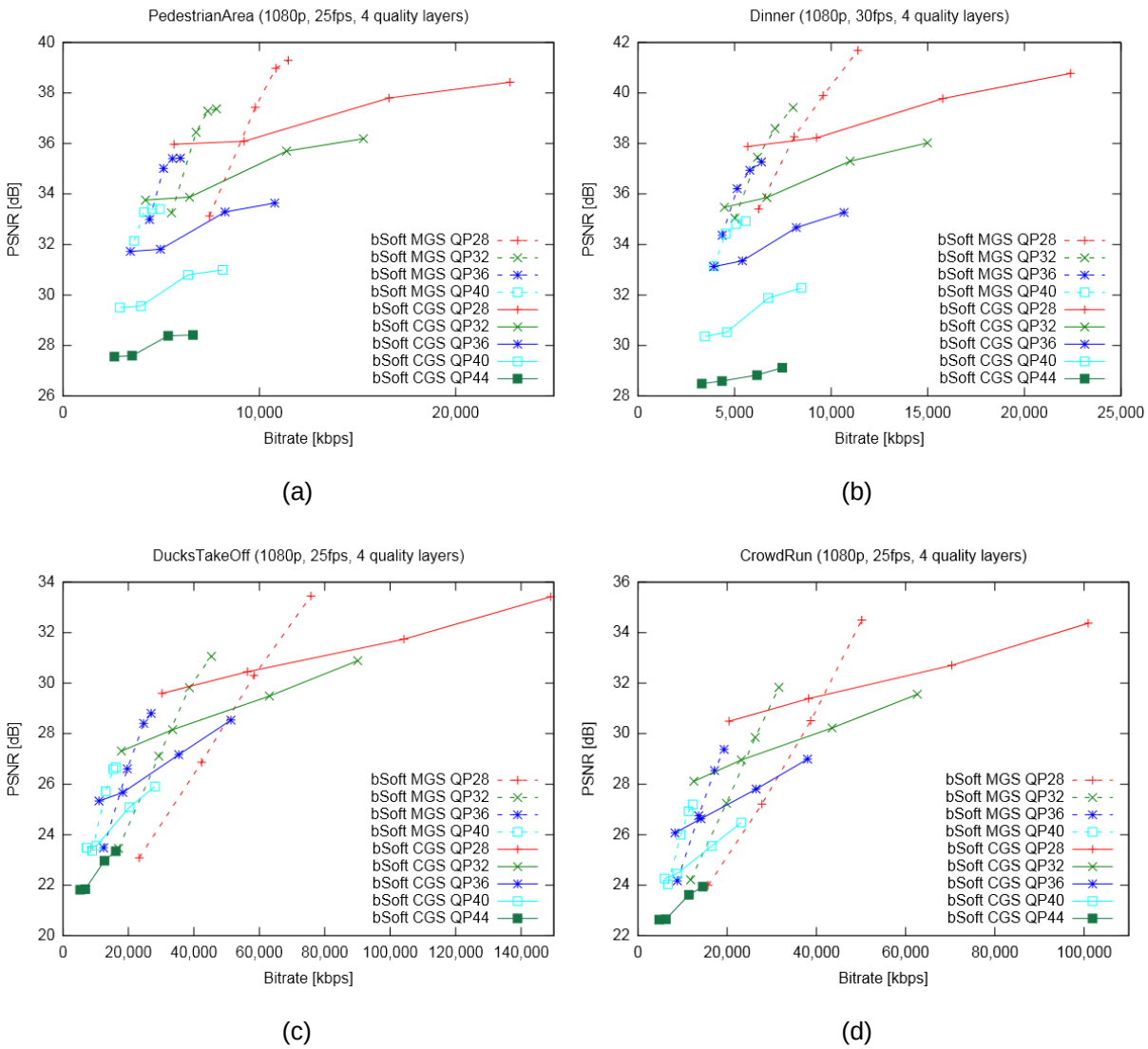


Figure 17: bSoft PSNR results for MGS vs. CGS for (a) *PedestrianArea*, (b) *Dinner*, (c) *DucksTakeOff*, and (d) *CrowdRun* sequences.

The relative bitrate penalties of additional MGS layers are provided in Table 5. The bitrate differences are measured at the highest layers; the values describe the bitrate increases for adding one MGS layer.

On average across all sequences, the JSVM encoder requires around 11.7% more bitrate for adding one MGS layer, the MainConcept encoder around 15.2% more bitrate, the VSS encoder around 19.7%, and the bSoft encoder only around 8.2% more bitrate. The overhead for the JSVM roughly confirms the findings of previous studies on lower resolutions [27][33], overheads for the MainConcept and VSS encoders are a bit higher than expected. For all encoders, the bitrate penalty for additional layers generally decreases with the number of MGS layers used as starting point. While the bSoft encoder has the lowest coding overhead for additional MGS layers, it has also the most stable one. For the JSVM, the overhead varies between 3.7% and 18.7%, depending on the content.

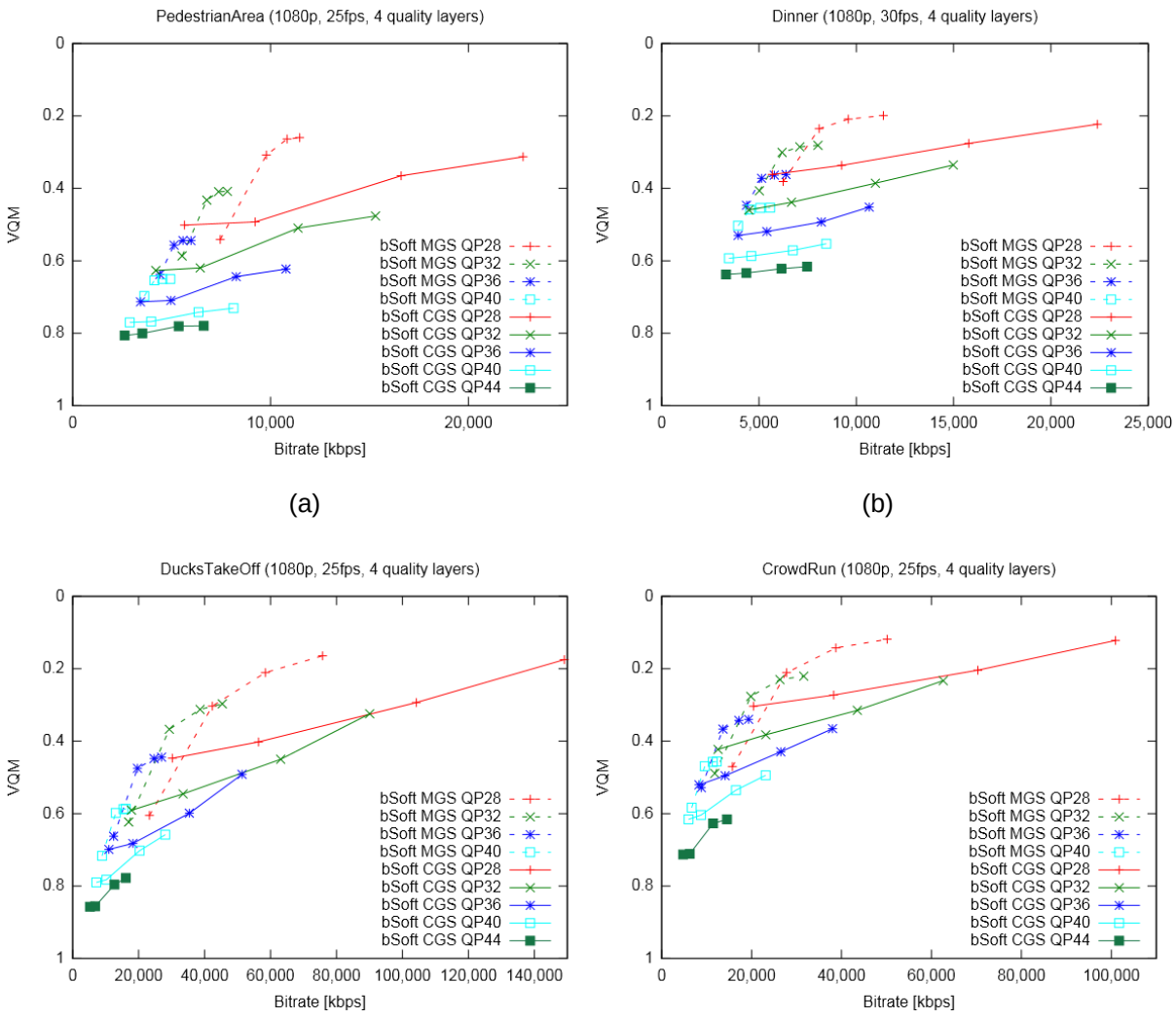
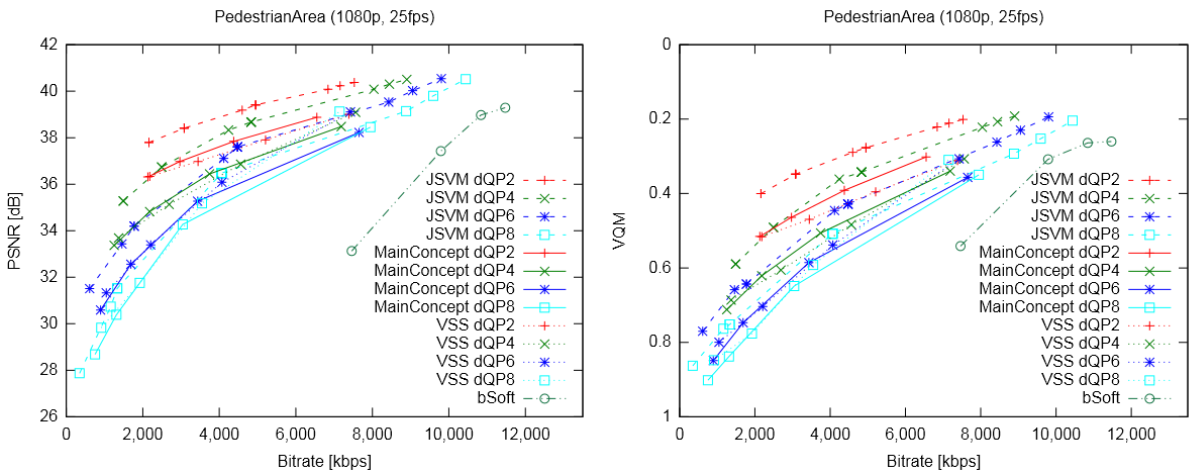


Figure 18: bSoft VQM results for MGS vs. CGS for (a) *PedestrianArea*, (b) *Dinner*, (c) *DucksTakeOff*, and (d) *CrowdRun* sequences.

3.4.4 Quality Scalability Modes

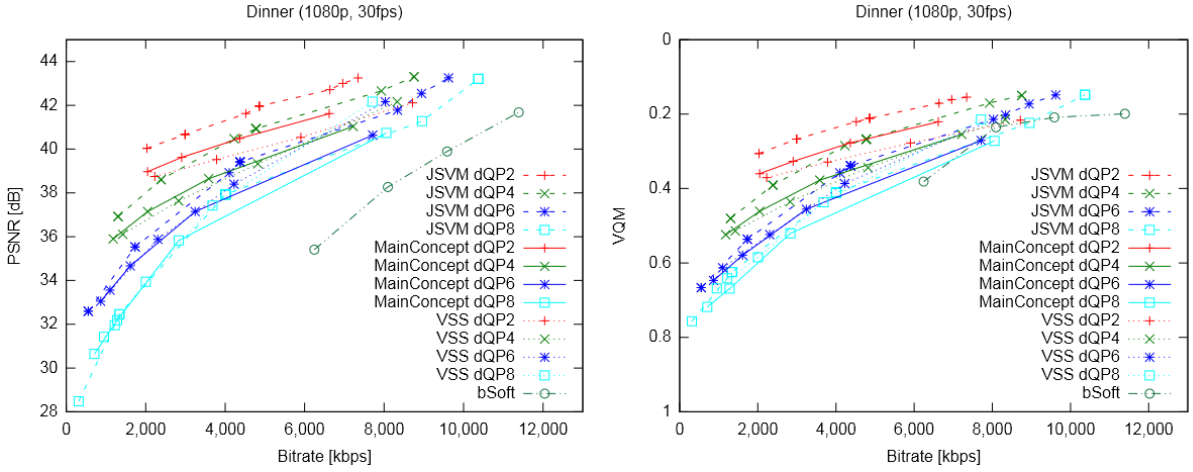
There are two approaches for quality scalability in SVC: CGS and MGS. CGS deploys the same mechanisms as spatial scalability but for a single resolution, while MGS offers a finer granularity for frame-based quality adaptation. For the bSoft encoder, this is achieved by partitioning the transform coefficients of a coded picture in order to obtain different qualities of a video. The RD results for the bSoft encoder in Section 3.4.3 have indicated quite low quality of lower SVC layers for MGS. In this section, we compare MGS and CGS performance of the bSoft encoder.

The rate-distortion performance of the bSoft encoder for MGS and CGS layers is shown in Figure 17. We compared the extraction points of SVC streams for the *CrowdRun* sequence with 4 MGS layers against 4 CGS layers with dQP of 2. Note again that all QP declarations correspond to the highest SVC layer. As we used the



(a) (b)

Figure 19: Varying dQP between MGS layers for different encoders for *PedestrianArea* sequence, (a) PSNR results and (b) VQM results.



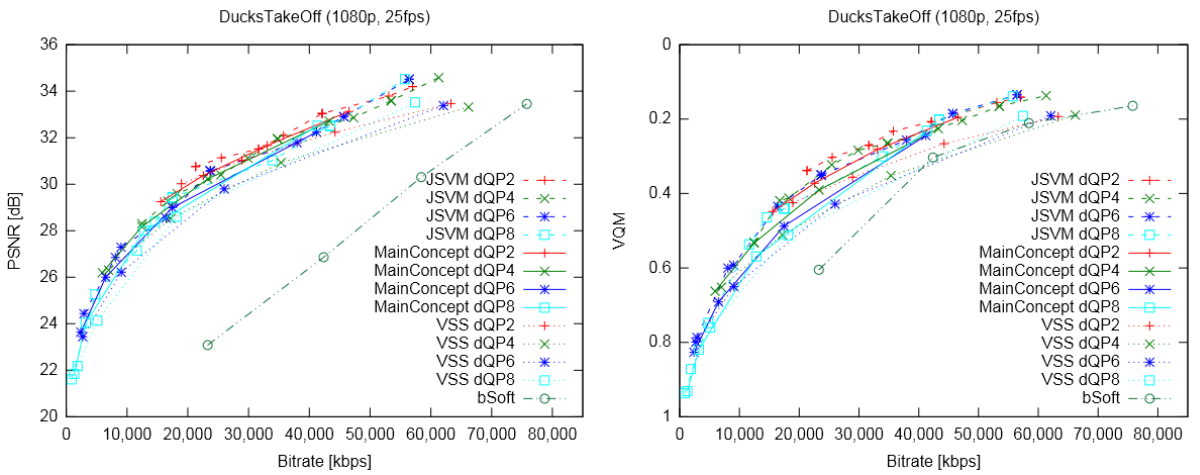
(a) (b)

Figure 20: Varying dQP between MGS layers for different encoders for *Dinner* sequence, (a) PSNR results and (b) VQM results.

same QPs for all test sequences, the *DucksTakeOff* and *CrowdRun* sequences resulted in extremely high bitrates for QP=28.

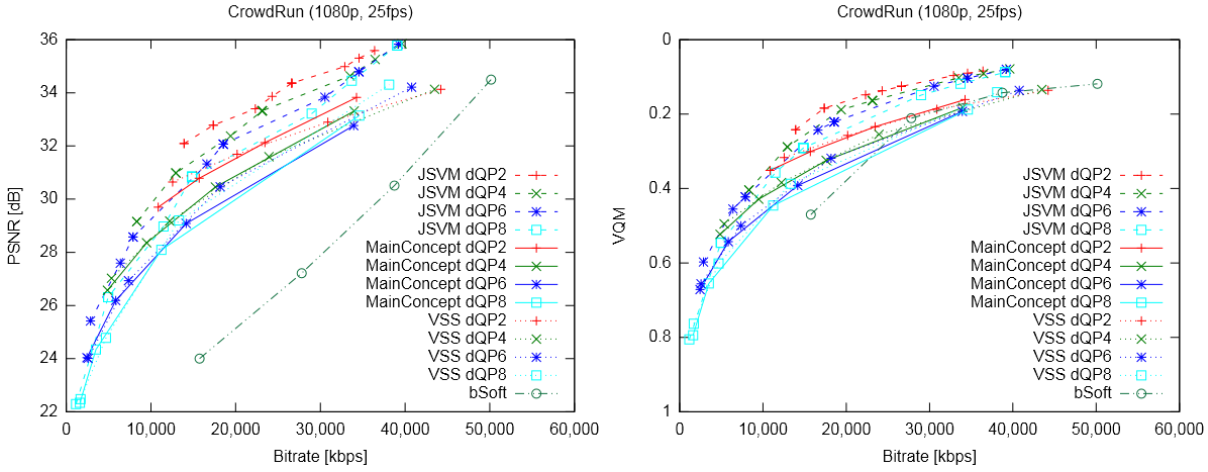
The results show that MGS has better RD performance at the highest layer, but for lower layers of the bitstream the RD performance degrades quickly. On the other hand, CGS maintains a steady RD performance, although bitrates are higher. For the lower SVC layers, RD performance of CGS is generally better than for MGS at the same bitrate.

The PSNR results of base layers in CGS mode (i.e., the left-most data points of continuous lines in the figure) follow a curve with a high slope as we would usually expect from the different quantizations. For MGS mode, on the other hand, the PSNR results of base layers are relatively constant, independent of the QP. In other words, the base quality remains the same, regardless of the quality we want to achieve at



(a) (b)

Figure 21: Varying dQP between MGS layers for different encoders for *DucksTakeOff* sequence, (a) PSNR results and (b) VQM results.



(a) (b)

Figure 22: Varying dQP between MGS layers for different encoders for *CrowdRun* sequence, (a) PSNR results and (b) VQM results [1].

the highest layer. The encoder puts a lot of information for prediction of higher layers into the base layer, but this information does not increase the quality of the base layer itself. Moreover, the RD performance of SVC layers in CGS mode is always better for lower QPs, e.g., the RD curve for the SVC layers of the bitstream labeled *bSoft CGS QP28* lies above the RD curve for *bSoft CGS QP32*.

The aforementioned behavior only appears for PSNR, VQM results for MGS mode (Figure 18) show stronger bending of the RD curves of SVC layers. Also, VQM results of the base layers depend on the QP, as normally expected. In contrast to PSNR results, the RD curves for MGS mode intersect, e.g., the first EL of the *CrowdRun* sequence at QP=28 yields better VQM performance than the highest EL at QP=32.

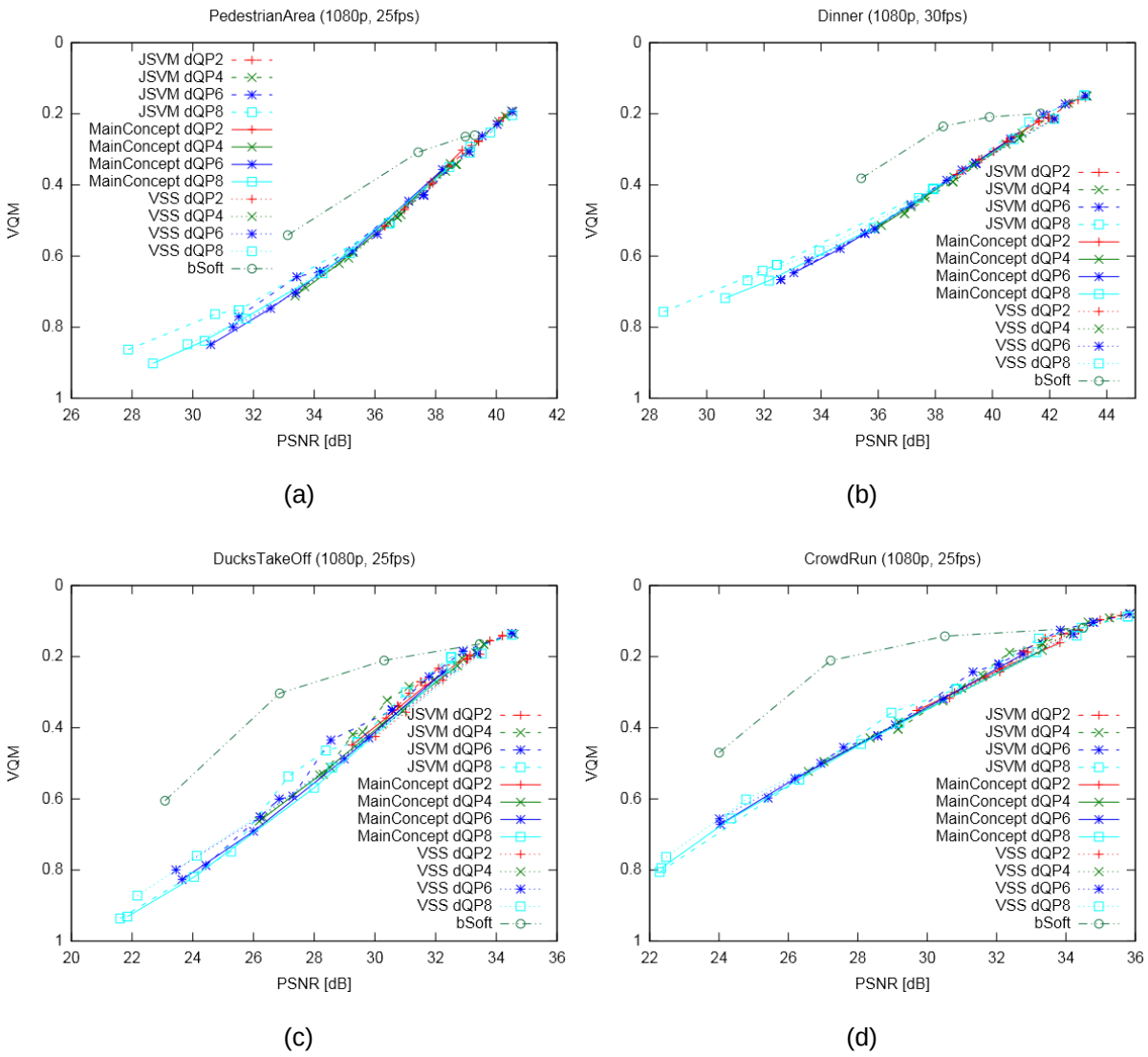


Figure 23: Correlation between PSNR and VQM for varying dQP of MGS layers for different encoders for (a) *PedestrianArea*, (b) *Dinner*, (c) *DucksTakeOff*, and (d) *CrowdRun* sequences [1].

The other encoders show far less differences between CGS and MGS, the VSS encoder yields almost identical RD performance for CGS and MGS modes. The JSVM encoder supports only up to 3 spatial layers.

3.4.5 Requantization of MGS Layers

In this test, the encoding performance of SVC encoders for a single spatial resolution (1920x1080) with four MGS layers and varying dQP between those layers was evaluated. The PSNR and VQM results for all test sequences are shown in Figure 19, Figure 20, Figure 21, and Figure 22 respectively. Note again that different line types are used for the encoders, dQPs are marked by point types (e.g., dQP=2 has the same point type and color across the encoders). As the figures are hard to read due to the high number of depicted lines, a breakdown of results of the JSVM for all dQPs and of the different encoders is given later on in this section. The bSoft

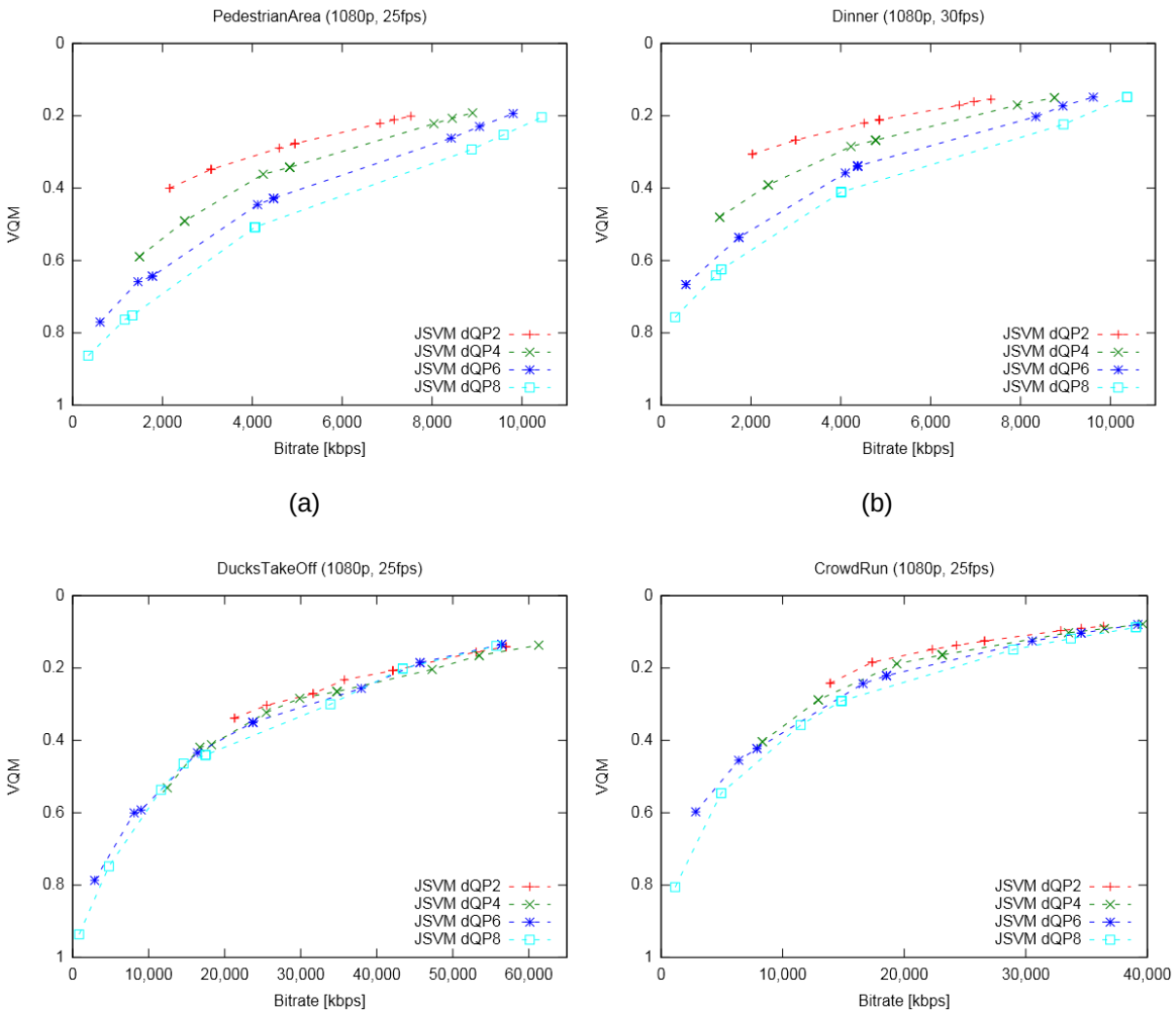


Figure 24: VQM results for varying dQP between MGS layers for JSVM encoder for (a) *PedestrianArea*, (b) *Dinner*, (b) *DucksTakeOff*, and (d) *CrowdRun* sequences.

encoder distributes transform coefficients automatically across layers, eliminating the need for different dQP encodings in this test. Also note that JSVM bitstreams allow for further extraction points in addition to the MGS layers.

It can be observed that a dQP of 2 is sufficient for serving a decent range of bitrates at 4 layers, while having the best RD performance. Obviously, higher dQP values (e.g., 6 or 8) cause such a strong quantization of the base layer that VQM results drop even below poor quality, in particular for sequences with low TI, such as *PedestrianArea* in Figure 19 (b) and *DucksTakeOff* in Figure 21 (b).

The selection of an appropriate bitrate for the base layer is an important aspect for SVC-based adaptive media streaming. A low bitrate reduces the risk of stalling if the bandwidth is insufficient. On the other hand, our results show how quickly the quality of the base layer degrades at lower bitrates (i.e., higher dQPs). The low quality of the base layer also influences the quality of all enhancement layers (cf. Figure 24 later

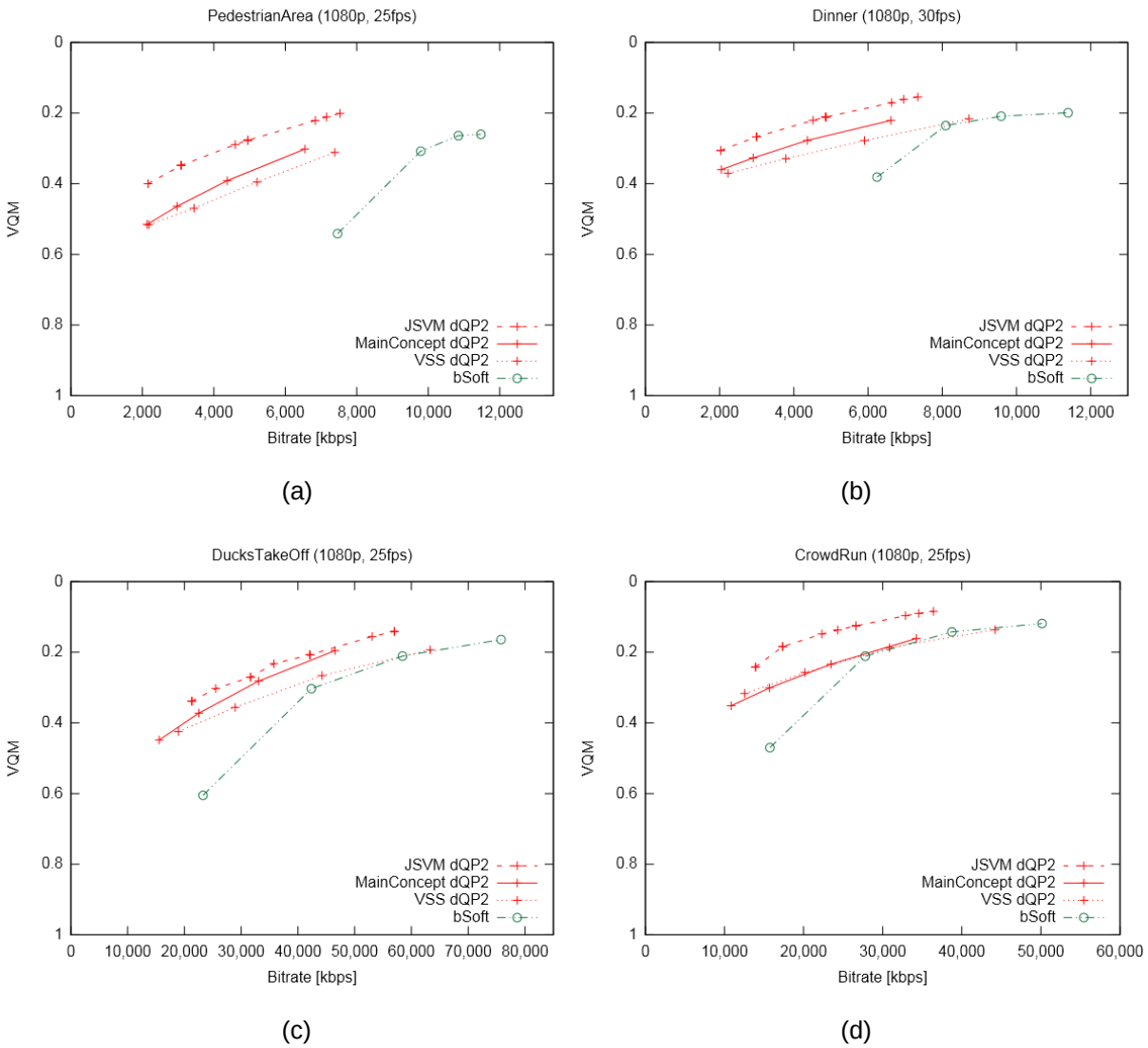


Figure 25: VQM results for dQP=2 between MGS layers for different encoders for (a) *PedestrianArea*, (b) *Dinner*, (b) *DucksTakeOff*, and (d) *CrowdRun* sequences.

on), as the encoder struggles to predict high quality signals from the low quality base layer.

As mentioned, the bSoft encoder automatically distributes transform coefficients to create MGS layers. The base layer achieves quite low PSNR results and the bitstream has significantly higher bitrates than other encoders for the same PSNR.

We observe that the bSoft encoder, while having somewhat lower PSNR results, has good VQM results (especially for sequences with high SI), which are on par with the other encoders in terms of rate-distortion. Due to VQM's better correlation with the human visual system, these results suggest that the actual visual quality of the bSoft encoder is higher than indicated by PSNR values.

To further analyze this behavior, we compared PSNR vs. VQM for all four sequences in Figure 23. The plots clearly show that especially the lower layers of the bSoft encoder yield better VQM results than other encoders at the same PSNR. For other encoders, the plots show a strong correlation between PSNR and VQM for the

respective sequences, with some small exceptions towards lower layers. Note however, that this correlation is content-dependent.

In order to investigate the impact of varying dQP on coding performance more closely, Figure 24 shows results for the JSVM encoder based on Figure 19 (b) for *PedestrianArea*, Figure 20 (b) for *Dinner*, Figure 21 (b) for *DucksTakeOff*, and Figure 22 (b) for *CrowdRun* respectively. Note that the x-axes are adjusted to the selected results. As noted before, it can be observed that dQP=2 is typically sufficient to cover adequate bitrate ranges. Furthermore, lower dQP values result in better RD performance, although the effect decreases for more complex scenes (with higher SI and/or TI). The plots also clearly show further extraction points in addition to the MGS layers. Those are achieved by discarding some MGS layer NALUs from the bitstream. As expected, the RD performance is better for extracting exactly an MGS layer than it is for any additional extraction points.

Next, we take a closer look at those very results, investigating different encoders for dQP=2. In Figure 25 we filtered the results for dQP=2 from the VQM results of Figure 19 (b) for *PedestrianArea*, Figure 20 (b) for *Dinner*, Figure 21 (b) for *DucksTakeOff*, and Figure 22 (b) for *CrowdRun* respectively. Although the RD performances of SVC layers slightly vary across encoders, they generally have some extraction points that offer comparable RD performance across encoders. As expected, the JSVM exhibits the best RD performance across all test sequences. As a trend, the RD curve of the VSS encoder remains somewhat parallel to that of the JSVM, i.e., the VSS encoder distributes bitrates and quality across MGS layers in a similar way to the JSVM, but at lower RD performance. On the other hand, the MainConcept encoder starts at low RD performance at the base layer but the RD curve gets closer to the JSVM for

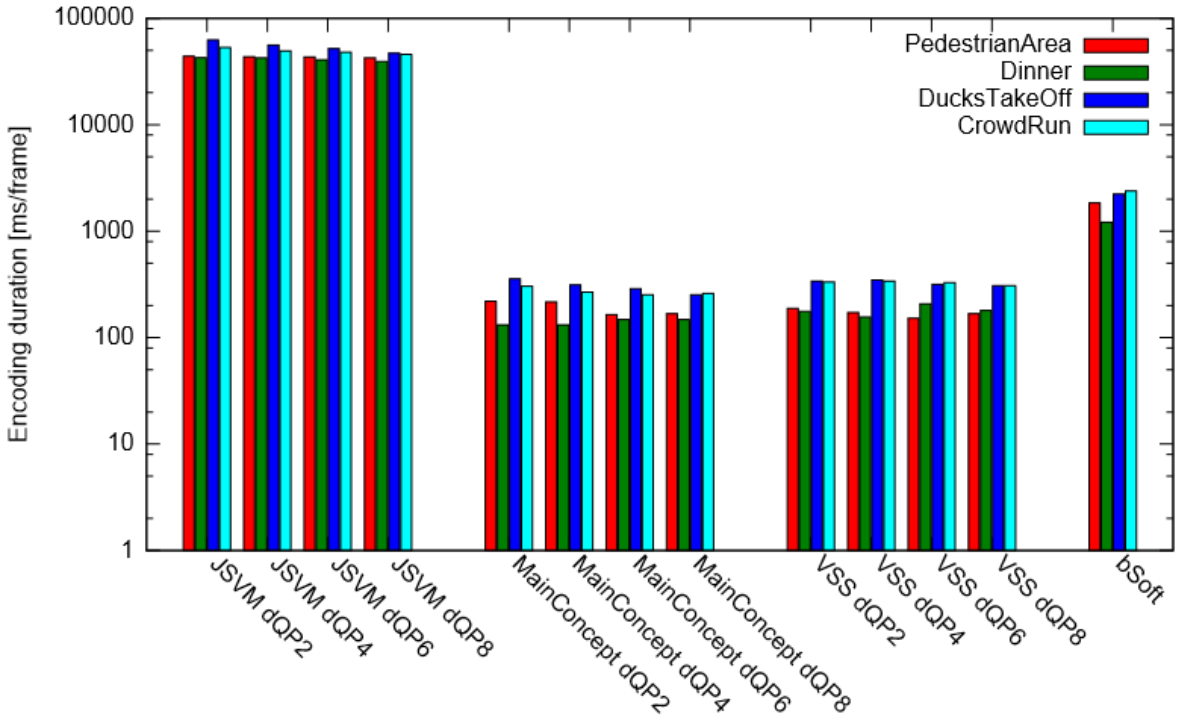


Figure 26: Encoding durations for varying dQP between MGS layers for different encoders.

higher layers. Note again, that the bSoft encoder automatically distributes transform coefficients to form MGS layers. Thus, it does not fully fit into an evaluation of dQP=2. Nevertheless, the results are included in Figure 25 and it can be observed that the bSoft encoder performs better for more complex sequences (with higher SI and/or TI).

3.4.6 Encoding Durations

Based on the previous test, we analyze encoding durations for the different encoders.

Encoding durations per frame are shown in Figure 26 and corresponding average durations across dQPs are provided in Table 6. Note that results are depicted on a logarithmic scale. The fastest encoders are MainConcept and VSS, which are two orders of magnitude faster than the JSVM. The bSoft encoder is still one order of magnitude faster than the JSVM. For all encoders, encoding speeds are slightly slower at lower dQP values. Also, the *Dinner* sequence yields shortest encoding durations across all encoders (probably because it is a synthetic scene), followed by *PedestrianArea*.

As industrial SVC encoders are optimized for encoding speed, they typically sacrifice some RD performance, e.g., by using fast block search algorithms for motion estimation. On the other hand, the JSVM accepts high computational complexity throughout the video coding tool chain to ensure high RD performance.

Based on our findings, we conclude that, out of the industrial encoders, the MainConcept encoder is better suited for good RD performance at the highest layer, while the VSS encoder yields a more stable RD performance across layers. For the bSoft encoder, the bitrate should be considered; on the other hand, we noted that the encoder is better suited for more complex sequences and that the VQM results for the encoder indicate higher RD performance than the PSNR results do.

Table 6: Average encoding durations of different encoders.

Sequence	JSVM [ms/frame]	MainConcept [ms/frame]	VSS [ms/frame]	bSoft [ms/frame]
PedestrianArea	43304	192	170	1840
Dinner	41200	140	180	1212
DucksTakeOff	54376	302	328	2228
CrowdRun	49079	271	327	2384
Average	**46990**	**226**	**251**	**1916**

3.5 Hybrid SVC-DASH with High-Definition Content

DASH enables the client to select and adjust characteristics of a stream (e.g., resolution and bitrate) on the fly while benefitting from existing HTTP infrastructures. While DASH is traditionally used with single-layer coding formats such as AVC, the usage of SVC can offer further advantages in terms of adaptation capabilities and optimization of resource utilization [59].

The successful deployment of SVC in DASH strongly depends on proper and educated encoding configurations to facilitate adaptive streaming. In this section we propose a hybrid SVC framework for DASH and HD content, comprising encoding guidelines and quality evaluations for various scalability options with a special focus on multiple resolutions. In particular, we suggest using multiple independent SVC streams, each providing a given resolution corresponding to a certain device class and allowing for SNR scalability. In this section, we give an overview of DASH and its deployment with SVC, further validate our coding recommendations, and investigate scalability options providing quality evaluations for major encoders.

This section focuses on SVC encoding for DASH by testing configurations for typical DASH deployments, i.e., high number of enhancement layers and support of multiple spatial resolutions. Nevertheless, the results also apply to other SVC-based media streaming scenarios such as RTP streaming or P2P streaming. While the validation of bitrate recommendations for 2 bitrates and the combination of spatial scalability and MGS have been covered in Sections 3.4.1 and 3.4.2 respectively, we argue that evaluations for 4 bitrates are more relevant to DASH deployments.

3.5.1 Deployment of SVC in DASH

With DASH, a server may offer multiple representations of the same content where each representation is typically characterized by – but not limited to – a specific resolution and bitrate. Those representations are described in an XML-based manifest file, called MPD, which the client retrieves before starting the streaming session. The client picks the representation that is best suited for its current context (e.g., display resolution and available bandwidth). Each representation is split into temporal segments (e.g., 2-10 sec. each). The client can adapt to fluctuating network conditions by switching to lower bitrate representations at segment boundaries. Traditionally, representations are encoded as separate/independent (AVC) bitstreams. The deployment of SVC can bring some advantages in terms of storage (alleviating the need for multiple bitstreams of the same content to be stored at the server), cache performance [59], and adaptation [61]. With AVC, if the download of a segment cannot be completed before playout time, e.g., due to a sudden bandwidth decrease, the client has to decide whether to continue the download and risk stalling or to discard the current segment and switch to a lower representation. Note that the downloaded bits of the discarded segment are wasted.

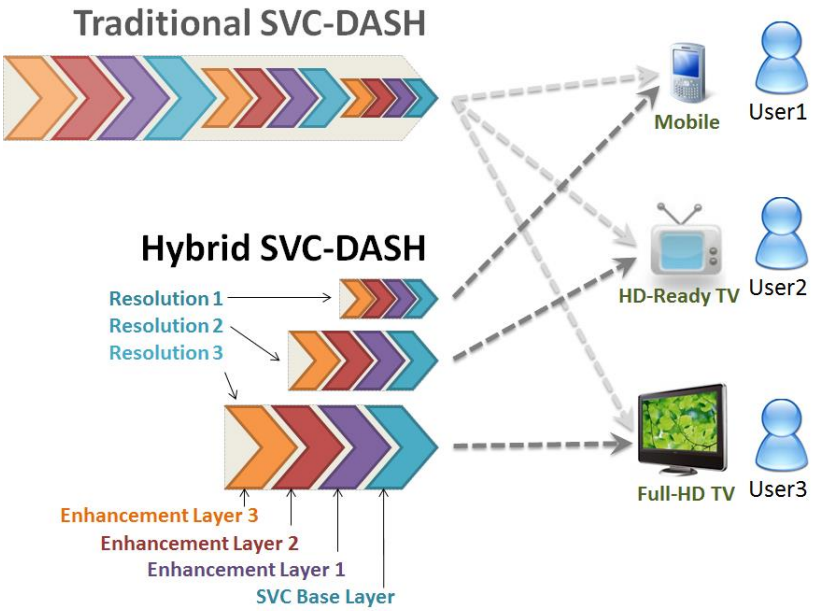


Figure 27: Hybrid SVC-DASH.

As an optimization for stream-switching, double decoding can be applied, in which the client uses as many frames as possible from the unfinished higher representation before displaying the lower representation. With this approach, less downloaded bits are wasted. Nevertheless, the client must download the entire segment of the lower representation to ensure proper decoding.

SVC can be deployed in DASH as follows. Each representation contains an SVC layer and describes the dependencies between layers as shown in Listing 1. The `dependencyId` attribute indicates which other representations (i.e., lower SVC

```
<AdaptationSet>
  <Representation id="0" width="960" height="540"
bandwidth="1200000">
    <SegmentList> <SegmentURL media="540p-BL-seg1.264"/>
    </SegmentList>
  </Representation>
  <Representation id="1" dependencyId="0" width="960"
height="540" bandwidth="1975000">
    <SegmentList> <SegmentURL media="540p-EL1-seg1.264"/>
    </SegmentList>
  </Representation>   <!-- Further representations... -->
  <Representation id="4" dependencyId="0 1 2 3" width="1920"
height="1080" bandwidth="4000000">
    <SegmentList> <SegmentURL media="1080p-EL4-seg1.264"/>
    </SegmentList>
  </Representation>  <!-- Further representations... -->
</AdaptationSet>
```

Listing 1: Simplified MPD for SVC streaming of multiple resolutions with a single bitstream featuring spatial scalability [2].

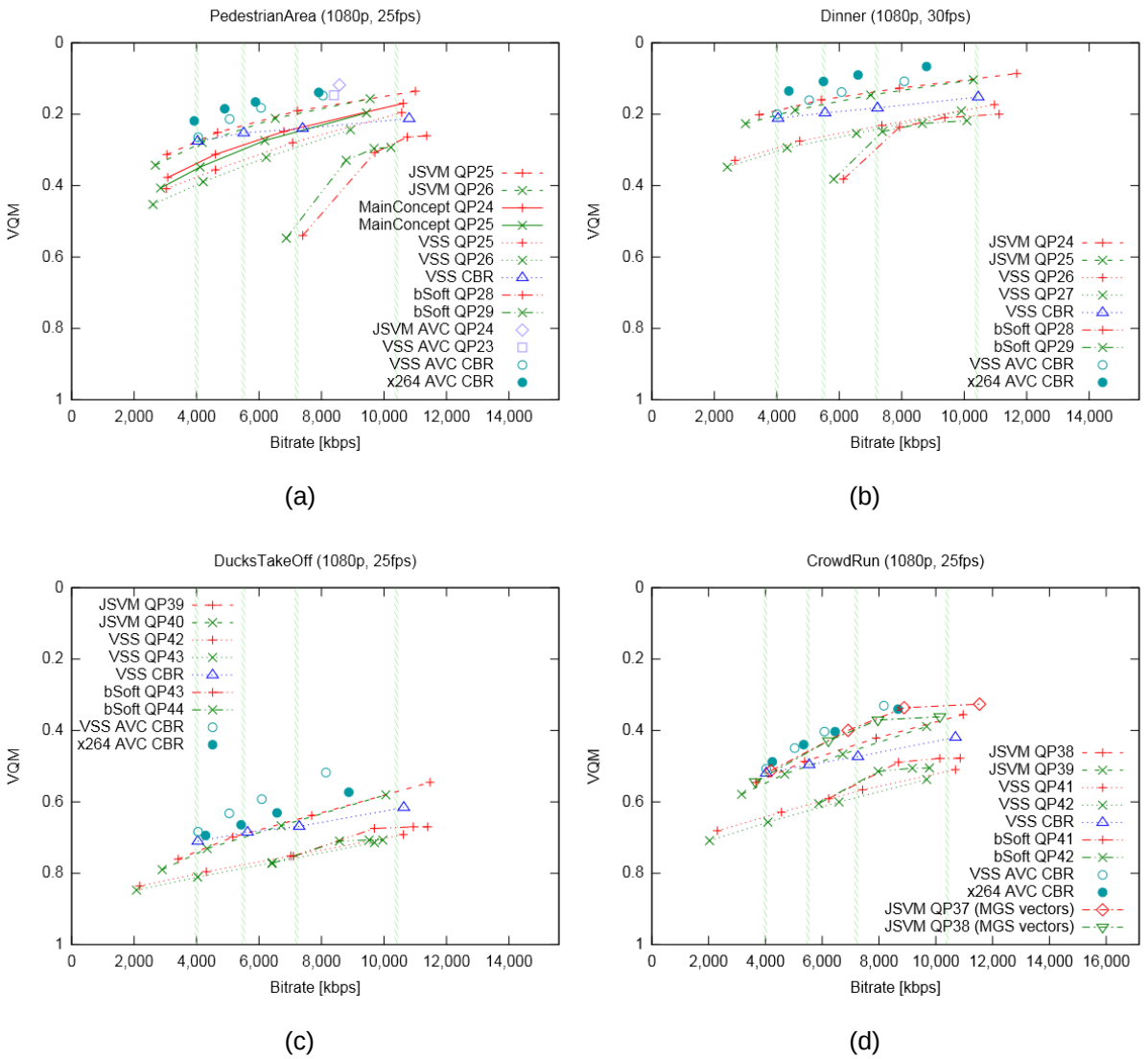


Figure 28: VQM results of AVC and SVC with 4 bitrates for (a) *PedestrianArea*, (b) *Dinner*, (c) *DucksTakeOff*, and (d) *CrowdRun* sequences [2].

layers) are required for decoding a given representation. As long as the client has downloaded the SVC base layer, it can decode at least a basic representation of the content, thus avoiding the risk of stalling. Each additional enhancement layer increases the video quality.

We argue that well-chosen SVC configurations are an important aspect towards a successful deployment of SVC. Throughout this section, we discuss and evaluate several deployment options for SVC in DASH. One option is to use a single SVC bitstream to comprise all representations. The advantage of such a configuration is that the redundancy of having multiple similar bitstreams for a single content is removed. Furthermore, caching performance can be increased as all clients use the same SVC base layer. The downside of this approach is that the coding overhead increases with the number of SVC layers, specifically when covering a high range of resolutions. If the coding overhead becomes too high, it will outweigh the advantages of SVC.

Our proposal is to encode the content into multiple independent SVC bitstreams, one per resolution (e.g., representing certain device classes), and only relying on SVC quality scalability. The approach is referred to as *hybrid SVC-DASH* and the idea behind this approach is to confine the coding overhead by avoiding spatial scalability while benefitting from SVC's advantages. The approach is illustrated in Figure 27. Provided a sufficient bitrate range for each bitstream, a client will try to maintain one resolution during the entire streaming session as resolution switches are more disturbing for the viewer than mere bitrate changes [127][128]. While the lower resolution can be upsampled to be displayed with the same size as the higher resolution, upsampling causes undesirable blurring artifacts.

SVC offers two modes for quality scalability, CGS and MGS. In order to obtain a higher number of SVC quality layers for covering a higher range of bitrates, these two modes could be combined. However, the issue arises that not all of these layers are actually useful for a client as we will discuss later. In the following sections, we establish and validate encoding recommendations for SVC streaming.

3.5.2 SVC Encoding Performance

Based on the bitrate suggestions deduced in Section 3.3.1 and Table 3 in particular, Table 7 provides selected bitrate recommendations for AVC and SVC streaming with 4 bitrates at resolutions from 1920x1080 (1080p) down to 640x360. The bitrate values are the same as in Table 2 (for AVC) and Table 3 (for SVC); however, we focus exclusively on 4 bitrates for DASH, providing a more concise set of recommendations. Quality evaluations based on these recommendations are given in the following subsection.

In this section, we validate the devised SVC coding recommendations for various encoders and provide RD performance evaluations for several scalability options.

In addition to the SVC encoders JSVM, MainConcept, VSS, and bSoft, we also test the AVC encoder *x264* [129]. Again, we use the four test sequences *PedestrianArea*, *Dinner*, *DucksTakeOff*, and *CrowdRun*, using the first 250 frames of each sequence.

As indicated in Section 3.3.2, the JSVM encoder supports MGS layers by using requantization as well as by splitting transform coefficients (i.e., MGS vectors). While the evaluations of the JSVM in the previous section were based only on requantization, we also include results for distribution of transform coefficients. Since the MainConcept and VSS encoders rely on requantization, our focus remains on requantization for the JSVM encoder as well.

3.5.2.1 Encoder Comparison and Bitrate Validation for 4 Quality Layers

We first compare the RD performance of the x264 encoder to SVC encoders in order to establish a base line for our further tests. For SVC we use a single-layer

Table 7: Selected bitrate recommendations for SVC streaming [2].

Resolution	Bitrate suggestions (4 bitrates) [kbps]	
	AVC streaming	SVC streaming
1920x1080	8000, 6000, 5000, 4000	10400, 7200, 5500, 4000
1280x720	6000, 4000, 2500, 1500	7800, 4800, 2750, 1500
960x540	2700, 2250, 1800, 1200	3500, 2700, 1975, 1200
640x360	1600, 1250, 900, 600	2075, 1500, 990, 600

configuration (i.e., an AVC-compatible base layer) and a configuration with 4 MGS layers. Single-layer (AVC) bitstreams are encoded in CBR mode with target bitrates suggested for AVC streaming. SVC bitstreams with 4 MGS layers are encoded in fixed QP rate control mode for all SVC encoders. The re-quantization between MGS layers was set to a deltaQP of 2 (except for the bSoft encoder, which does not need any requantization as explained in Section 3.3.1). Additionally, the sequences were encoded with the VSS encoder in CBR mode as it was the only one of the tested SVC encoders to provide decent CBR support at all tested resolutions.

We also tested the JSVM encoder using MGS vectors with a partitioning into three MGS slices containing 1, 2, and 13 transform coefficients. The partitioning was found through empirical testing to best match the recommended bitrates. The deltaQP between the base layer and the enhancement layer was set to 6, which amounts to the same as the 3 requantized enhancement layers with a deltaQP=2.

The VQM results for the tested sequences at 1080p resolution are shown in Figure 28. Note again that the y-axis of VQM results is an impairment scale from 1 (high distortion) to 0 (no distortion), indicating the expected quality of a video. For fixed QP mode, bitstreams with bitrates just below and just above the bitrate suggestions for the highest SVC layer (cf. Table 7) are shown. The results for the MainConcept encoder and for AVC configurations in fixed QP mode are only shown for *PedestrianArea* for the sake of readability. The RD performance of the MainConcept encoder in relation to JSVM and VSS for the *PedestrianArea* sequence is representative for the other sequences. Results for the JSVM encoder in MGS vector mode are only shown for the *CrowdRun* sequence for the same reason.

As expected, AVC yields a higher RD performance than SVC with multiple MGS layers. However, at the lowest bitrate, the SVC bitstream from the VSS encoder in CBR mode (labeled *VSS CBR*) has only marginal overhead compared to the corresponding AVC bitstream from the same encoder (labeled *VSS AVC CBR*). Whether the x264 encoder outperforms VSS depends on the content.

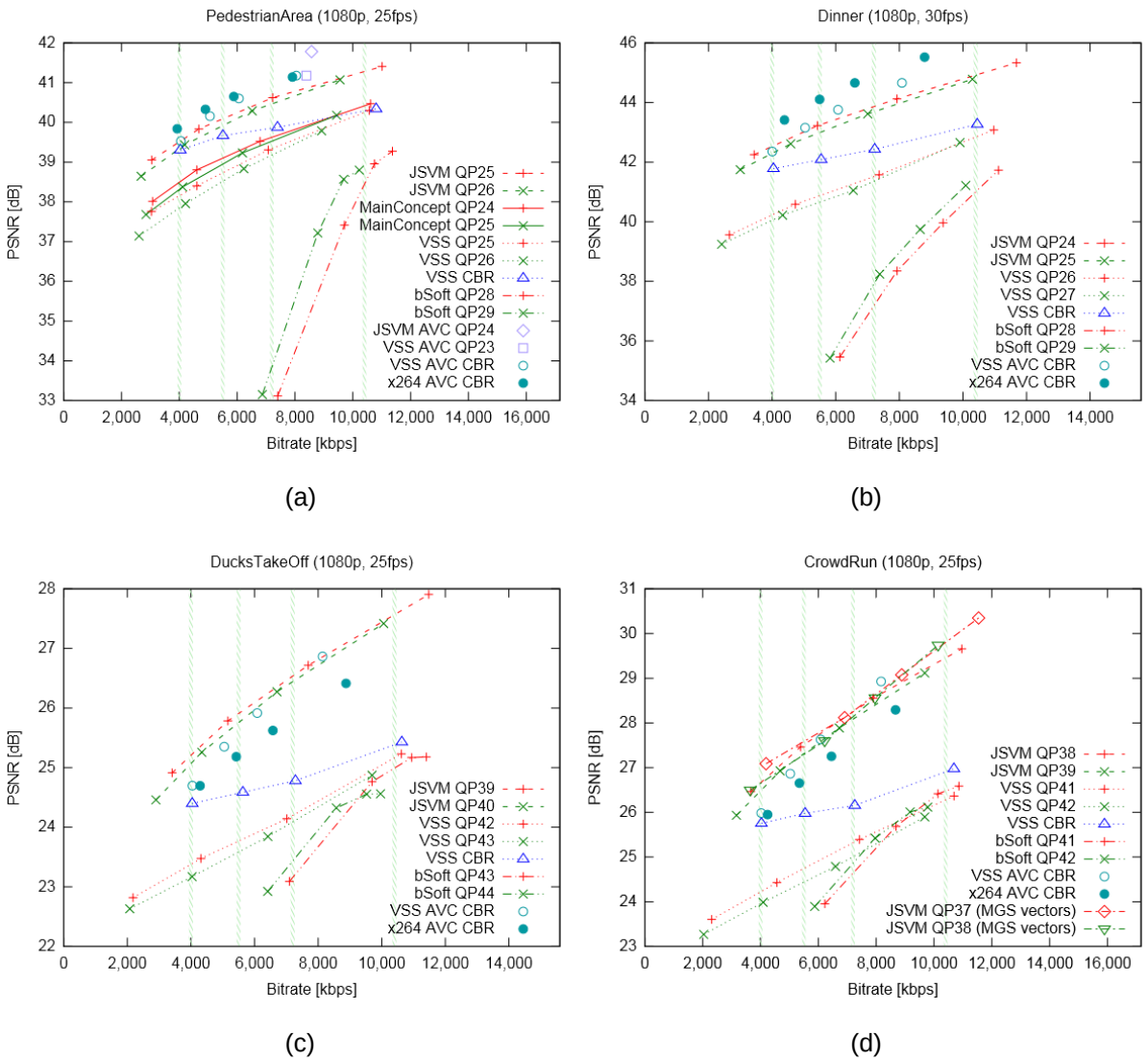
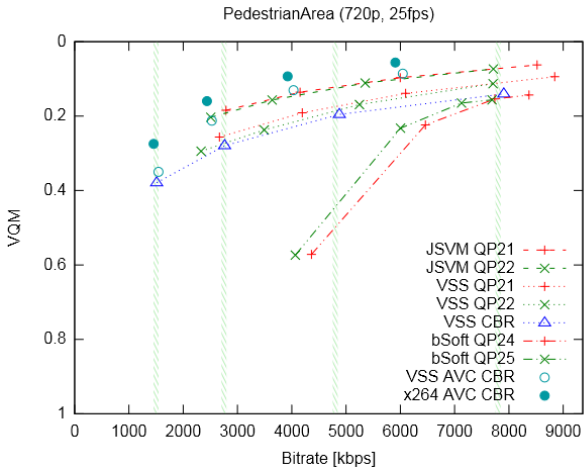


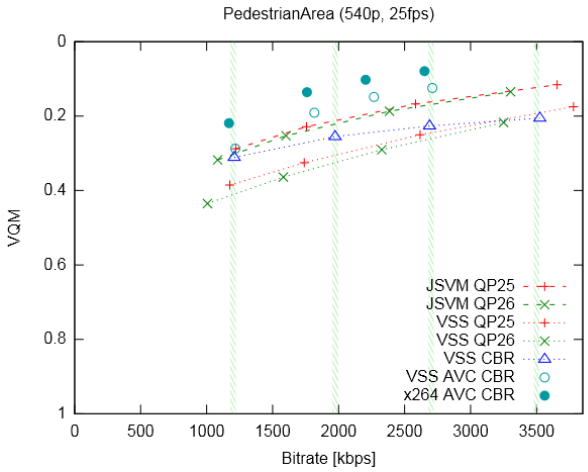
Figure 29: PSNR results of AVC and SVC encoders with 4 bitrates for (a) *PedestrianArea*, (b) *Dinner*, (c) *DucksTakeOff*, and (d) *CrowdRun* sequences [2].

The JSVM encoder with 4 MGS layers tends to reach the quality of the AVC encoders in several cases if we consider the expected SVC coding overhead discussed in Section 3.3.1. That is, qualities of the individual layers of the JSVM encoder are roughly on the same level as the corresponding AVC encoders. For example, in Figure 28 (a) the JSVM with 4 layers at QP=25 (labeled *JSVM QP25*) has a VQM result of 0.136 while the VQM score for the x264 encoder at 8,000 kbps (labeled *x264 AVC CBR*) is 0.139 and the VQM score of the JSVM in AVC mode at QP=24 (labeled *JSVM AVC QP24*) is 0.118. However, there are significant discrepancies in RD performance between AVC encoders, which constrict conclusive comparisons.

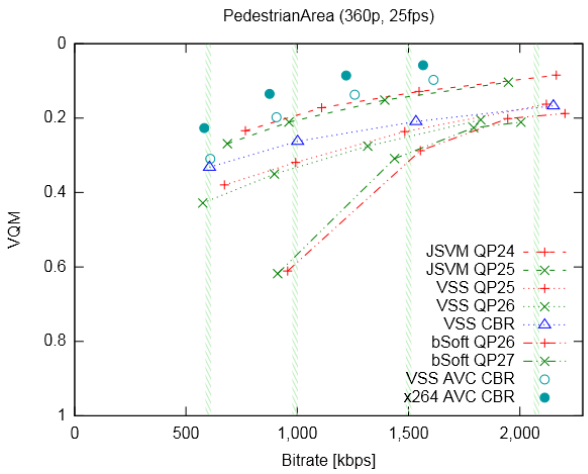
Among SVC encoders, the JSVM yields the best RD performance, followed by MainConcept, VSS and bSoft. The bSoft encoder allocates only poor quality to the base layer. However, the bSoft encoder outperforms VSS for more complex sequences such as *CrowdRun*. Sequences with high SI exhibit poor overall encoding performance for all encoders, as observed earlier in Section 3.4.1.



(a)



(b)



(c)

Figure 30: VQM results of AVC and SVC encoders with 4 bitrates at (a) 1280x720, (b) 960x540, and (c) 640x360 resolutions [2].

For the JSVM encoder, the MGS vector mode yields slightly higher RD performance than requantization. Starting from virtually the same base layer quality as the

Table 8: Storage requirements for SVC streaming per resolution.

Resolution	AVC bitstreams	SVC bitstream	Reduction
1920x1080	23,000 kbps	10,400 kbps	54.8%
1280x720	14,000 kbps	7,800 kbps	44.3%
960x540	7,950 kbps	3,500 kbps	55.8%
640x360	4,350 kbps	2,080 kbps	52.2%

requantization mode, the first two MGS slices have higher RD performance than requantization mode. From the second MGS slice to the full MGS enhancement layer, VQM results almost stagnate. It appears that the last and biggest MGS slice (13 of 16 coefficients) contributes little to the video quality. We find this behavior for other sequences as well.

We note that the VSS encoder in CBR mode yields a surprisingly high quality at the base layer, but enhancement layers only bring little quality increases. On the one hand, such a low slope in RD performance makes bitrate switches in media streaming less perceivable. On the other hand, a higher bitrate that does not yield higher quality is basically a waste of bandwidth.

For comparison, Figure 29 shows the PSNR results for all sequences. It can be observed that several encoders yield better VQM performance than the PSNR results indicate (in particular for the *CrowdRun* sequence in Figure 29 (d)). For example, the PSNR results for AVC bitstreams are below the RD performance of the JSVM, while the VQM results show the opposite. We find this behavior for encoding in CBR mode for several sequences and for the bSoft encoder at lower layers for all sequences. In contrast to the corresponding VQM results, the JSVM encoder with MGS vector mode only outperforms requantization mode at the full MGS enhancement layer in terms of RD performance.

Figure 30 shows the VQM results at lower resolutions for the *PedestrianArea* sequence. Again, results for the MainConcept encoder are not shown for the sake of readability. Due to an encoder error, we were not able to obtain results for the bSoft encoder at resolution 960x540.

As with 1080p, the JSVM encoder with 4 MGS layers tends to reach the quality of the AVC encoders in most cases considering the expected SVC coding overhead.

In terms of storage requirements, SVC is more efficient than AVC with multiple representations for 4 MGS layers (if accepting small quality reductions in some cases) as shown in Table 8. Of course, the storage reduction comes at the cost of the discussed SVC coding overhead for every streaming session. The bitrate recommendations from Table 7 for 4 MGS layers yield consistent qualities for all resolutions. The applied re-quantization with a dQP of 2 for fixed QP mode correlates with the bitrate suggestions of lower layers to a reasonable extent for JSVM, MainConcept, and VSS encoders as further discussed in [1].

```
<AdaptationSet>
  <Representation id="0" width="960" height="540"
bandwidth="1200000">
    <SegmentList> <SegmentURL media="540p-BL-seg1.264"/>
    </SegmentList>
  </Representation>
  <Representation id="1" dependencyId="0" width="960"
height="540" bandwidth="1975000">
    <SegmentList> <SegmentURL media="540p-EL1-seg1.264"/>
    </SegmentList>
  </Representation>  <!-- Further representations with
enhancement layers at 960x540... -->
<Representation id="4" width="1920" height="1080"
bandwidth="4000000">
    <SegmentList> <SegmentURL media="1080p-BL-seg1.264"/>
    </SegmentList>
  </Representation>
  <Representation id="5" dependencyId="4" width="1920"
height="1080" bandwidth="5000000">
    <SegmentList> <SegmentURL media="1080p-EL1-seg1.264"/>
    </SegmentList>
  </Representation>  <!-- Further representations with
enhancement layers at 1920x1080... -->
</AdaptationSet>
```

Listing 2: Simplified MPD for SVC streaming of multiple resolutions with one bitstream per resolution [2].

3.5.2.2 Combination of Spatial Scalability and MGS

SVC streaming of multiple resolutions can be achieved by either encoding one SVC bitstream that features spatial scalability or encoding several bitstreams, one per resolution. Example MPDs for the two approaches are depicted in Listing 1 and Listing 2 respectively. In Listing 2, two individual SVC bitstreams are described, one with spatial resolution 960x540, the other with 1920x1080. Each bitstream has multiple quality enhancement layers, described as representations depending on the base layer.

In this section we evaluate the RD performance of SVC bitstreams with both spatial and quality scalability compared to hybrid SVC-DASH with just quality scalability.

In SVC, each layer is identified by its dependency (i.e., resolution), quality, and temporal id, commonly denoted *DQT*. We consider two different extraction paths for achieving spatial scalability. A quality layer *q* of an upper resolution *d*, e.g., DQT=(*d*,*q*,0), can either depend on the same quality layer of the previous resolution, i.e., DQT=(*d*-1,*q*,0), or on the highest layer *Q* of the previous resolution, i.e., DQT=(*d*-1,*Q*,0). The first extraction path (subsequently denoted *partial extraction path*), which is implemented in the reference software, yields a lower bitrate at the expense of discarded enhancement information from the lower resolution. The VSS

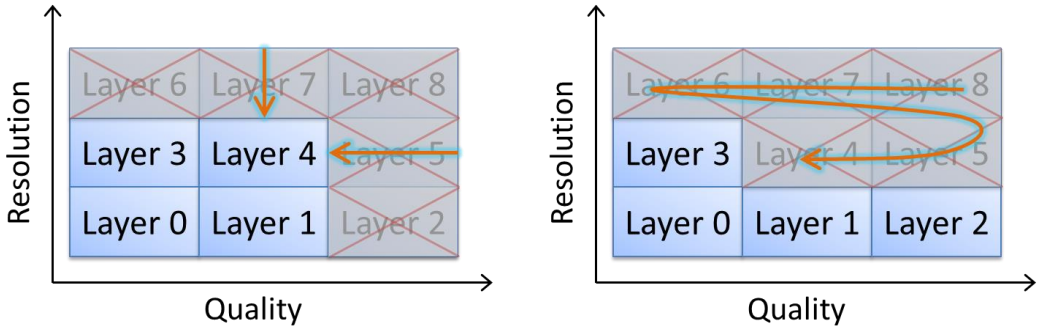


Figure 31: Adaptation for (a) partial extraction path and (b) full extraction path.

encoder also supports the second extraction path (subsequently denoted *full extraction path*). Figure 31 illustrates how adaptation is performed for both extraction paths. With the JSVM tool set, adaptation is performed by specifying the DID for the spatial layer and the QID for the MGS layer, based on which the SVC layers are extracted from the bitstream. Conversely, the VSS decoder extracts SVC layers based on the absolute layer number as depicted in Figure 31 (b).

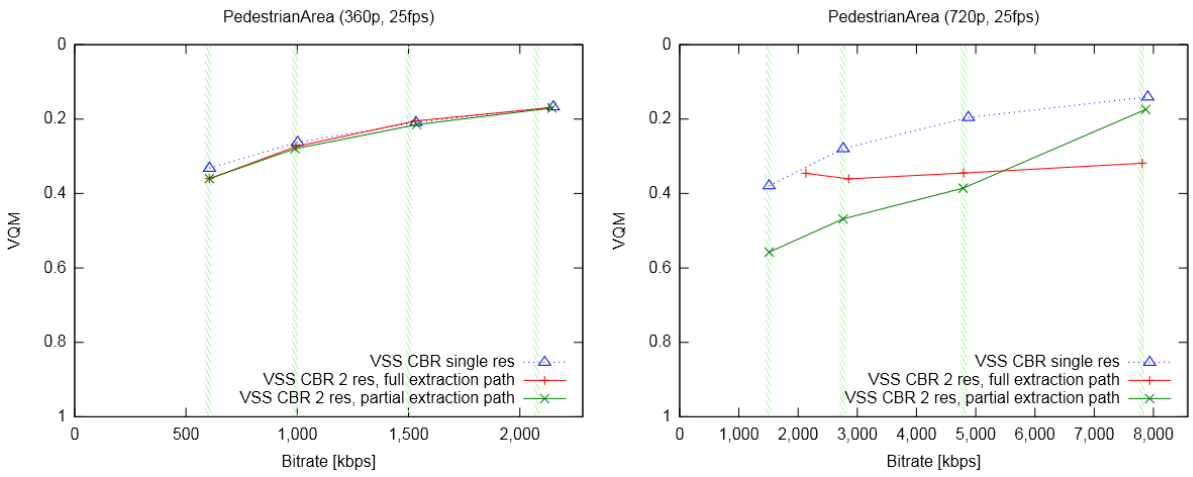
Figure 32 shows VQM results for both extraction paths at resolutions 640x360 and 1280x720 for the VSS encoder. Note that the bitstreams range over both resolutions. Single resolution SVC bitstreams are shown for comparison.

At the lower resolution, both extraction paths have roughly the same RD performance as the single resolution bitstream with only a slight overhead at the base layer. Note however that the PSNR results for both extraction paths are at the base layer 0.2 dB lower and at the highest layer around 0.7 dB lower than for the single resolution bitstream.

At the higher resolution, the *full extraction path* starts at a quality that is on par with the single resolution bitstream RD performance. Since the bitstream for *full extraction path* depends on the highest layer of the lower resolution, it starts at a bitrate of 2,134 kbps. Subsequent enhancement layers do not increase the quality of the bitstream; rather the first enhancement layer even reduces the quality.

On the other hand, the *partial extraction path* starts at a low quality but increases almost to the quality of the single resolution bitstream for the highest layer. Even with the low starting quality, we argue that the *partial extraction path* is far better suited for multi-resolution SVC streaming.

The VQM results for both extraction paths at resolutions 960x540 and 1920x1080 are shown in Figure 33. Again, there is only negligible loss at 960x540 (similar to Figure 32 (a)). Since the lower resolution has a target bitrate of 3,500 kbps at the highest layer and the higher resolution starts at 4,000 kbps, the *full extraction path* is able to meet that target bitrate and the quality increases with enhancement layers at the higher resolution. Still, we consider the *partial extraction path* to be better suited for spatial scalability in SVC streaming.



(a) (b)

Figure 32: VQM results of spatial scalability for the VSS encoder. The lines labeled *VSS CBR 2 res* represent single bitstreams ranging over both resolutions (a) 640x360 and (b) 1280x720 [2].

In terms of PSNR, the average quality loss due to coding overhead across all sequences and both resolution pairs are shown in Table 9 for the *partial extraction path*. As spatial scalability only yields around 24.6% reduction of storage requirements compared to , we argue that *hybrid SVC-DASH* with one SVC bitstream per resolution is better suited for the given use case. Moreover, *hybrid SVC-DASH* makes a clear distinction between quality scalability – valuable for dynamic adaptation – and different resolutions supporting heterogeneous devices.

Due to an encoder error, we were not able to encode a bitstream with all four resolutions with the VSS encoder.

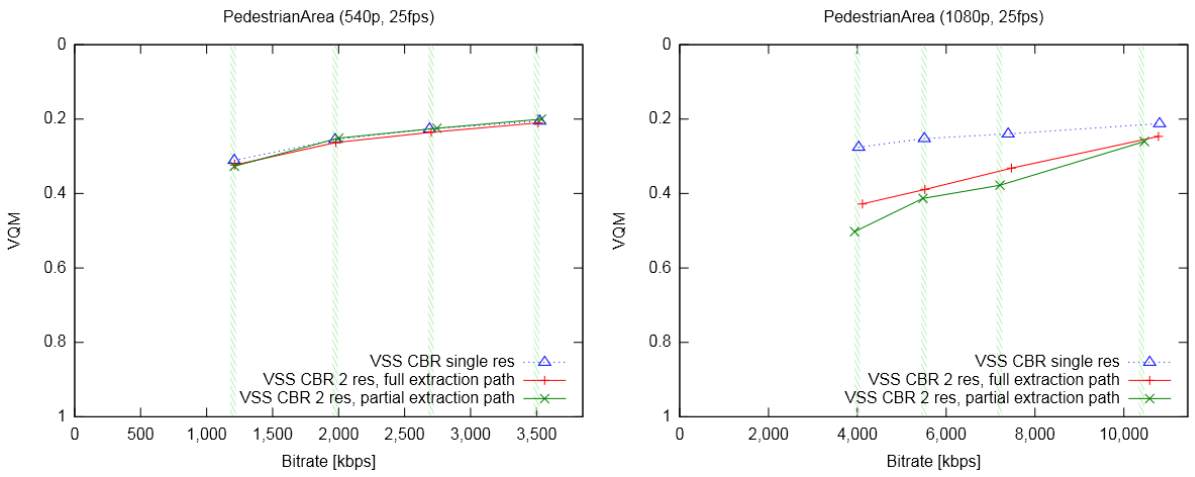
3.5.2.3 Combination of CGS and MGS

In the following test, we evaluate the RD performance for combining CGS and MGS modes in one bitstream. The encoding configuration comprises 4 CGS layers and 4 MGS layers, resulting in 16 quality layers per stream.

Figure 34 shows the PSNR results for the combination of CGS and MGS for the *PedestrianArea* sequence encoded with the bSoft encoder with the QP at the highest layer set to 28. For comparison, PSNR results of the bitstream with 4 CGS layers (labeled *bSoft 4CGS*) and the bitstream with 4 MGS layers (labeled *bSoft 4MGS*) are also shown. The combination of CGS and MGS (labeled *bSoft 4CGSx4MGS*) is depicted with lines that show the possible extraction paths for each quality layer. For

Table 9: PSNR loss for spatial scalability.

Resolution	Layer 0	Layer 1	Layer 2	Layer 3
Resolution 1	0.13 dB	0.25 dB	0.31 dB	0.47 dB
Resolution 2	2.54 dB	2.64 dB	3.00 dB	0.77 dB



(a) (b)

Figure 33: VQM results of spatial scalability for the VSS encoder. The lines labeled *VSS CBR 2 res* represent single bitstreams ranging over both resolutions (a) 960x540 and (b) 1920x1080 [2].

example, starting at the base layer, we can either add one MGS layer, resulting in the layer with DQT=(0,1,0), or can add one CGS layer in order to obtain the layer with DQT=(1,0,0). From either of these two layers, the layer with DQT=(1,1,0) can be reached.

Due to the sharp decrease of PSNR for lower layers for MGS mode as observed in Sections 3.4.4 and 3.5.2.1, also the combination of CGS and MGS suffers from this behavior along MGS layers. Thus, the depiction of PSNR results resembles a grid, where MGS layers form the (almost) vertical lines. This also means that the bitstream contains many extraction points that just have a high bitrate but very low PSNR. In particular, the layers with DQT values of (1,0,0), (2,0,0), (3,0,0), (1,1,0), (2,1,0), and (3,1,0) are not useful for adaptation. For example, the layer with DQT=(3,0,0) has a PSNR of 33.23 dB at 13,578 kbps, while the layer with DQT=(0,2,0) has a PSNR of 35.86 dB at only 6,770 kbps. We conclude that out of the entire 16 SVC layers only the 10 layers forming the outer curve of *bSoft 4CGSx4MGS* are useful for adaptation but at poor overall RD performance.

The discussed configuration of 4 CGS layers and 4 MGS layers was not supported by the tested version of the VSS encoder. The configurations of the JSVM and MainConcept encoders do not allow for such combination of CGS and MGS.

3.6 Conclusions

In this chapter, we have investigated encoding guidelines of dominant industry solutions for MPEG-AVC-based media streaming and devised SVC encoding guidelines therefrom. We have validated these guidelines together with further evaluations of encoding configurations of high-definition video content relevant for adaptive media streaming in content-aware networks. Our tests have also highlighted

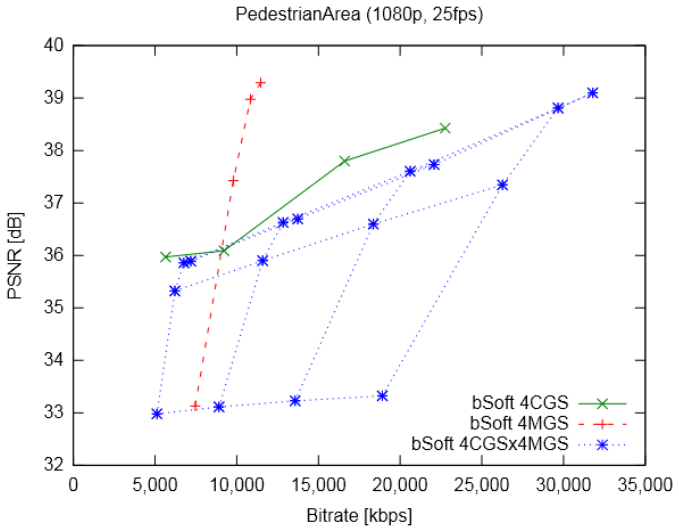


Figure 34: PSNR results for combination of CGS and MGS for the bSoft encoder [2].

characteristics of various encoders. Furthermore, we have investigated deployment options of SVC for DASH with a special focus on scalability options. Additional performance evaluations of major encoder implementations with HD content have focused on SVC-DASH.

Our evaluations on HD SVC encoding performance show that CBR as well as fixed QP rate control modes yield solid quality for the devised bitrate suggestions for all resolutions. Our findings also indicate that it is more suited for media streaming to encode one SVC stream per spatial resolution rather than a single stream comprising all resolutions. For several encoders, the number of SVC layers at 1080p resolution induces higher bitrate overheads than anticipated. We also found that some encoders yield better RD performance in VQM than PSNR results.

Additionally, our findings suggest that a hybrid SVC-DASH approach with one SVC bitstream featuring quality scalability per resolution provides a good trade-off between the advantages of SVC and its coding overhead. Furthermore, we tested the combination of CGS and MGS modes in one bitstream. The results show that only some combinations of layers are useful for adaptation but at poor overall RD performance.

Based on our study, encoding recommendations and evaluation results for high-definition SVC streaming can be summarized as follows:

- Bitrate recommendations for SVC at seven common resolutions are given in Table 3. The recommendations were devised from an extensive survey of industry solutions. A typical streaming session would comprise six to twelve SVC extraction points (or conversely different AVC streams) at two to four resolutions.
- The bitrate recommendations have been validated for various encoders. The tested video sequences show a significant impact of a sequence's Spatial Information on the coding efficiency.

- For multiple MGS enhancement layers, each additional layer induces a coding overhead of slightly above 10%, depending on the encoder.
- The JSVM reference software outperforms proprietary encoders in terms of coding efficiency. Proprietary encoders are up to two orders of magnitude faster than the JSVM.
- For Dynamic Adaptive Streaming over HTTP, we propose a *hybrid SVC-DASH* approach that features one independent SVC base layer at each resolution. The approach separates bitrate adaptation from support of heterogeneous devices. Compared to a single SVC bitstream, *hybrid SVC-DASH* provides around 2.2 dB higher PSNR quality at the highest resolution for quality layers below the highest layer, subject to a moderate increase in storage requirements.
- Even though some proprietary SVC encoders support the combination of CGS and MGS quality scalability mechanisms, such a combination has shown unnecessary coding overhead and little added value in terms of suitable extraction points for adaptation.

There are several aspects to SVC encoding and to video coding for streaming in general that have not been discussed in this chapter.

In our evaluations, we have assumed that SVC is transported as elementary streams. In practice, video data is often encapsulated by a container format such as MPEG-2 Transport Stream (TS), AVI, Matroska Multimedia Container (MKV), or MP4. Container formats are important for interleaving audio and video data, providing stream access information and streaming hints as well as other metadata. The container format can affect the bandwidth requirements for streaming as investigated by Kofler et al. [62]. A low overhead container format for adaptive HTTP streaming is proposed by Riiser et al. [130]. We decided to rely on elementary streams in order to avoid any interference from the container format overhead.

Media segmentation is an important aspect for adaptive HTTP streaming. It controls the flexibility for consecutive adaptation operations. While DASH solutions typically deploy segments of 2-10 seconds [86][93], the sheer number of segment files for content of long durations and small segment durations can cause file system and network overhead [131]. For example, a 2-hour movie at 2 second segments and 6 different representations has 21,600 individual files. Alternatively, DASH also supports HTTP GET requests with byte ranges, thus not requiring for segments to be stored in individual files. The byte ranges are indicated in the MPD. Media segments have to be cut at GOP boundaries in order to ensure that each segment can be processed and decoded independently. The segment duration also limits the maximum GOP size, which, in turn, influences the coding efficiency. We configured all encoders in our tests to emit IDR frames at a fixed rate as explained in Section 3.3.1 in order to support fixed segmentation.

Many adaptive media streaming solutions calculate the current streaming rate based on available network bandwidth and the expected media bitrate. Such adaptation logics often assume that the bitrate does not vary too much during the streaming session. Therefore, the content is often encoded in CBR mode. While the fixed QP rate control mode is supported by all tested SVC encoders, CBR support is still sparse (cf. Section 3.4.1). The traffic variability of an SVC bitstream (in particular for fixed QP rate control mode) is also a challenge for media streaming [132]. Different sizes of coded frames can lead to bursts of video traffic, depending on the spatio-temporal complexity of a scene. For (non-live) HTTP-based streaming, traffic variability only matters on a per-segment basis. Intuitively, the per-segment traffic variability is lower than that of individual frames. For SVC, DASH clients request the base layer prior to the enhancement layers of a segment. If an enhancement layer is not fully downloaded due to traffic variability, it will merely result in the playback of a lower quality. Additionally, typical DASH clients buffer three or more 2-second segments [61], further alleviating the impact of traffic variability. As the segment size can be indicated in the MPD (via byte ranges), the adaptation logic may consider traffic variability in its decisions. Due to the short duration of the test sequences (250 frames), we did not evaluate the bitrate variability in our tests. Nevertheless, both rate control modes, CBR and fixed QP are subject to changing video quality depending on a scene's spatio-temporal complexity as can be observed in Sections 3.4.1 and 3.5.2.1. Some encoders offer VBR or average bitrate (ABR) rate control modes. Another option for rate control is equitable quality streaming [133], which finds for each frame a QP for encoding that yields a predefined quality. This way, the entire sequence has roughly the same, constant video quality. While fixed QP is typically used in video quality evaluations, CBR is often used in industrial streaming solutions [95][97].

At the time of writing, MPEG is also developing implementation guidelines for DASH [134], including content generation guidelines for GOP structures, stream access points, and enabling bandwidth adaptation.

4 SVC Tunneling

4.1 Introduction

Today's omnipresent demand for access to multimedia content via diverse devices places new challenges on efficient content delivery. Scalable media coding formats, such as SVC, enable efficient content adaptation within the network and reduce network resource utilization in multicast scenarios. *SVC tunneling* describes the deployment of scalable media coding in the network, independently of the coding formats supported at the sender or receiver side. Assume that the content has been encoded in a non-scalable coding format a-priori on the sender side and/or that the receiver uses a legacy device, which does not support scalable media coding formats either. How can in-network adaptation be applied to improve media delivery in such a scenario? With the proposed *SVC tunneling* approach, the content is transcoded to SVC at the sender side, allowing for SVC-based in-network adaptation during delivery, and is transcoded back to a non-scalable coding format at the receiver side for device-independent access. An obvious downside of this approach is the loss in video quality due to transcoding. This chapter will introduce and evaluate the concept of SVC tunneling, investigating the trade-off between the transcoding-induced quality loss and the benefits of SVC streaming.

The work presented in this chapter is published in [3], [4], [5], and [9].

While the SVC extension of AVC has proven to be a useful tool for the advanced delivery of video content, it has not yet found major adoption in practice (with perhaps the exception of Google+ Hangout as discussed in Section 3.2.1). The deployment of SVC for content delivery enables bandwidth savings for multicast scenarios [76][77] and facilitates more robust video transport in content-aware networks [45]. Many devices, however, do not support scalable video formats and rely on non-scalable formats, e.g., MPEG-4 AVC, or even legacy formats like MPEG-2. One solution to the problem of deploying SVC streams in such an environment is the transcoding of video streams at the ingress and egress points of the network and the deployment of SVC tunneling within the network, thus enabling SVC content delivery and device-independent access.

For a better understanding of the *SVC tunneling* concept and its development within ALICANTE, this section briefly highlights the relevant aspects of the ALICANTE architecture from Section 2.3. Towards the goal of an advanced *media ecosystem*, an SVC (layered-multicast) tunnel is developed in ALICANTE, inspired by IPv6-over-IPv4 tunnels.

Video multicast (e.g., for IPTV services) to heterogeneous devices can be traditionally realized by two approaches; a third approach is provided by the ALICANTE architecture.

The first approach is to use a non-layered video format (such as AVC or MPEG-2) and send all content representations simultaneously. This approach is referred to as simulcast mode. Content representations may comprise different resolutions or different quality versions of a video. For simulcast mode, it is also possible to send the content in different video formats, thus enabling format independence for the receiver. The bandwidth requirement for delivering the content is the sum of the bitrates of all representations being consumed by at least one user.

The second approach is to use SVC and to configure the SVC layers to fit these representations. In such a (receiver-driven) layered mode, the maximum required bandwidth is the bitstream up to the highest SVC layer being consumed by at least one user. Compared to AVC simulcast, this mode can reduce the required network capacity by around 18% [76]. The use of SVC at the network layer also empowers a content-aware network to perform efficient in-network adaptation, e.g., for QoS management.

ALICANTE introduces a third approach for multicast content delivery as illustrated in Figure 35. Within the content-aware network, only scalable media resources, such as SVC, are delivered, allowing for in-network adaptation at MANEs. If the content at the server side originally has been encoded in a non-scalable legacy video format, e.g., MPEG-2, it is transcoded to SVC at the Home-Box layer before delivery. Layered multicast is deployed at the CAN layer. When arriving at the client side, the scalable media resources can be transcoded to a format supported by the end-user terminal (e.g., again MPEG-2). The Home-Box, which is a next generation interconnected home-gateway, performs the transcoding. The Home-Box sends the transcoded content via unicast through the home network towards the terminal for consumption. This approach combines the format independence of the simulcast mode with the capabilities for bandwidth reduction and efficient in-network adaptation. However, this approach reduces the video quality due to its potentially two transcoding steps. This quality impact of the full SVC tunneling approach with both transcoding steps is investigated in Section 4.3. There are two variations of the SVC tunneling approach which require only one transcoding step. First, if the content is originally encoded in SVC, there is no need for transcoding at the server side. Second, if transcoding to SVC is performed at the server side but the terminals support SVC, the second transcoding step can be omitted.

For the evaluations in this chapter, we assume that both content provider and end user require the same video format (i.e., MPEG-2). The ALICANTE architecture is more general in this respect and allows for different video formats at the sender and receiver sides as detailed later on in Section 5.3.

SVC tunneling also enables advanced QoS/QoE management in content-aware networks. MANEs distributed across the network can perform in-network adaptation [45] on the SVC bitstream in order to adjust to changing network conditions.

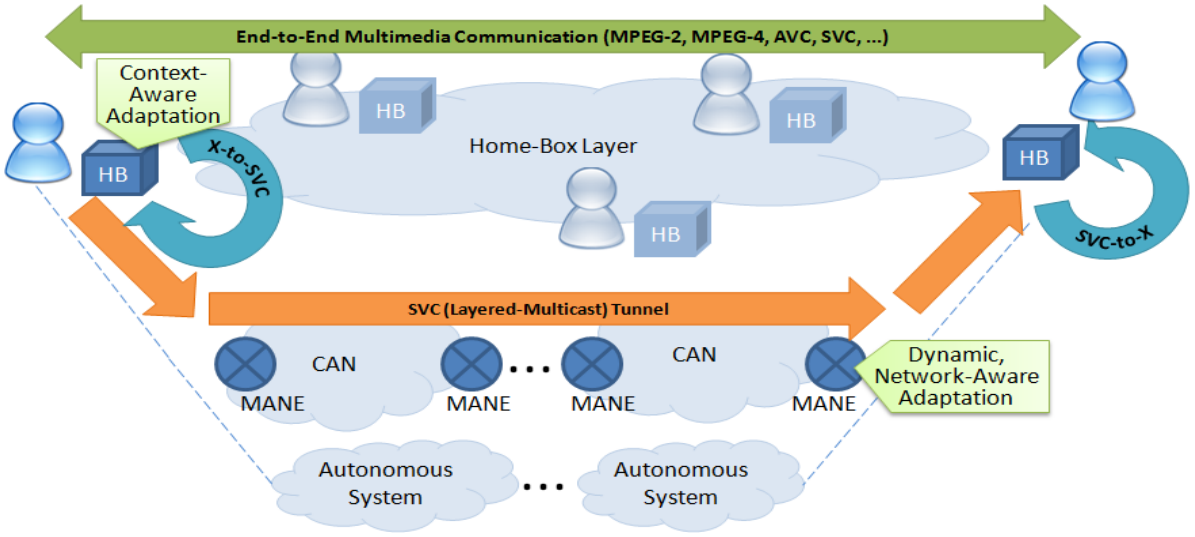


Figure 35: Adaptation Framework Overview [5].

4.2 Concept and Considerations

In this section, we discuss the main aspects of SVC tunneling, comprising SVC transcoding and related work, variants for partial SVC tunneling, as well as quality, rate control, and delay considerations. SVC tunneling provides device-independent media access through transcoding. Any video coding format can be used at the server or client side, e.g., MPEG-1 [135], MPEG-2 [31], Motion JPEG 2000 [136], MPEG-4 Visual [137], AVC [23], VC-2 [138], VP8 [139], or HEVC [140]. The transcoding speed and quality depends on the selected video coding format. As SVC is an extension of AVC, transcoding can be performed much faster and with less quality impact than for other formats. For our evaluations, we chose MPEG-2 on the server and client sides for the following reasons. First, it is still a popular legacy video coding format due to its adoption in DVDs and by the digital television industry. Second, transcoding to and from SVC is challenging in terms of quality loss and delay. Thus, the use of MPEG-2 can be regarded as a realistic worst-case scenario for SVC tunneling. Our evaluations establish a baseline for SVC tunneling, from which its efficiency can be improved for other video coding formats.

4.2.1 SVC Transcoding

SVC follows a layered coding scheme comprising a base layer and one or more enhancement layers providing scalability along various dimensions [27]. Three basic scalable coding modes are supported, namely spatial scalability, temporal scalability, and SNR scalability, which can be combined into a single coded bitstream.

When it comes to compression, SVC with two layers of either quality scalability or dyadic spatial scalability requires about 10% more bitrate than single layer AVC for the same video quality [33]. But compared to MPEG-2, which requires approx. 170%

more bitrate than AVC [141], SVC still provides a bitrate reduction of about 59% (calculated from $1 - \left(\frac{1+0.1}{1+1.7}\right)$) with respect to MPEG-2. For an SVC bitstream with 4 layers, the theoretical bitrate reduction would be around 52%.

In the following, we discuss related work in the area of video transcoding and focus on transcoding from MPEG-2 to SVC and vice versa.

4.2.1.1 Transcoding to SVC

The simplest but slowest architecture of transcoding between two video formats is accomplished by fully decoding the video and then re-encoding the pixels into the target format, which is known as pixel domain transcoding (PDT), cascaded transcoding, or full transcoding [142][143]. It usually provides the best quality and is used as a reference for more advanced transcoding mechanisms. Since the video has to be fully decoded and fully re-encoded, this technique is rather slow and computationally expensive. The computational complexity can be reduced by using information from the coded source video to create the target video. For example, the motion vectors can be extracted and mapped to the target coding format. Advanced transcoding is usually performed in the transform domain. The transform coefficients are extracted from the encoded source video and converted to the transform coefficients of the desired format. This technique is called transform domain transcoding (TDT) [142]. TDT is considerably faster than PDT but usually introduces higher quality losses [144].

However, each format has its own way of encoding videos. For example, MPEG-2 uses DCT, while H.264/AVC uses low-complexity integer transform (HT), and the conversion between them is not trivial [145]. Furthermore, both formats deploy different coding tools (e.g., AVC introduces intra-prediction and allows multi-frame references for inter-frame prediction) [146][147]. This leads to specialized transcoders for each format conversion.

A special case of transcoding is bitstream rewriting, which converts the video from one format to another without any quality losses. Bitstream rewriting is only possible if both video formats use the same bitstream syntax and coding techniques, which is the case for AVC and SVC. De Cock et al. have developed a technique for low-complexity AVC-to-SVC transcoding in [148] and subsequently improved it into an AVC-to-SVC bitstream rewriting technique in [149] and [150]. This rewriting provides perfect reconstruction at the (highest) enhancement layer. A multi-layer control mechanism for the trade-off between quality and bitrate at the base layer was added in [151] together with improved motion data refinement. The proposed technique only targets SNR scalability, spatial scalability is not supported. Note that lossless bitstream rewriting still increases the bitrate even at perfect reconstruction if the target format has a lower coding efficiency, as it is the case for AVC-to-SVC rewriting. For further details on AVC-to-SVC transcoding, the interested reader is referred to [152].

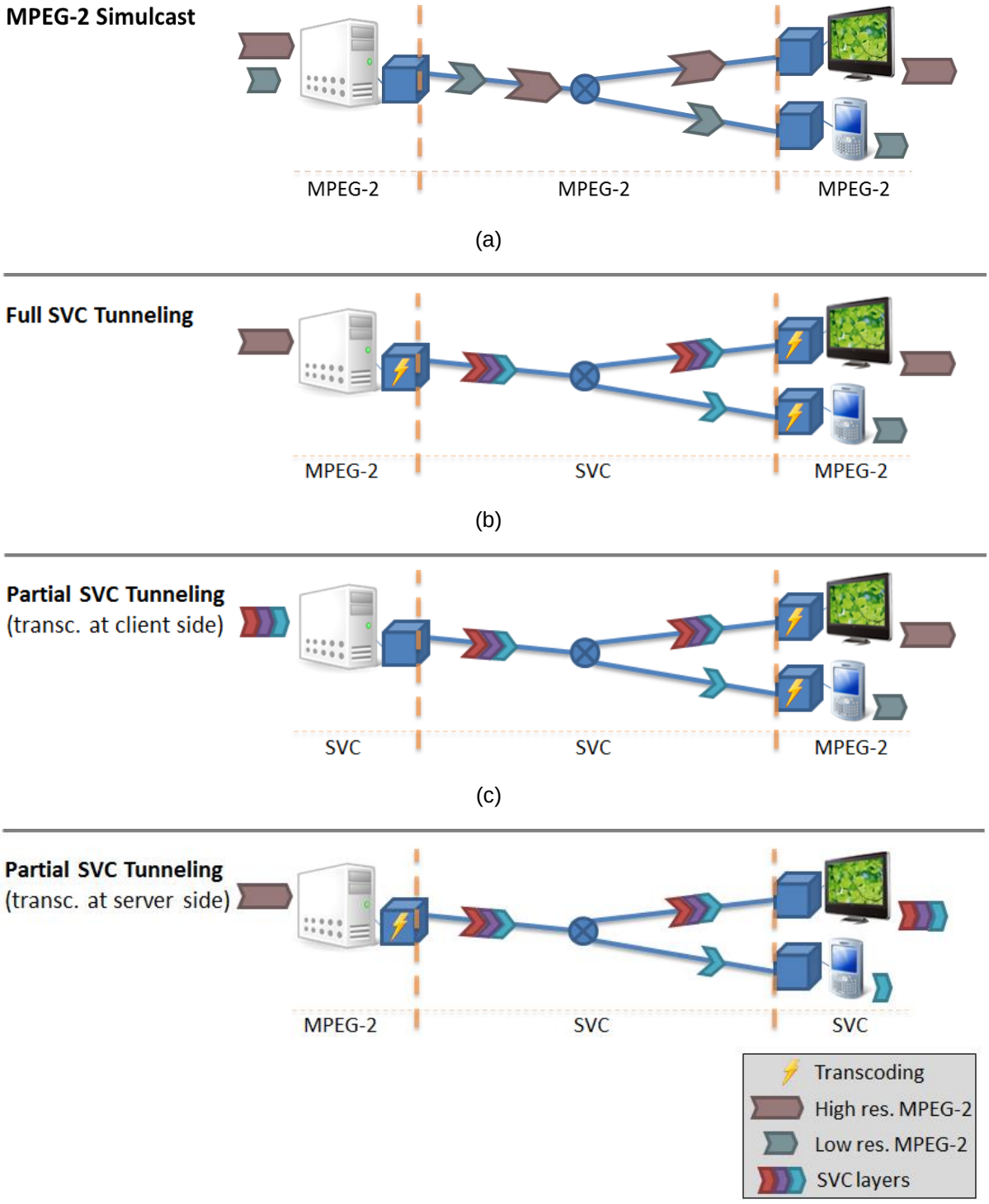


Figure 36: Multicast streaming scenarios for (a) reference MPEG-2 simulcast, (b) full SVC tunneling, (c) partial SVC tunneling with SVC-encoded source content, and (d) partial SVC tunneling with SVC-capable end-user terminals.

While a variety of transform domain transcoders from different formats to AVC exist (e.g., from MPEG-2 [153][144][154][155][146][156]), for SVC only transcoding and rewriting techniques from AVC as the source format have been researched so far [150]. To the best of our knowledge, no MPEG-2-to-SVC TDT has been addressed in any research so far. In order to transcode from MPEG-2 to SVC, either PDT or

cascaded TDTs can be deployed. In the first case, the video is decoded from MPEG-2 and then re-encoded to SVC. In the latter case, a fast MPEG-2-to-AVC transform domain transcoder and a fast AVC-to-SVC rewriter are cascaded.

In this work, we focus on MPEG-2-to-SVC PDT rather than cascaded TDTs. The PDT architecture is more generic and allows for transcoding from virtually any source format to SVC by simply plugging in the appropriate decoder. PDT also establishes a proper baseline for MPEG-2-to-SVC transcoding.

4.2.1.2 Transcoding from SVC

The SVC base layer is backward-compatible to AVC. The full SVC bitstream can be converted to AVC through lossless bitstream rewriting [157][158][159]. Note that such bitstream rewriting requires certain modifications to inter-layer intra prediction and residual prediction in the encoding process as discussed in [158] and is only applicable to SNR scalability [159]. Different techniques for SVC-to-AVC transcoding supporting spatial scalability were proposed in [160] and [161]. Sablatschan et al. have evaluated the performance of SVC-to-AVC bitstream rewriting on a MANE in [162].

To the best of our knowledge, transcoding techniques from SVC have only been researched for AVC as target format. For transcoding SVC to other target formats, such as MPEG-2, two architectures are possible, similar to the X-to-SVC transcoding architectures described above. The first architecture uses PDT, fully decoding the SVC bitstream and re-encoding it to the target format. The second architecture comprises cascaded TDTs, i.e., SVC-to-AVC bitstream rewriting followed by fast TDT from AVC to the target format (e.g., AVC-to-MPEG-2 TDT [163][164]). Again, we have chosen the more general PDT architecture for SVC-to-MPEG-2 transcoding in order to establish a proper baseline for future research.

4.2.1.3 Repeated Transcoding

In multicast scenarios, the content may have been originally encoded to a non-scalable legacy format like MPEG-2 (e.g., DVD-Videos) and also the user terminals may require MPEG-2 for playback. We have proposed an SVC tunnel for content delivery in the ALICANTE architecture that could be deployed in order to enable QoS/QoE management at the network and possibly to reduce network load. However, such an SVC tunnel requires two transcoding operations, first MPEG-2-to-SVC transcoding at the server side and second SVC-to-MPEG-2 transcoding at the client side (i.e., the Home-Box). Since the PSNR is computed from the mean squared error (MSE), which contains quadratic terms, it is not possible to estimate the quality impact of this repeated transcoding by just accumulating the PSNR values of each transcoding run.

4.2.2 Partial SVC Tunneling

The full SVC tunneling approach assumes that the content at the server is pre-encoded in a non-scalable format and that the end-user terminal does not support SVC. If either the content is available in SVC or the end-user terminal supports SVC, we speak of *partial SVC tunneling*, which requires only one transcoding step. The different scenarios are illustrated in Figure 36.

Partial SVC tunneling obviously enables higher bandwidth efficiency and lower quality loss than the *full SVC tunneling* approach.

4.2.3 Delay and Rate Control Considerations

The transcoding steps for SVC tunneling introduce quality loss and additional delay. Delay may not matter for non-real-time media services, like pre-recorded TV broadcasts or Video on Demand services, if the content can be transcoded and prepared in advance. On the other hand, especially for live content low transcoding delay and high processing performance of the transcoding equipment are crucial. Typically, video encoders have higher computational complexity than decoders. SVC encoder speeds were evaluated in Section 3.4.6. The real-time constraints for live streaming and video conferencing scenarios can be met either by more powerful equipment or by reducing the computational complexity of the encoding process (typically at the expense of RD performance). In our evaluations, we will focus on the traditional architecture (i.e., high computational complexity at the encoder) due to the characteristics of available encoders.

MPEG-2 encoders are considerably faster than SVC encoders, due to lower encoding complexity and their implementation maturity. For the SVC tunneling chain, transcoding delay is accumulated from MPEG-2 decoding, SVC encoding, SVC decoding, and MPEG-2 encoding. In non-live scenarios, we assume that the content has been transcoded to SVC prior to streaming, thus reducing that part of the delay. In general, the client side must receive an entire GOP before the SVC decoder is able to decode the video due to the prediction structure inside a GOP. Similarly, the MPEG-2 encoder will need roughly one GOP for inter-frame prediction before the stream can be emitted to the end-user terminal. This structural delay could be reduced to zero by avoiding prediction from future frames at the expense of lower coding efficiency [27]. The evaluations presented in this chapter focus on quality impact of SVC tunneling, leaving aside any delay aspects, real-time constraints, and processing performance.

Since most streaming scenarios of SVC tunneling require on-the-fly transcoding from SVC to MPEG-2 on the client side (i.e., at the Home-Box), the maximum supported resolution and frame rate are limited by the decoder and encoder implementations as well as the equipment performance. Details on the transcoding speed in an integrated test-bed and on transcoding delays will be reported later on in Section 5.5.2.

The alignment of encoding configurations between source and target material is an important aspect of transcoding. Maintaining the same GOP size and rate control mode during transcoding prevents unnecessary quality loss. In our evaluations, we will investigate the suitability of fixed QP and CBR rate control modes for SVC tunneling and how the rate control should be configured for transcoding. A first intuitive approach that we test for CBR mode is to apply the same bitrate as the source material for the target material. However, this does not take the different RD performances of the source and target codecs into account. While increased bitrates of the target material reduce the quality loss, they also affect the bandwidth efficiency of the approach. Thus, SVC tunneling configurations are always a trade-off between quality loss one is willing to accept and achievable bandwidth efficiency.

4.3 Evaluations

In the following tests, we gradually improve the transcoding configurations for both transcoding steps, MPEG-2-to-SVC and SVC-to-MPEG-2, also investigating the quality loss characteristics at different target qualities.

Note that, at least in theory, SVC tunneling with AVC as source and target formats could be achieved without quality loss, based on lossless AVC-to-SVC bitstream rewriting [149][150][151] and lossless SVC-to-AVC bitstream rewriting [157][158][159]. In our evaluations, we rather focus on MPEG-2 as the source and target formats.

The first evaluation (Section 4.3.1) was performed to investigate the overall feasibility of SVC tunneling. It was therefore performed on only two test sequences, while further evaluations in Sections 4.3.2 and 4.3.3 use four test sequences.

As streaming scenarios require the SVC-to-MPEG-2 transcoding at the client side to be performed in real-time, we limited our evaluations to a resolution of 352x288 (see also Annex D).

This chapter focuses on the quality impact of SVC tunneling. The impact on the delay due to transcoding is documented later on in Section 5.5.2.1.

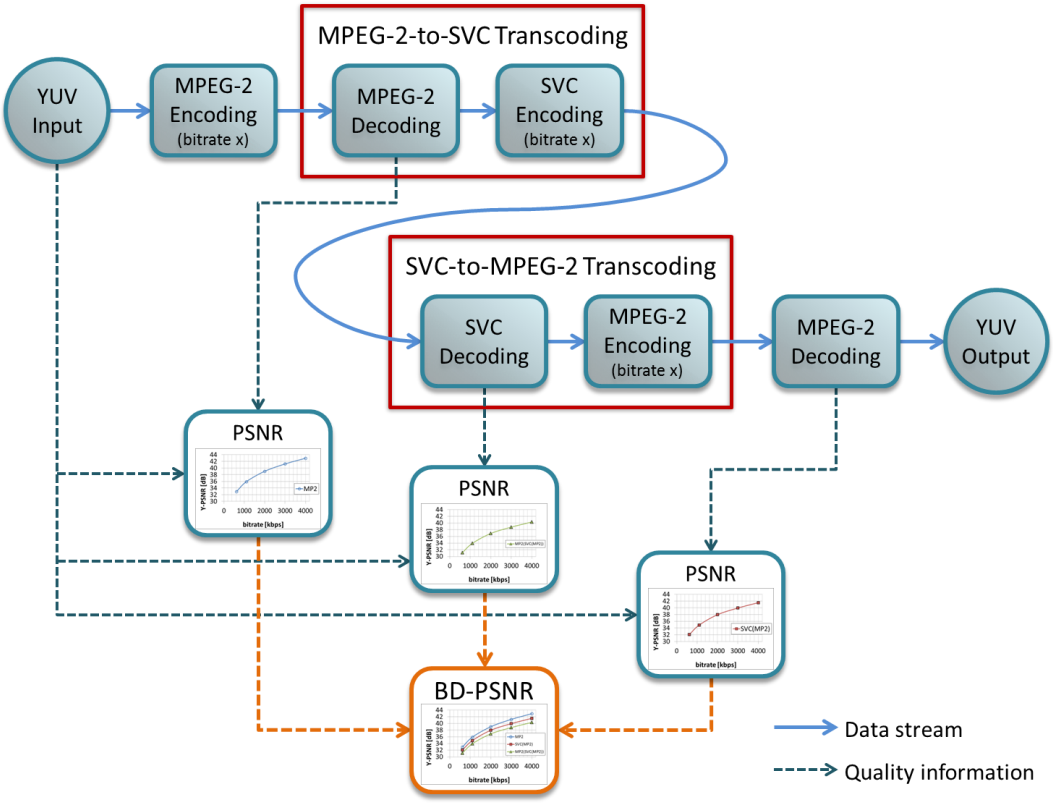


Figure 37: Test-bed setup for same-bitrate evaluation of SVC tunneling.

4.3.1 Same-Bitrate Evaluation

We first performed an evaluation based on the primitive configuration of maintaining the same bitrate for both transcoding operations in order to establish a baseline for further tests.

4.3.1.1 Initial Test-Bed Setup

In order to evaluate the quality impact of the repeated transcoding of SVC tunneling, we performed the transcoding operations on two standard test sequences, *Foreman* and *Mobile* (CIF at 30 fps, 300 frames).

In the first step, each sequence was encoded from raw YUV to MPEG-2 using *FFmpeg* version SVN-r25599 [165] and its *mpeg2video* codec. In the second step, the output stream was then transcoded by decoding it using the *GPL MPEG-1/2 DirectShow Decoder Filter* Version 0.1.2 [166] and encoding it to SVC using the *MainConcept SVC/AVC/H.264 Video Encoder* Version 1.0.0.236699 [108] DirectShow filter. The SVC bitstream has three layers with the following encoder configuration. The base layer is specified at QCIF at 30 fps and 15% of the entire target bitrate and the first enhancement layer with CIF at 30 fps and 30% of the entire target bitrate. The second enhancement layer (i.e., highest layer) is specified with

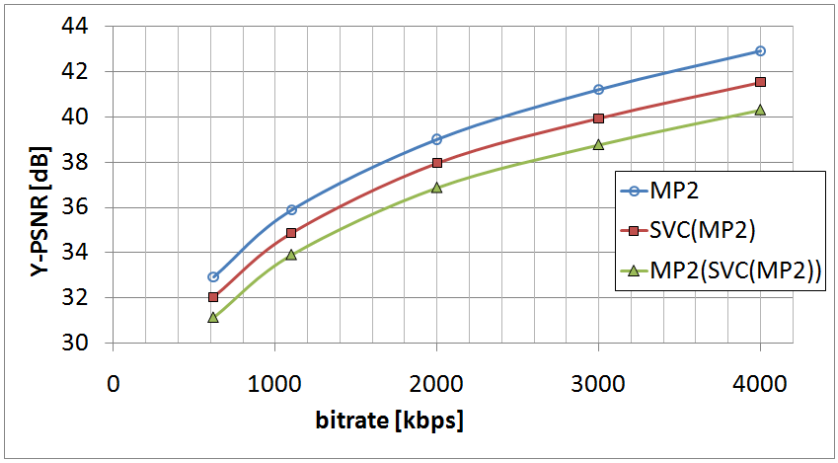


Figure 38: Y-PSNR for repeated transcoding of *Foreman* sequence [3].

CIF at 30 fps. (Note that this configuration does not comply with the encoding recommendations provided in Chapter 3 and was only used in this first evaluation.)

The bitstream was transcoded back to MPEG-2 in the final step by decoding it using the *MainConcept SVC/AVC/H.264 Video Decoder* Version 1.0.0.236699 DirectShow filter and encoding it using *FFmpeg* and the *mpeg2video* codec.

The encoding of each sequence was performed at several target bitrates. At all three steps, encoding was performed at fixed target bitrates in CBR mode, i.e., the video was encoded to MPEG-2 at the same target bitrate as it was transcoded to SVC and transcoded back to MPEG-2. For example, if the video was initially encoded to MPEG-2 at a target bitrate of 2,000 kbps, it was transcoded to SVC at 2,000 kbps target bitrate and transcoded back to MPEG-2 at this same target bitrate.

The PSNR was always measured against the original raw YUV video. The differences in PSNR and bitrates between two steps were calculated as Bjontegaard Delta (BD) [167][168]. The BD measures the average distance between two RD curves along the PSNR and bitrate axes. An illustration of the test-bed setup is given in Figure 37.

4.3.1.2 Experimental Results

The RD curves of the repeated transcoding are shown in Figure 38 for the Foreman sequence and in Figure 39 for the Mobile sequence, each for the extraction of all SVC enhancement layers. The RD curve after MPEG-2 encoding is labeled *MP2*, the RD curve after MPEG-2-to-SVC PDT is labeled *SVC(MP2)*, and the RD curve for the final SVC-to-MPEG-2 PDT is labeled *MP2(SVC(MP2))*.

For the Foreman sequence, both transcoding steps (MPEG-2-to-SVC and SVC-to-MPEG-2) have nearly the same impact on the video quality. Conversely, the Mobile sequence indicates only slight quality losses for MPEG-2-to-SVC transcoding (BD-PSNR of 0.5 dB between first and second curve) compared to the impact of SVC-to-MPEG-2 transcoding (BD-PSNR of 1.5 dB between second and third curve).

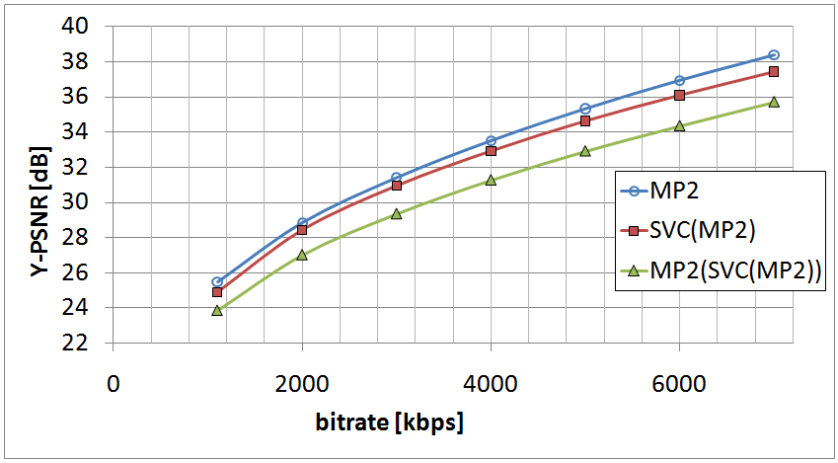


Figure 39: Y-PSNR for repeated transcoding of *Mobile* sequence [3].

The individual and average results are represented as the BD of the PSNR and bitrate in Table 10. On average, the repeated transcoding results in a total PSNR drop of 2.1 dB or conversely a bitrate increase of 43% in order to compensate for the PSNR drop.

Based on these results, the bandwidth requirements for a multicast streaming architecture can be estimated for three scenarios as shown in Figure 40. The MPEG-2 simulcast mode, in which 3 quality versions of the content (as specified in Section 3.3) are streamed, requires 145% of the bitrate of the original MPEG-2 video. The simulcast mode is depicted as a baseline in Figure 40, labeled Scenario 1. The other two scenarios considered in the figure are the full SVC tunneling mode, labeled Scenario 2, with MPEG-2-to-SVC and SVC-to-MPEG-2 transcoding, and, as Scenario 3, SVC multicast streaming with only MPEG-2-to-SVC transcoding at the ingress point of the network. SVC content delivery in both latter scenarios reduces the bandwidth requirements at the core network by approx. 31% w.r.t. the simulcast mode, at the expense of degraded video quality (-2.1 dB for Scenario 2 and -0.8 dB for Scenario 3). Scenario 3 assumes that the end-user terminals also support SVC and thus no SVC-to-MPEG-2 transcoding is required.

Note that the content delivery in all three scenarios is based on the same video at the sender. In order to obtain equal video quality results at the end-user terminals for MPEG-2 simulcast mode and SVC layered multicast, the bitrate of the MPEG-2 video for simulcast mode (Scenario 1) could be throttled according to Table 10. However, this would imply deliberately sending suboptimal video quality to simulcast mode users, which only makes sense if the available network bandwidth is scarce.

4.3.1.3 Discussion of Experimental Results

While the SVC tunnel architecture with the presented test setup provides a moderate reduction of bandwidth over MPEG-2 simulcast, it should be noted that the repeated transcoding in this test setup has used fixed target bitrates for all three operations in order to establish a valuable baseline for further research.

Table 10: Bjontegaard Delta of RD curves for repeated transcoding [3].

***Foreman* sequence:**	**BD-PSNR**	**BD-bitrate**
1st to 2nd curve (MPEG-2→SVC)	-1.1 dB	23%
2nd to 3rd curve (back to MPEG-2)	-1.0 dB	23%
1st to 3rd (MPEG-2→SVC→MPEG-2)	**-2.1 dB**	**51%**
***Mobile* sequence:**	**BD-PSNR**	**BD-bitrate**
1st to 2nd curve (MPEG-2→SVC)	-0.5 dB	8%
2nd to 3rd curve (back to MPEG-2)	-1.5 dB	26%
1st to 3rd (MPEG-2→SVC→MPEG-2)	**-2.1 dB**	**36%**
Average:	**BD-PSNR**	**BD-bitrate**
1st to 2nd curve (MPEG-2→SVC)	-0.8 dB	15%
2nd to 3rd curve (back to MPEG-2)	-1.3 dB	25%
1st to 3rd (MPEG-2→SVC→MPEG-2)	**-2.1 dB**	**43%**

The current architecture can be improved for both transcoding steps. In the MPEG-2-to-SVC transcoding step, the target bitrate for SVC could be reduced according to SVC's higher coding efficiency over MPEG-2. In theory, it should be possible to reduce the bitrate of an SVC stream with 2 layers by 59% (cf. Section 4.2.1) while maintaining BD-PSNR results similar to fixed target bitrates. However, as coding efficiency may vary depending on the content, the selection of appropriate SVC target bitrates remains a research challenge. The naïve solution is to statically reduce the bitrate for MPEG-2-to-SVC transcoding by about 59%. Since the coding efficiencies of MPEG-2 encoders have also improved over the years [169], this number is probably obsolete by now. A more elaborate solution would be to encode the raw video to SVC and measure the ratio of MPEG-2 bitrate vs. SVC bitrate in order to determine the appropriate bitrate reduction for that specific content. But this solution is not applicable for scenarios in which the content is only available in MPEG-2 (e.g., DVD-Videos). Another solution would be to steer the transcoding through the QP instead of target bitrate. As noted in Section 4.2.3, we argue that transcoding to SVC should apply the same rate control mode as the original MPEG-2 encoding.

In addition to the architectural enhancements, the particular configuration of the transcoding setup implemented in this section could be improved in several aspects, such as the choice of SVC layer configuration or the deployment of other transcoding components.

The two transcoding steps for SVC tunneling result in a PSNR drop of 2.1 dB. Based on the proposed mapping of PSNR to the MOS in [116], the perceptibility of the PSNR drop, represented as Differential MOS (DMOS) on the Absolute Category

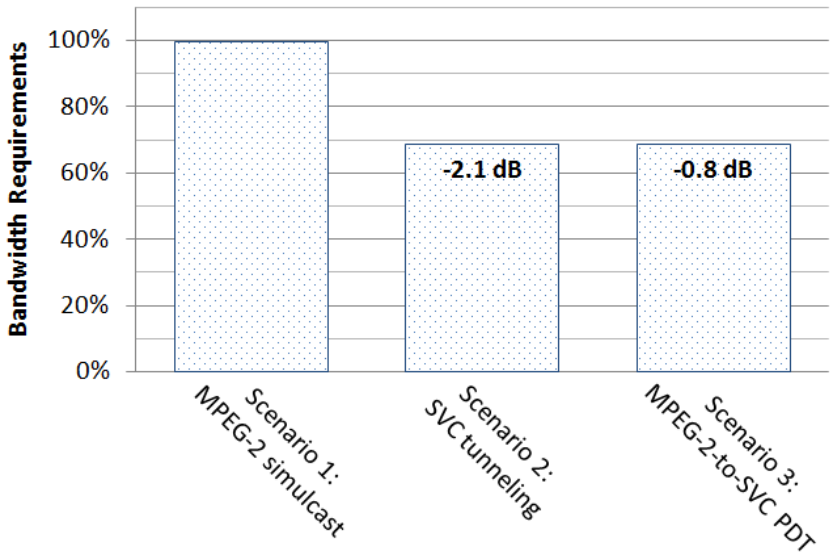


Figure 40: Estimated bandwidth requirements at the core network and corresponding quality degradation for multicast streaming [3].

Rating (ACR) scale from 1 (bad) to 5 (excellent) [125], can be roughly estimated between 4.83 in the best case and 4.63 in the worst case according to Equation (1).

$$DV(PVS) = V(PVS) - V(REF) + 5 \qquad (1)$$

$DV(PVS)$ is the calculated differential viewer score and $V(PVS)$ and $V(REF)$ are the individual viewer scores of the processed video sequence and reference respectively. This equation is also applicable to the MOS because the function for $V(PVS)$ and the arithmetic mean of the MOS are commutative. Note that this is only a first estimate to provide an impression of the perceptibility of the quality reduction. As noted in Section 3.3.2, other mappings between PSNR and MOS have been proposed as well.

In this section, we have performed a first evaluation of the quality impact induced by repeated transcoding at network borders. This transcoding chain results in a total PSNR decrease of 2.1 dB, with around 1/3 of the quality impact attributed to the initial MPEG-2-to-SVC transcoding step. A bitrate increase of 43% (compared to a single MPEG-2 bitstream) is required to compensate the quality loss, which is still less than the necessary bandwidth for MPEG-2 simulcast-based streaming (i.e., $30\% + 15\% = 45\%$ bitrate increase compared to the single bitstream).

The results of two test sequences for an SVC tunneling test-bed setup with CBR encoding and same bitrates for both transcoding operations have been presented. The following sections will ameliorate the test-bed setup with respect to transcoding configurations and will cover more test sequences.

Table 11: SVC layer configurations for initial encoding at CBR and fixed QP rate control modes.

Label	Rate control	Layer 3	Layer 2	Layer 1	Layer 0
Q1	**fixed QP**	16	22	28	34
	CBR	3,000 kbps	2,100 kbps	1,200 kbps	300 kbps
Q2	**fixed QP**	20	26	32	38
	CBR	2,000 kbps	1,400 kbps	800 kbps	200 kbps
Q3	**fixed QP**	24	30	36	42
	CBR	1,500 kbps	1,050 kbps	600 kbps	150 kbps
Q4	**fixed QP**	28	34	40	46
	CBR	1,000 kbps	700 kbps	400 kbps	100 kbps

4.3.2 Comparing Rate Control Modes for SVC Tunneling

4.3.2.1 Test-Bed Setup and Quantization Considerations

One major drawback of the test-bed setup described in Section 4.3.1.1 is the low flexibility due to the same bitrates for both transcoding operations. This configuration does not take different coding performances of MPEG-2 and SVC into account. The configuration also limits the bandwidth efficiency of SVC tunneling considerably.

Furthermore, we found the previous selection of test content and the SVC encoding configurations to be insufficient for further investigations. Two test sequences are too few for a reliable evaluation. As discussed in Chapter 3, we consider the use of quality scalability to be more suitable to most SVC-based streaming scenarios compared to a combination of spatial and quality scalability.

To overcome these limitations, we performed further tests for comparing SVC tunneling of fixed QP encoding mode against CBR encoding mode using the following setup. The test was performed with the test sequences *Foreman*, *Container*, *Hall_Monitor*, and *Stefan*, each having a resolution of 352x288 and 25 fps frame rate. These test sequences were selected to represent a wider range of typical videos than the previous selection. The test sequences were initially encoded to MPEG-2, transcoded in a first transcoding step to SVC using PDT, and in a second transcoding step back to MPEG-2 using PDT. These transcoding scenarios were performed for fixed QP and CBR encodings separately. For comparing the required bandwidth of SVC tunneling with MPEG-2 simulcast, we selected for each extracted SVC layer an MPEG-2 encoding with best matching Y-PSNR.

One challenge in this setup is the selection of a suitable QP or target bitrate for the SVC encoding in the first transcoding step. We chose an experimental approach where the original YUV sequence is encoded to SVC with several target qualities (i.e., QP or target bitrate) and then the configuration that yields a Y-PSNR just above

Table 12: Y-PSNR results of SVC layers for the *Hall_Monitor* sequence with various encoders and rate control modes, adopted from [5].

Target Quality	bSoft (fixed QP)				
	Bitrate [kbps]	***L3 [dB]***	***L2 [dB]***	***L1 [dB]***	***L0 [dB]***
Q1	4482	44.74	33.05	26.89	23.20
Q2	2446	42.03	32.95	26.89	23.20
Q3	1244	39.84	32.77	26.87	23.21
Q4	699	37.83	32.48	26.86	23.23
	MainConcept (CBR)				
	Bitrate [kbps]	***L3 [dB]***	***L2 [dB]***	***L1 [dB]***	***L0 [dB]***
Q1	3095	43.87	42.64	41.03	37.74
Q2	2202	42.30	41.07	39.53	36.30
Q3	1622	40.96	39.75	38.32	35.09
Q4	1058	38.68	37.43	36.03	32.62
	MainConcept (fixed QP)				
	Bitrate [kbps]	***L3 [dB]***	***L2 [dB]***	***L1 [dB]***	***L0 [dB]***
Q1	3270	42.97	39.15	35.94	32.96
Q2	1867	40.10	36.62	33.38	30.39
Q3	1191	37.79	34.22	30.88	27.84
Q4	816	35.46	31.74	28.34	25.30

that of the MPEG-2 encoding is selected. Due to different coding mechanisms, MPEG-2 has a different range of QP values (1-32) than AVC and SVC (0-51). Thus, the mapping between the two is not straight-forward. For the second transcoding step (back to MPEG-2), we applied again the target quality of the initial MPEG-2 encoding.

The SVC encoding was configured with four MGS layers. We tested two industry-grade SVC encoders, i.e., MainConcept v1.5 [108] and bSoft v120403 [103]. MPEG-2 encoding was performed via FFmpeg v0.8 [165]. The bSoft encoder distributes transform coefficients automatically to create MGS enhancement layers. The MainConcept encoder performs re-quantization to obtain those MGS layers. Compared to the highest layer, we reduced the QP by 6 per MGS layer for fixed QP or conversely the target bitrate by 30% (of the total bitrate) for CBR. While Section 3.4.5 has suggested a deltaQP of 2, the test-bed setup of this section covers a higher range of bitrates. Since the bSoft encoder always yielded better RD performance for fixed QP mode, we did not perform CBR mode tests for the bSoft encoder.

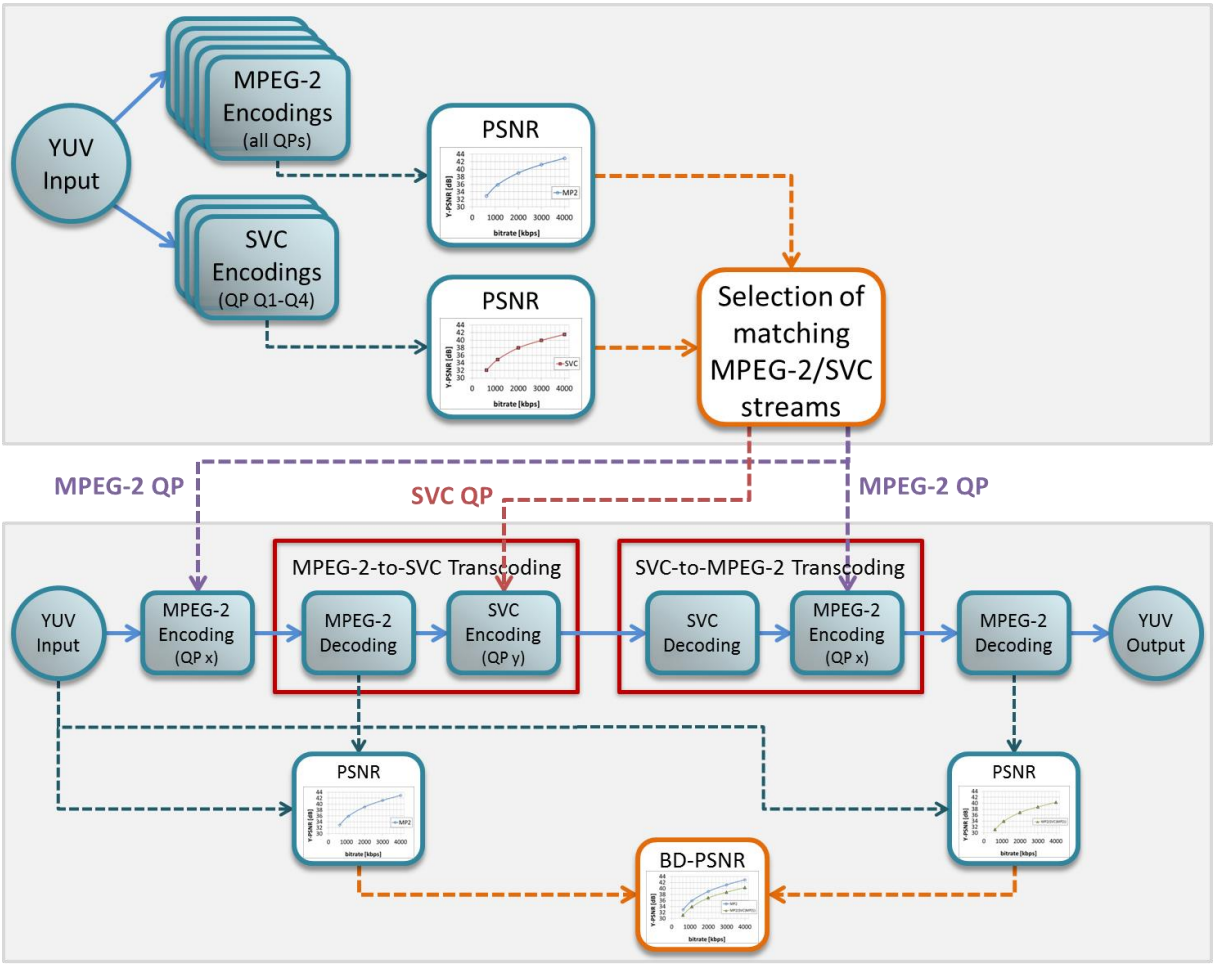


Figure 41: Test-bed setup for QP selection and SVC tunneling evaluation.

The starting points of the test are four SVC encoding configurations (*Q1*, *Q2*, *Q3*, *Q4*) with highest layer QP of {16, 20, 24, 28} for fixed QP and target bitrate of {3, 2, 1.5, 1} Mbps for CBR. Table 11 lists the corresponding configurations for CBR and fixed QP rate control modes.

Note that in order to simplify our test-bed setup we kept the encoding configurations for SVC (i.e., *Q1*, *Q2*, *Q3*, *Q4*) static across sequences and selected the encoding quality for the original MPEG-2 streams accordingly. We acknowledge that in real-life scenarios the initial MPEG-2 streams are fixed and the SVC encoding configurations have to be adjusted instead. However, the quality configuration for MPEG-2 did not show any notable fluctuations between sequences. As each test was performed with four different quality configurations anyway, this simplification had no effect on the evaluated quality loss and RD performance characteristics.

The qualities of the SVC layers (labeled *L3* for highest layer and *L2*, *L1*, *L0* for the lower layers respectively) of the *Hall_Monitor* sequence are exemplarily shown in Table 12. While the bSoft encoder yields good overall RD performance, the automatic distribution of transform coefficients allocates little quality to the lower layers (due to a uniform rate distribution among layers) compared to our configuration of the MainConcept encoder.

Table 13: Y-PSNR results for MPEG-2 with fixed QP for the *Hall_Monitor* sequence.

QP	Bitrate [kbps]	Y-PSNR [dB]		QP	Bitrate [kbps]	Y-PSNR [dB]
1	7842	48.67		17	222	32.69
2	3631	43.69		18	214	32.41
3	1900	41.23		19	207	32.13
4	1309	39.75		20	199	31.85
5	873	38.77		21	194	31.60
6	674	37.85		22	188	31.37
7	543	37.14		23	184	31.15
8	456	36.46		24	179	30.93
9	384	35.90		25	174	30.71
10	337	35.41		26	172	30.51
11	305	34.98		27	168	30.31
12	286	34.52		28	166	30.14
13	271	34.09		29	163	29.96
14	257	33.71		30	161	29.79
15	246	33.32		31	159	29.63
16	234	33.01		32	125	20.98

The test-bed setup for the selection of MPEG-2 and SVC QPs and the subsequent evaluation are illustrated in Figure 41. The upper part of the figure shows the selection process, the lower part depicts the evaluation (similar to Figure 37, with the exception of the BD-PSNR calculation).

The qualities of MPEG-2 streams of the *Hall_Monitor* sequence encoded at QPs from 1 to 32 are shown in Table 13. Based on the proposed approach that the highest SVC layer should have a Y-PSNR just above that of the MPEG-2 stream, the starting points for our SVC tunneling evaluations for the bSoft encoder are the MPEG-2 streams with QP=2 for *Q1* (44.74 dB ≥ 43.69 dB), QP=3 for *Q2*, QP=4 for *Q3*, and QP=6 for *Q4*. The MPEG-2 streams that form the starting points for the other test sequences (for both bSoft and MainConcept encoders) in fixed QP mode were selected accordingly. For the MainConcept encoder in CBR mode, the initial MPEG-2 sequences were encoded in CBR mode at the following 32 bitrates: 9,500 kbps, 9,000 kbps, 8,500 kbps, 8,000 kbps, 7,500 kbps, 7,000 kbps, 6,500 kbps, 6,000 kbps, 5,500 kbps, 5,000 kbps, 4,500 kbps, 4,000 kbps, 3,500 kbps, 3,000 kbps, 2,750 kbps, 2,500 kbps, 2,250 kbps, 2,000 kbps, 1,750 kbps, 1,500 kbps, 1,250 kbps, 1,000 kbps, 900 kbps, 800 kbps, 700 kbps, 600 kbps, 500 kbps, 400 kbps, 300 kbps, 200 kbps, 100 kbps, and 50 kbps. The bitrates were selected to cover the entire range of bitrates that the FFmpeg encoder was able to encode for the test

Table 14: Bjontegaard Delta for SVC tunneling, adopted from [5].

Sequence	bSoft (fixed QP)		MainConcept (fixed QP)		MainConcept (CBR)	
	BD-PSNR [dB]	***BD-bitrate [%]***	***BD-PSNR [dB]***	***BD-bitrate [%]***	***BD-PSNR [dB]***	***BD-bitrate [%]***
Foreman	-2.08	50.3	-2.03	53.7	-2.40	61.6
Container	-1.57	38.2	-1.99	51.0	-2.91	66.9
Hall_Monitor	-0.75	22.6	-1.40	54.1	-1.82	73.6
Stefan	-2.59	41.0	-2.09	32.1	-2.88	53.4
Average	**-1.74**	**38.04**	**-1.88**	**47.7**	**-2.50**	**63.9**

sequences, i.e., the FFmpeg encoder was not able to encode any of the test sequences at bitrates above 9,500 kbps or below 50 kbps.

4.3.2.2 Experimental Results and Discussion

The BD results for the two transcoding steps are shown in Table 14. The BD is measured between the initial and final MPEG-2 encodings. As mentioned before, we applied a flexible approach for the target quality of SVC encoding. This means that the bitrates of the SVC streams did not correspond to those of the initial and final MPEG-2 streams. Thus, the BD is applicable neither to the MPEG-2-to-SVC transcoding step nor the SVC-to-MPEG-2 transcoding step, but only to the result of the entire transcoding chain.

Sequences with lower spatial detail and lower amount of movement (such as *Hall_Monitor*, *Container*) typically show less quality degradation than those with higher amounts. The overall results showed lower quality impact for fixed QP mode (-1.74 dB for bSoft encoder, -1.88 dB for MainConcept encoder on average) than for CBR mode (-2.50 dB on average).

The SVC layers were transcoded to MPEG-2 streams, the PSNR of each stream was calculated and again compared to the set of initial MPEG-2 streams to select the closest matching qualities for comparing the SVC tunneling bandwidth requirements to those of MPEG-2 simulcast. This calculation yields the MPEG-2 streams needed to perform a simulcast with the same qualities as the corresponding SVC tunneling setup.

We acknowledge that the approach for generating the initial MPEG-2 streams at lower qualities may prove difficult in a real-life scenario where the content is only available as one pre-encoded MPEG-2 stream. However, in such a scenario, a transrating tool [170][171] can be used for obtaining lower bitrate versions of the content. Thus, the generated MPEG-2 streams mark an upper bound for the quality achievable at the server side, and therefore also the best possible RD performance for MPEG-2 simulcast.

Table 15: Comparison of required bandwidths for SVC tunneling vs. MPEG-2 simulcast, adopted from [5].

Target Quality	bSoft (fixed QP)		MainConcept (fixed QP)		MainConcept (CBR)	
	SVC tunneling [kbps]	***MPEG-2 simulcast [kbps]***	***SVC tunneling [kbps]***	***MPEG-2 simulcast [kbps]***	***SVC tunneling [kbps]***	***MPEG-2 simulcast [kbps]***
Q1	5333	3041	3694	3454	3286	4721
Q2	3446	2025	2418	2082	2242	3191
Q3	2201	1452	1650	1277	1687	2093
Q4	1438	1102	1132	900	1109	1287
Average	**3105**	**1905**	**2224**	**1928**	**2081**	**2823**

Table 15 presents the comparison of average required bandwidths for SVC tunneling and MPEG-2 simulcast streaming. Columns labeled *SVC tunneling* show required bandwidths for delivering the content which has been transcoded from MPEG-2 to SVC (i.e., first transcoding step). For the second transcoding step, the content is transcoded back into the final MPEG-2 encoding. The required bandwidths for MPEG-2 simulcast (of the initial MPEG-2 encoding, cf. Table 13 for the *Hall_Monitor* sequence) with the same quality (in terms of Y-PSNR) as that final MPEG-2 encoding are shown in columns labeled *MPEG-2 simulcast*.

For the tested configurations, only CBR mode yields lower overall bandwidth requirements for full SVC tunneling than for equivalent MPEG-2 simulcast, reducing the required bandwidth by up to 32% (and 26% on average). SVC tunneling with fixed QP mode performs worse than equivalent MPEG-2 simulcast, even though it yields less quality degradation. This is attributed to the comparatively high quality of lower SVC layers in CBR mode (cf. Table 12), which manifests in higher bitrates of MPEG-2 simulcast in order to match that quality. We argue that the bandwidth efficiency of SVC tunneling depends more on the configuration of lower SVC layers than on the encoder implementation. For example, the MainConcept encoder in fixed QP mode was configured with a deltaQP of 6, while Section 3.4.5 suggests a deltaQP of 2 for SVC encoding, which would lead to higher qualities of the lower SVC layers. Furthermore, the number of SVC enhancement layers plays an important role for the bandwidth efficiency of SVC tunneling. Note that SVC tunneling with fixed QP mode may still be favorable over MPEG-2 simulcast in scenarios where only one of the two transcoding steps is needed (e.g., if the client's media player supports SVC), since every transcoding step has an impact on video quality.

Yang et al. [156] have proposed a logarithmic model for mapping MPEG-2 QPs to AVC QPs. A simplified version of the model is shown in Equation (2).

$$QP_{AVC} = a \cdot \log_2(QP_{MP2}) + b \qquad (2)$$

QP_{AVC} is the calculated QP for AVC (or SVC in our case), QP_{MP2} is the MPEG-2 QP, a and b are model parameters. Based on our averaged test results, the model parameters are $a = 7.1$ and $b = 4.8$ for the MainConcept encoder in fixed QP mode, and respectively $a = 7.1$ and $b = 8.9$ for the bSoft encoder. Note that this is only a rough estimate and that the mapping is content-dependent.

From the initial approach of applying the same bitrates for both transcoding steps, we have improved the configuration to select the quality of the MPEG-2-to-SVC transcoding based on the RD performance correlation between MPEG-2 and SVC. With this configuration, the coding performance characteristics of different codecs are taken into account. We have evaluated SVC tunneling with a focus on comparing the impact of fixed QP and CBR encoding modes with respect to quality degradation and bandwidth efficiency. The results indicate smaller quality impact for fixed QP mode (-1.74 dB and -1.88 dB, depending on the encoder) than for CBR (-2.50 dB), but the comparison of required bandwidth only yields a reduction for SVC tunneling with CBR mode (26%). In the following section, we will detail our studies of RD characteristics of the transcoding process in order to determine the trade-off between quality loss and bandwidth efficiency.

4.3.3 Advanced Configuration Options for SVC Tunneling

4.3.3.1 Test-Bed Setup and Configuration Improvements

The configurations for the quality evaluations of SVC tunneling discussed in Section 4.3.2 can be further improved. So far, the rate control for transcoding in fixed QP mode was configured as follows. For finding a suitable QP for SVC encoding, the original sequence was encoded at various QPs and we selected the QP yielding a quality just above that of the MPEG-2 encoded stream. With this QP, the reconstructed sequence (from MPEG-2) was encoded. (To be precise, we kept the SVC QP static and selected the MPEG-2 QP accordingly in order to simplify our test-bed setup.) For transcoding back to MPEG-2, the initial MPEG-2 QP was chosen. Thus, we assumed that this SVC QP would yield a decent quality at a moderate bitrate. While this is a reasonable assumption for estimating parameters for the SVC tunneling setup, we will investigate the effect of the SVC QP in this section. Furthermore, we will evaluate the effect of the MPEG-2 QP in the SVC-to-MPEG-2 transcoding step. Choosing the same MPEG-2 QP as the initial one is a straightforward solution. But if we assume no bandwidth constraints in the home network between the Home-Box and the end-user terminal, the MPEG-2 QP could be reduced, yielding a higher quality. As an extreme case, the MPEG-2 QP could be set to 1, causing almost no quantization.

With the same test-bed setup as in Section 4.3.2.1 (i.e., MPEG-2 encoding with the FFmpeg encoder, SVC encoding of 4 quality layers in fixed QP mode with the bSoft

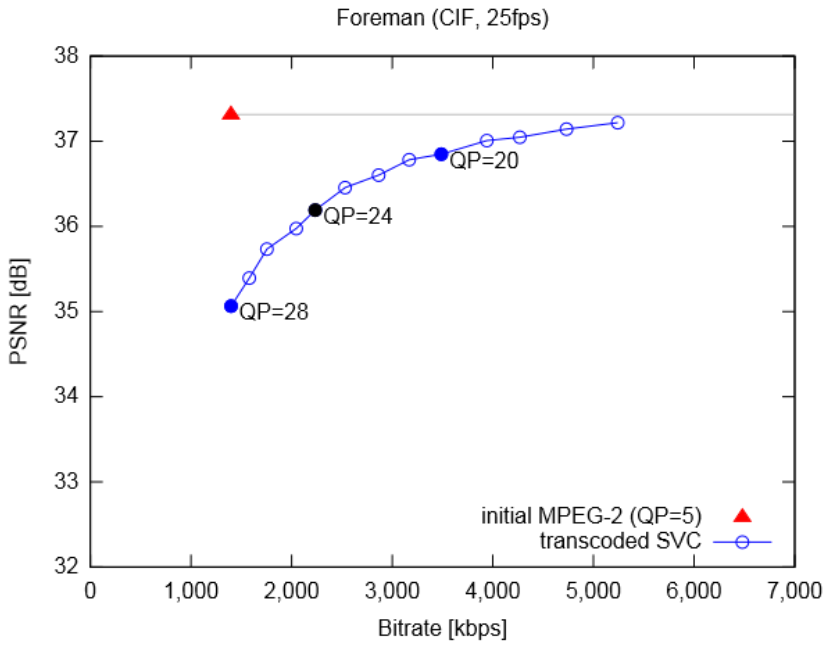


Figure 42: RD results for transcoding MPEG-2 to SVC at various QPs for the *Foreman* sequence.

encoder, test sequences *Foreman*, *Container*, *Hall_Monitor*, and *Stefan*, each at CIF resolution and 25 fps frame rate), the impact of different SVC QPs was evaluated.

4.3.3.2 Experimental Results

Figure 42 shows the transcoding RD results for the *Foreman* sequence. The sequence was first encoded to MPEG-2 with QP=5 (cf. Section 4.3.2.1), then it was transcoded via pixel-domain transcoding to SVC with QPs ranging from 16 to 28. The SVC QP of 24 is highlighted as it is the suggested QP from Section 4.3.2. SVC QP=24 yields a quality loss of 1.12 dB, for QP=28 the quality loss doubles to 2.25 dB. For lower QPs, the quality loss goes down to 0.46 dB for SVC QP=20 and even 0.09 dB for QP=16, but at the expense of very high bitrates (3.75 times the MPEG-2 bitrate for SVC QP=16). Note that the quality for SVC is constrained by the MPEG-2 stream. No matter how much bitrate is used, the quality of the SVC stream can never surpass that of the reconstructed MPEG-2 sequence it is based on.

The evaluation mostly confirms our initial approach for selecting the SVC QP. But it also indicates that within a certain QP range (from around 20 to 28), both quality and bitrates remain within reasonable bounds.

In the next step, we transcode the SVC streams back to MPEG-2 at various QPs. We selected the SVC streams with QP={20,24,28} as starting points. The highest layer of each stream was transcoded to MPEG-2 at QPs ranging from 1 to 8. The RD results for the *Foreman* sequence are shown in Figure 43. For each transcoded MPEG-2 stream, the initial suggestion from Section 4.3.2 (i.e., MPEG-2 QP=5) is highlighted. The initial MPEG-2 stream and the SVC streams are shown for reference. Note that

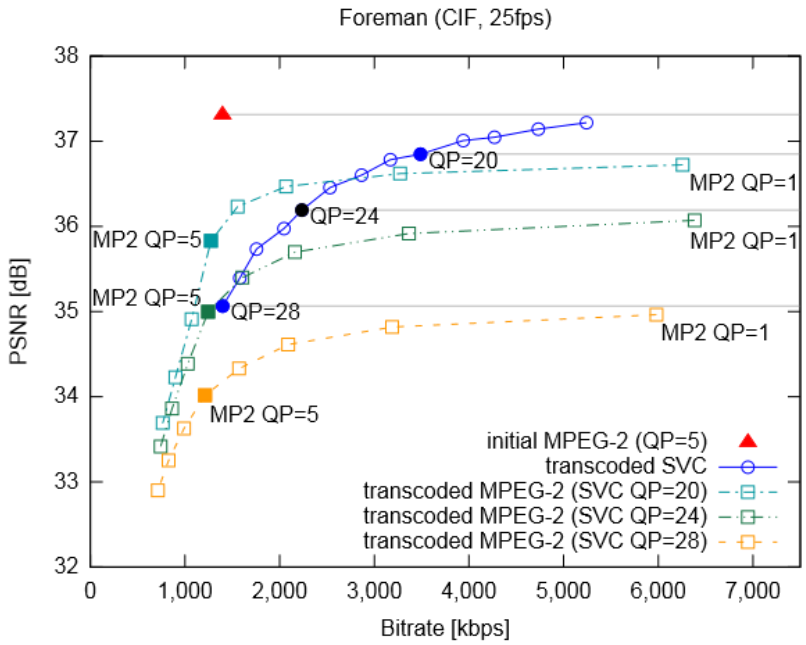


Figure 43: RD results for transcoding MPEG-2 to SVC and back to MPEG-2 at various QPs for the *Foreman* sequence.

QP=1 is the lowest possible quantization, explaining its high bitrates. Again, the quality of the transcoded MPEG-2 streams is constrained by the corresponding SVC stream. The Bjontegaard Delta is no longer applicable to this configuration as the bitrates of the transcoded streams do not correspond to the initial MPEG-2 streams.

From Figure 43 we conclude that the initial suggestion of MPEG-2 QP=5 is somewhat inefficient in terms of PSNR as it causes unnecessary quality loss (around 1.1 dB compared to SVC). Setting the MPEG-2 QP to 4 yields only 0.7 dB quality loss compared to SVC at a slight bitrate increase. Assuming an overprovisioned link between the transcoding Home-Box and the end-user terminal, the video can even be transcoded to MPEG-2 at a QP of 1, yielding mere 0.12 dB quality loss compared to SVC. The total quality loss (compared to the initial MPEG-2 stream) at transcoded MPEG-2 QP=1 is 1.24 dB for SVC QP=24 and as low as 0.59 dB for SVC QP=20, in contrast to the initial 2.32 dB for the suggestion from Section 4.3.2 (SVC QP=24, transcoded MPEG-2 QP=5).

In order to compare the performance of SVC streaming to MPEG-2 simulcast, appropriate QPs for the MPEG-2 streams transcoded from lower SVC layers have to be devised. The approach followed in Section 4.3.2 was to compare the quality of a lower SVC layer to the set of initial MPEG-2 streams encoded from the original sequence at various QPs and to select the MPEG-2 QP with the closest matching PSNR (i.e., where $|\text{MPEG-2 PSNR} - \text{SVC layer PSNR}|$ is minimal). In the case of the *Foreman* sequence at the aforementioned configuration, the MPEG-2 QPs for transcoding SVC layers are (from lowest to highest layer): 32, 29, 13, and 5.

If we again assume an overprovisioned home network, all lower SVC layers can be transcoded at MPEG-2 QP=1 as well despite the inevitably high bitrates. For

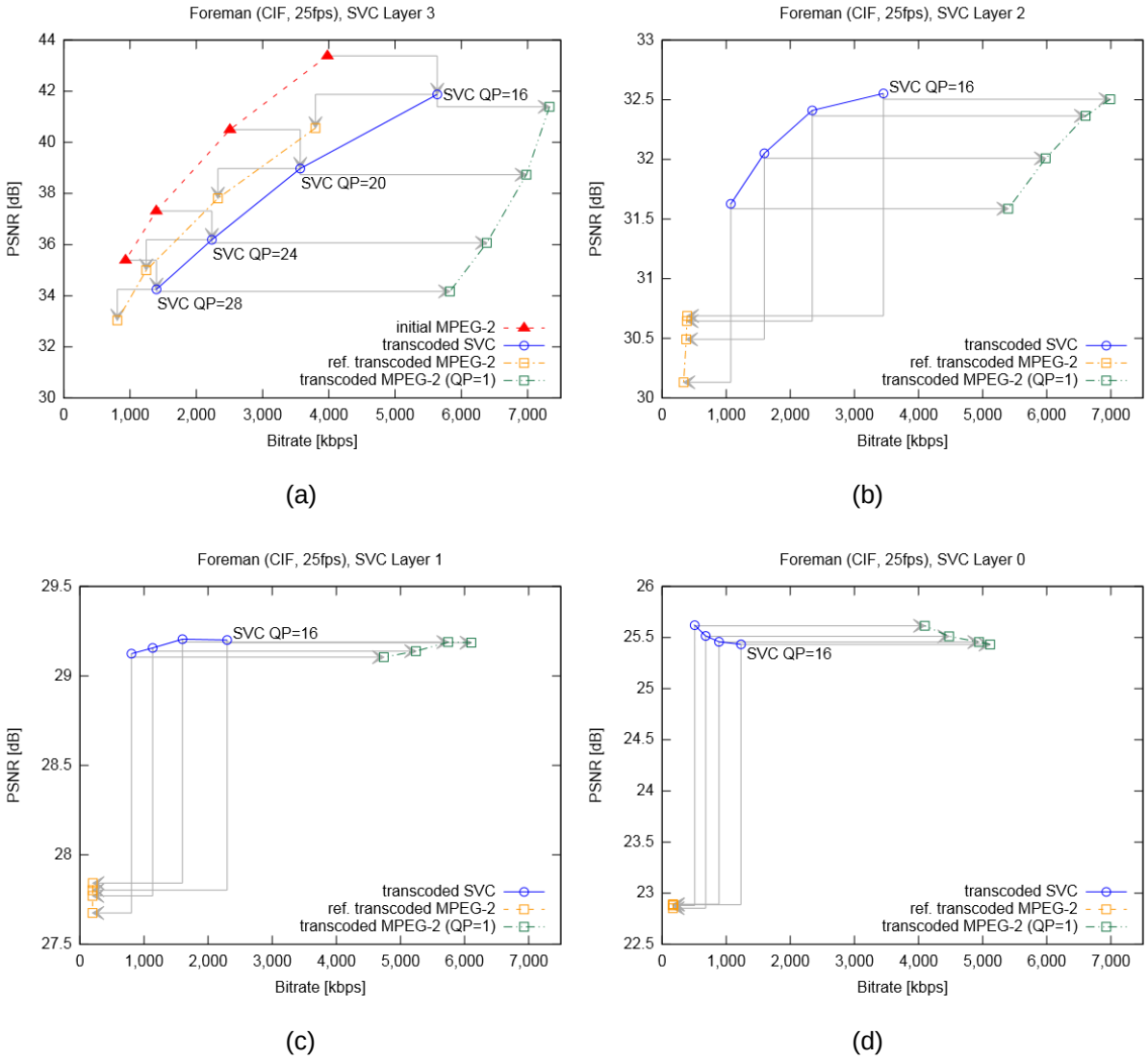


Figure 44: Rate-distortion performance for different QPs for SVC-to-MPEG-2 transcoding for the *Foreman* sequence at (a) SVC layer 3, (b) layer 2, (c) layer 1, and (d) layer 0.

example, at the previously suggested MPEG-2 QP of 32 for the base layer, the transcoded *Foreman* sequences has a PSNR of 22.85 dB at 171 kbps, whereas the same base layer transcoded at MPEG-2 QP=1 achieves a PSNR of 25.51 dB at 4476 kbps.

Figure 44 shows the impact of the QP for SVC-to-MPEG-2 transcoding on the RD performance of SVC layers for the *Foreman* sequence. For the highest layer, RD results of the initial MPEG-2 streams are included. Since the lower layers are generated by the SVC encoder, there are no corresponding initial MPEG-2 streams for these layers. Dotted arrows indicate the transcoding of individual data points. The previously suggested approach is labeled *ref. transcoded MPEG-2*.

Especially at lower layers, that approach causes significant quality losses, but also yields low bitrates. Since we assume an overprovisioned home network, transcoding with MPEG-2 QP=1 has only marginal quality losses. The rather strange RD performances of the SVC base layer in Figure 44 (d) are implementation dependent

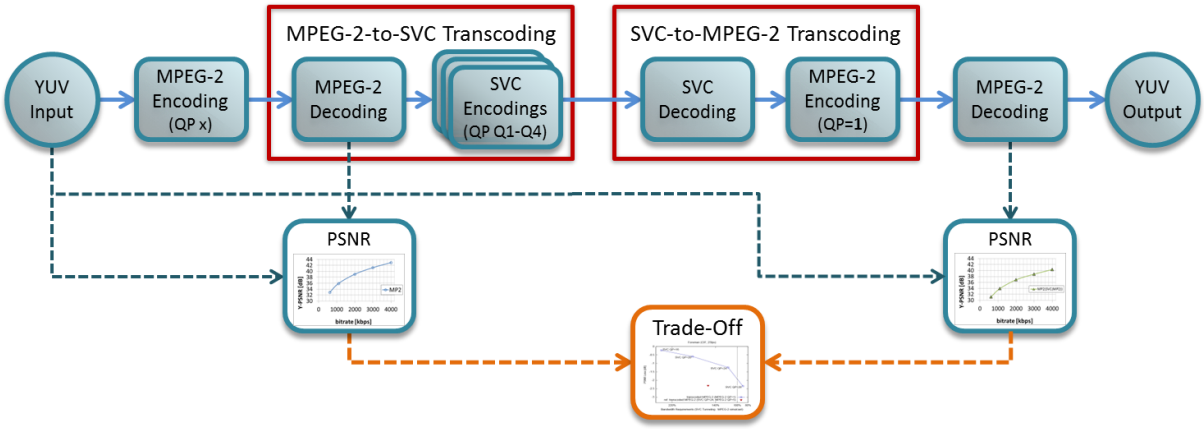


Figure 45: Test-bed setup for selection of QPs and evaluation of quality-versus-bandwidth trade-off.

(i.e., the bSoft encoder assigns more and more bitrate to the base layer to improve the quality of enhancement layer depending on it, but the base layer quality itself decreases). Thus, the results of the transcoded MPEG-2 streams at QP=1 follow the same characteristics.

The SVC QP for the initial transcoding step from MPEG-2 to SVC governs the bitrate trade-off between SVC tunneling and MPEG-2 simulcast. But it also controls the overall quality loss. Figure 45 illustrates the test-bed for evaluating the trade-off between quality loss and bandwidth savings.

With all SVC layers transcoded at MPEG-2 QP=1, Figure 46 illustrates the relation between the quality loss at the highest layer and the trade-off in bandwidth requirements for the *Foreman* sequence. The x-axis shows the bandwidth

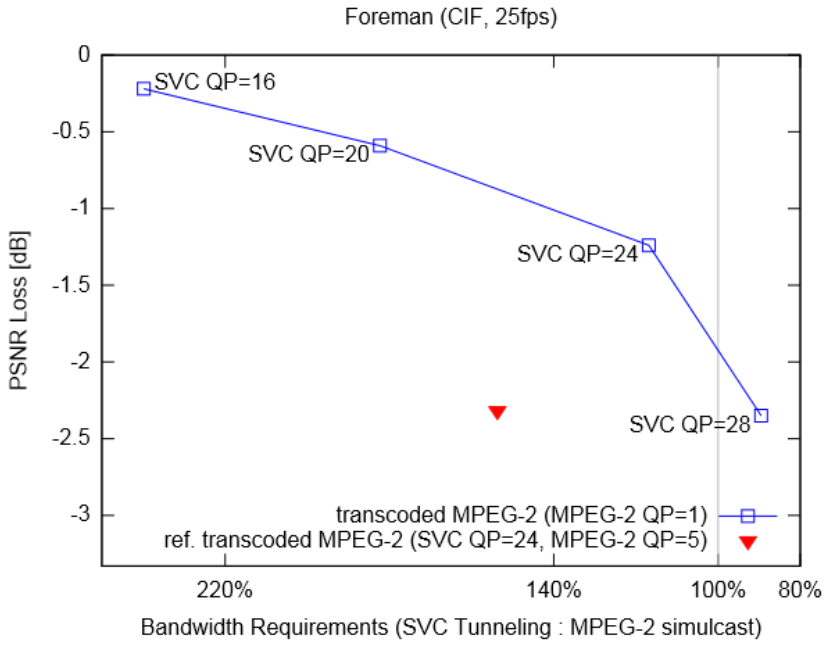


Figure 46: Trade-off between bandwidth requirements and quality loss of SVC tunneling for the *Foreman* sequence, adopted from [9].

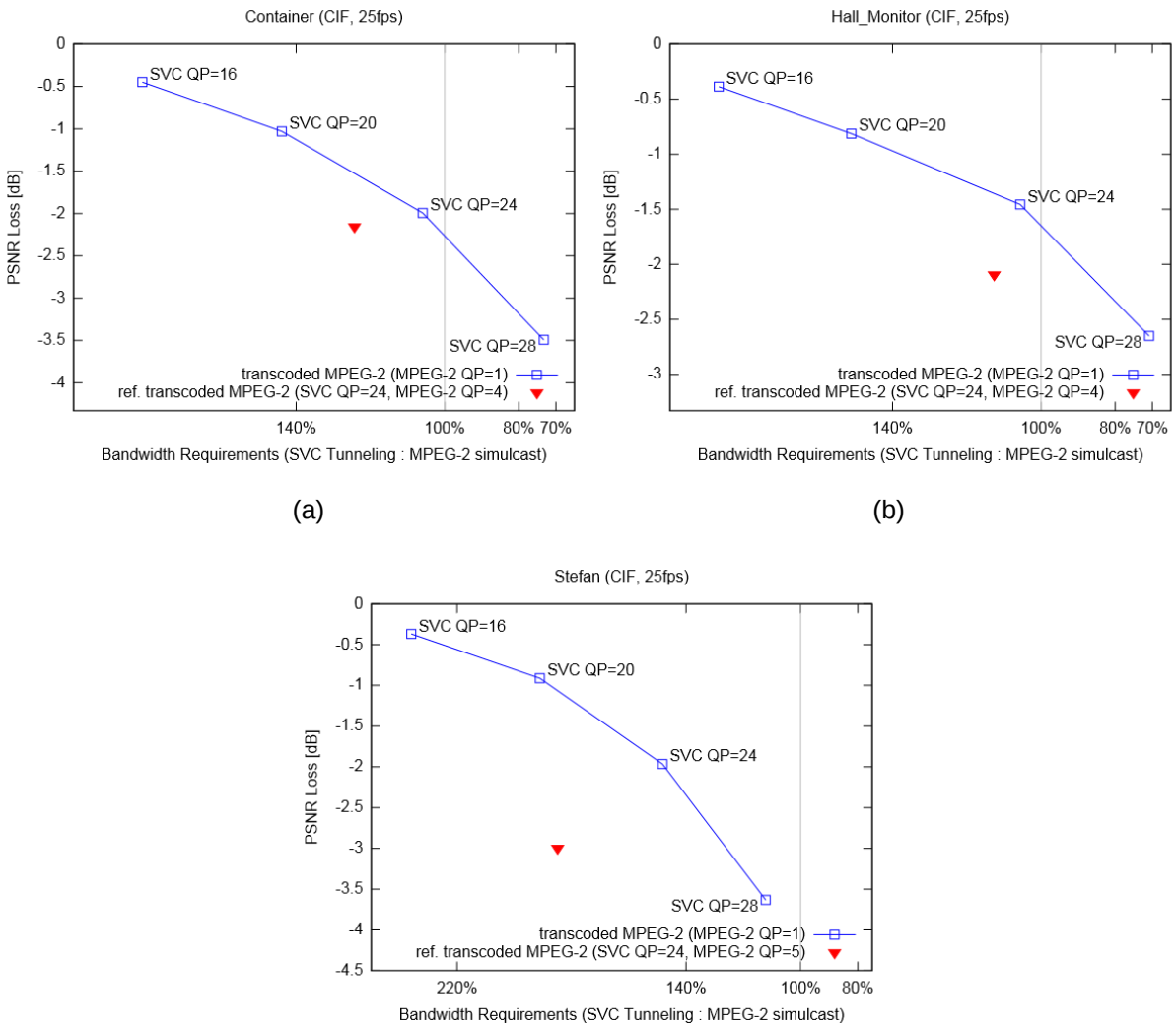


Figure 47: Trade-off between bandwidth requirements and quality loss of SVC tunneling for (a) *Container*, (b) *Hall_Monitor*, and (c) *Stefan* sequences, adopted from [9].

requirements of SVC tunneling compared to MPEG-2 simulcast at roughly the same quality, i.e., values below 100% mean that SVC tunneling is more efficient. The y-axis shows the quality loss at the highest layer compared to the initial MPEG-2 sequence. The bandwidth requirement and quality loss for the initial configuration from Section 4.3.2 is shown for reference. Note that while the quality loss is compared to the initial MPEG-2 sequence, the bandwidth comparison already uses the lower quality versions of the MPEG-2 streams. For example, at SVC QP=28, the quality drops by 2.35 dB, but the SVC tunneling only requires 89.5% of the bandwidth needed to stream MPEG-2 simulcast at that lower quality that is actually received by the end user.

It can be observed that in order for SVC tunneling to be more efficient, we have to take a quality loss of at least 2.0 dB for this sequence into account. While it is possible to achieve lower quality loss, SVC tunneling would require more bandwidth than MPEG-2 simulcast in such configurations. Due to the transcoded MPEG-2 QP of 1, the SVC QP can be increased from 24 to 28, at the same quality as the initial

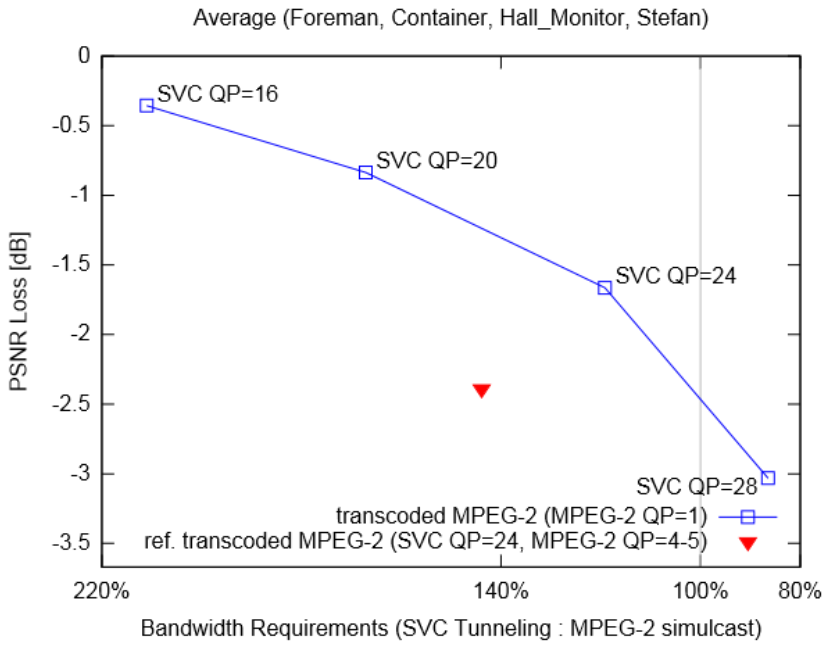


Figure 48: Average trade-off between bandwidth requirements and quality loss of SVC tunneling.

configuration (labeled *ref. transcoded MPEG-2 (SVC QP=24, MPEG-2 QP=5)*), but at far lower bandwidth requirements.

The same trade-off is shown in Figure 47 for test sequences *Container*, *Hall_Monitor*, and *Stefan* respectively. The *Container* and *Hall_Monitor* sequences have a similar performance as *Foreman*, whereas the *Stefan* sequence has stronger quality degradation and for SVC QP=28 it does not even pass the point where SVC tunneling would be more bandwidth efficient than MPEG-2 simulcast.

Figure 48 shows the trade-off averaged over all test sequences. The quality is reduced by around 2.5 dB on average when aiming for bandwidth efficient SVC tunneling using the bSoft encoder.

4.3.3.3 Partial SVC Tunneling Evaluation

As discussed in Section 4.2.2, two partial variants of SVC tunneling are possible, omitting either the first or the second transcoding step. One transcoding step less causes less quality loss and thus better bandwidth efficiency.

For both variants of partial SVC tunneling, Figure 49 shows the trade-off between bandwidth efficiency and PSNR loss for the *Foreman* sequence. The results for full SVC tunneling are shown for reference. For the case of only MPEG-2-to-SVC transcoding at the server side (i.e., assuming SVC support at the end-user terminal), the loss characteristics are similar to full SVC tunneling. But since the transcoding back to MPEG-2 is omitted, PSNR losses are lower. This also slightly increases the bandwidth efficiency.

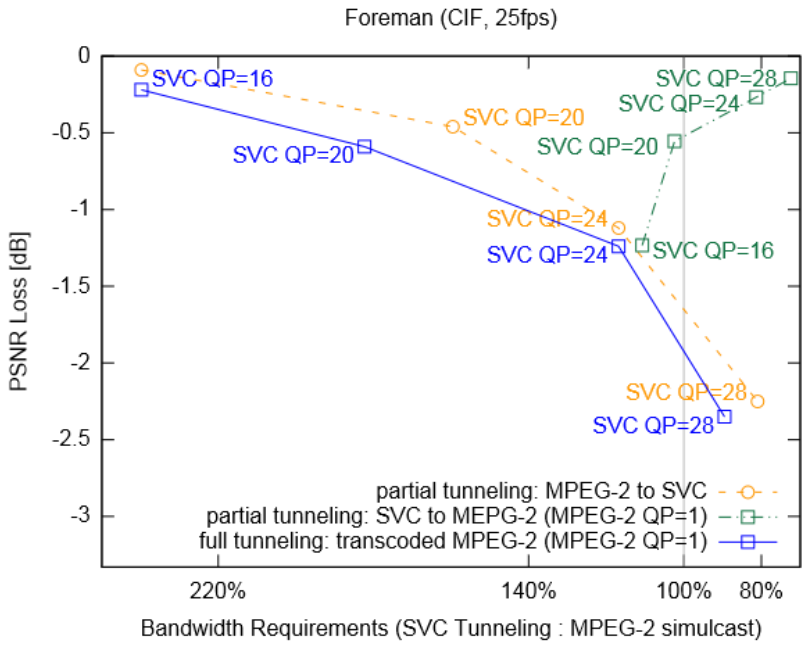


Figure 49: Trade-off between bandwidth requirements and quality loss of partial SVC tunneling for the *Foreman* sequence.

But if we assume the content to be available in SVC, thus only requiring transcoding to MPEG-2 at the client side (i.e., the line labeled *partial tunneling: SVC to MPEG-2 (MPEG-2 QP=1)*), we observe different characteristics. The configuration with SVC QP=28 has the lowest PSNR loss at 0.14 dB. This is due to the fact that for transcoding we always apply a MPEG-2 QP of 1. Stronger quantization in the SVC stream causes less quality loss in the transcoding step. Note that the figure only shows quality losses and not the absolute quality which of course decreases with stronger quantization.

4.3.3.4 JSVM-Based Evaluation

As evaluated in Chapter 3, the bSoft encoder tends to generate streams with quite high bitrates compared to the SVC reference software *JSVM*. Therefore, we also briefly evaluate the performance of SVC tunneling using the JSVM encoder. Using the same setup as for the bSoft encoder, the *Foreman* sequence was first encoded to MPEG-2 at QP=4 (chosen based on the described approach). The MPEG-2 stream was then transcoded to SVC via the JSVM at QPs ranging from 16 to 28 (with a deltaQP of 2). Following the assumption of an overprovisioned home network, all SVC layers were transcoded back to MPEG-2 at QP=1. As a reference for our initial approach, the SVC stream with QP=24 was also transcoded to MPEG-2 with QPs based on the best matching MPEG-2 quality.

The trade-off between the bandwidth requirements of SVC tunneling compared to MPEG-2 simulcast of roughly the same qualities and the overall PSNR loss at the highest layer are shown in Figure 50. It can be observed that SVC tunneling is far more efficient than MPEG-2 simulcast when using the JSVM. At SVC QP=16, the

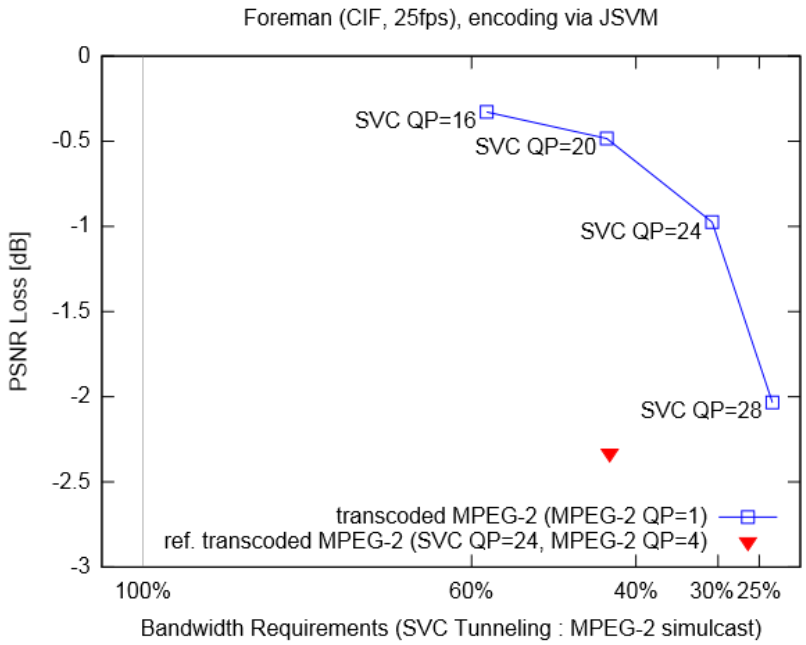


Figure 50: Trade-off between bandwidth requirements and quality loss of SVC tunneling for the *Foreman* sequence using the JSVM encoder.

transcoded MPEG-2 stream loses merely 0.33 dB compared to the initial MPEG-2 stream while requiring only 58.2% of the bandwidth a comparable MPEG-2 simulcast would need. The bandwidth requirements can be further reduced to 23.4% of MPEG-2 simulcast if a PSNR loss of 2.03 dB compared to the source material is taken into account.

The figure also reveals that the usage of the initial MPEG-2 QP (set to 4 in our case) causes significant quality degradation in the final transcoding step, resulting in a PSNR loss of 2.33 dB, compared to 0.97 dB yielded by an MPEG-2 QP of 1.

But the higher bandwidth efficiency comes at the cost of encoding speed. This makes the JSVM unsuitable for scenarios with on-the-fly transcoding on the server side. As evaluated in Section 3.4.6, the bSoft encoder is one order of magnitude faster than the JSVM. While the content can usually be transcoded to SVC prior to streaming, current implementations of SVC encoders are often only fully compatible with the decoder of the same implementation as noted in Section 3.3.2. Thus, deploying the JSVM decoder at the Home-Box for SVC-to-MPEG-2 transcoding also limits the transcoding speed. For scenarios with real-time constraints, we suggest the deployment of an industrial encoder in SVC tunneling solutions.

We conclude that the efficiency of SVC tunneling depends on the implementation and RD performance of the SVC encoder. With the bSoft encoder, which had the highest bitrates in our encoding evaluations in Chapter 3, SVC tunneling is more efficient than MPEG-2 simulcast at a tolerable quality loss (~2.5 dB). With the use of the JSVM encoder, SVC tunneling becomes even more bandwidth efficient, allowing for a higher degree of control between reduction of bandwidth requirements and the confinement of quality loss.

4.4 Conclusions

In this chapter, we have presented and evaluated the concept of *SVC tunneling* for multicast content delivery. The proposed architecture may require video transcoding from and to non-scalable legacy video formats, such as MPEG-2, at the ingress and egress points of the network. The goals of SVC tunneling for content delivery are the reduction of network load through cumulative layered multicast and the provisioning of QoS management in content-aware networks. In the ALICANTE architecture, SVC tunneling is deployed to enable device-independent media access while allowing dynamic in-network adaptation. The ALICANTE project explores the exploitation of content-aware networking for SVC tunneling, ranging from in-network adaptation to intelligent routing mechanisms. In several steps, we have evaluated the trade-off between quality loss due to transcoding and the bandwidth efficiency of the proposed approach.

The presented research focuses on MPEG-2 as the source and target formats in order to support legacy devices. This choice provides a baseline for other formats. We applied a pixel-domain transcoding approach with full decoding and re-encoding, for which we measured the corresponding quality degradations. For AVC as source and target formats, lossless bitstream rewriting can be applied, ideally avoiding quality losses at all. However, several restrictions, in particular with respect to SVC scalability options, would apply for AVC-to-SVC rewriting.

Throughout our evaluation, we have investigated various parameters that influence the efficiency of SVC tunneling. We have tested different SVC encoding configurations, SVC encoders, rate control modes, and the target qualities (QP, target bitrate) for transcoding. We have developed guidelines for controlling the transcoding-induced quality loss and shown the trade-off characteristics between the total quality loss and the bandwidth efficiency compared to MPEG-2 simulcast of the same (degraded) quality.

The following list summarizes the research contributions and key findings of this chapter:

- The performance of SVC tunneling in terms of quality impact, bandwidth efficiency, and transcoding speed strongly depends on the encoder implementation.
    - Proprietary encoders/decoder, such as *bSoft* and *MainConcept*, provide reasonable transcoding speed.
    - Since the *JSVM* reference software yields better RD performances, it enables lower quality loss and better bandwidth savings.
- The trade-off between quality loss and bandwidth efficiency is computed as follows. First, the quality loss of the video in the client's target coding format (i.e., transcoded from MPEG-2 to SVC and back to MPEG-2) in comparison to the (MPEG-2-encoded) source content is calculated. Then, the source content is encoded to MPEG-2 at various bitrates to match each of the qualities of the

extraction points of the SVC bitstream. These MPEG-2 encodings form the MPEG-2 simulcast set used as reference. Finally, the sum of the bitrates of this reference is compared to the bitrate of the SVC bitstream from SVC tunneling. Thus, the trade-off between the overall quality loss and the bandwidth savings over MPEG-2 simulcast of the same quality as the SVC bitstream is obtained.

- We have evaluated the quality loss of transcoding to and from SVC for MPEG-2 as the source and target formats. These findings enable advanced control of the quality impact of SVC tunneling.
    - The naïve approach to use the same target bitrates for MPEG-2 encoding, MPEG-2-to-SVC transcoding and SVC-to-MPEG-2 transcoding lacks flexibility and does not take coding efficiency characteristics of the coding formats into account.
    - A selection mechanism of target qualities for transcoding (i.e., target bitrates or QPs) based on a comparison of PSNR video qualities between MPEG-2 and SVC allows for better adjustment to coding format characteristics. However, it still lacks the flexibility to control the trade-off between quality loss and bandwidth efficiency.
    - Assuming an overprovisioned home-network, as it is the case in the ALICANTE architecture between the Home-Box and the end-user terminal, the SVC-to-MPEG-2 transcoding step shall use as much bitrate as possible in order to reduce quality loss. In our evaluations, the quality loss of that transcoding step was reduced to 0.14 dB for the bSoft encoder.
    - We found that the most efficient mechanism for evaluating the trade-off between quality loss and bandwidth efficiency is the MPEG-2-to-SVC transcoding at various target qualities (i.e., SVC QPs). According to our evaluations, around 2.5 dB PSNR loss has to be taken into account for *full SVC tunneling* with the bSoft encoder in order to achieve bandwidth savings w.r.t MPEG-2 simulcast.
    - With the JSVM reference software, the quality loss for *full SVC tunneling* can be constrained to 0.33 dB while requiring 41.8% less bandwidth than a comparable MPEG-2 simulcast.

We have also evaluated scenarios in which only a partial deployment of SVC tunneling is needed, e.g., if no MPEG-2-to-SVC transcoding is required. Such scenarios obviously induce less quality degradation. Our results highlight the different trade-off characteristics with respect to quality loss and bandwidth requirements.

Within the ALICANTE architecture, SVC is deployed not only for multicast streaming, but also for RTP unicast streaming, P2P streaming, and DASH as discussed in the following chapter. The investigated quality impact of transcoding applies for those transport modes as well.

The following chapter also comprises an evaluation of SVC tunneling in an integrated streaming test-bed with in-network adaptation.

Future work should target the evaluation of SVC tunneling with different source and target formats. While SVC tunneling with AVC can theoretically be achieved without quality losses thanks to bitstream rewriting, the quality impact for commercially deployed lossy transcoders would be an interesting topic. In our evaluations, we have relied on a traditional GOP structure. In order to reduce the transcoding delay, the GOP structure could be optimized for low-delay encoding [27]. The impact of low-delay GOP structures on the coding efficiency and, subsequently, on the quality loss remains to be evaluated. Furthermore, MPEG and the ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC) are currently developing the successor of SVC, which will be based on the HEVC technology [57]. With this Scalable High-efficiency Video Coding (SHVC), further improvements of RD performance can be expected, which would also improve the bandwidth efficiency of our proposed scalable media coding tunneling approach.

5 Distributed Adaptation and Media Transport

5.1 Introduction

With the increasing popularity of multimedia services, it is essential that media streaming systems provide high QoE to the end users while using network resources optimally. We argue that media streaming for the FI requires the consideration of the entire media delivery chain, which includes adaptation at various locations along that chain. So far, we have discussed encoding considerations for SVC in Chapter 3 as well as SVC tunneling enabling format-independent transport of scalable media coding and media access for heterogeneous devices in Chapter 4.

In this chapter, we will investigate various aspects of SVC-based media transport and the adaptation associated with it. Our goal is to study how entities within or at the edges of the network can adapt content in order to provide the best QoS/QoE. The contributions of this chapter mainly comprise implementation and deployment aspects within the context of the ALICANTE project, which serve as a validation of the work conducted in the previous chapters. First, we will discuss the prospects and challenges of deploying scalable media coding in Content-Aware Networks for several use cases in Section 5.2. The section provides a generic view on *scalable media coding* and relevant architectural aspects towards adaptation (or flow processing in general), caching/buffering, and overall Quality of Service/Experience management for streaming via RTP, P2P, or HTTP. Note that *scalable media coding* refers to a general concept that can be applied to video coding as well as audio coding (cf. Section 2.2), wheras SVC denotes the H.264/AVC extension. While some of the techniques covered in this chapter are also applicable to audio coding, we base our discussions mainly on video coding. Section 5.3 will then describe the distributed adaptation architecture deployed in ALICANTE. Adaptation techniques for SVC will be detailed in Section 5.4. In addition to a review of related work on adaptation strategies and the description of the adaptation logic implemented within ALICANTE, we will also propose a new approach towards reducing the effects of adaptation on the viewing experience by performing smooth transitions between representations. The end-to-end adaptation approach from Section 5.3 will be validated in Section 5.5. The chapter is concluded in Section 5.6, where we also answer the research challenges towards a distributed adaptation decision-taking framework that were identified earlier in Section 2.3.3. Those research challenges address questions as to *where*, *when*, *how often*, and *how* to adapt. The section will also provide an outlook on future work in this area.

The work presented in this chapter is published in [10], [11], [5], and [14].

5.2 Scalable Media Coding Enabling Content-Aware Networking

The FI development [172] has raised a rich set of research issues given the huge, global impact of this technology and new societal needs for media services. The term FI encompasses a broad range of activities to improve the architecture of the current Internet – an Internet that works on technologies designed decades ago when no one could have foreseen the way the Internet is used today or tomorrow. While the current Internet architecture is characterized by many ad-hoc solutions and technologies that were designed for purposes different from their actual deployment, future developments have to address long-term goals toward the Internet's full potential [173]. A significant trend is recognized towards an information-centric orientation and, consequently, new challenges are emerging. In particular, significant changes in communications and networking have been proposed, including novel basic architectural principles. What are the implications of new networking principles for media streaming? How does the deployment of scalable media formats benefit from these developments? Before we answer these questions, let us briefly revisit the approaches towards the FI and the basics of scalable media formats. The new conceptions are generally divided into revolutionary (i.e., clean-slate) and evolutionary approaches. The revolutionary approaches are often referred to as *Information-Centric Networking (ICN)*, which is used as an umbrella term for related concepts such as Content-Oriented Networking (CON) and Content-Centric Networking (CCN) [174][175]. On the other hand, evolutionary (or incremental) approaches, such as *Content-Aware Networking*, aim at building upon existing Internet infrastructures. In this section, we will explain the role of *CAN* for multimedia services in more detail. We will present four media streaming use cases which characterize different requirements w.r.t. content-aware processing in the network and highlight the utility of scalable media formats.

Clean-slate ICN approaches, as surveyed in [172] and [175], are very promising, but they raise a long list of research challenges like the degree of preservation of the classic transport (TCP/IP) layering principles, naming and addressing, content-based routing and forwarding, management and control framework, in-network caching, energy efficiency, trust, security embedded in the content objects, Quality of Service and Experience, and media flow adaptation. Additionally, new business models are needed for users, content producers, consumers, and service/network providers; deployment issues such as compatibility with existing equipment, scalability, and privacy become crucial.

In parallel, evolutionary approaches towards the FI such as Content-Aware Networking are being proposed in [176] and developed within the ALICANTE project, enabling efficient *routing and forwarding* of content based on given *content and context characteristics* and also to enable content *adaptation*. ALICANTE deploys content- and context-aware strategies at the network edges as discussed in [7]. A

main challenge of evolutionary approaches is obviously overcoming the limitations of the current Internet [172].

The ALICANTE content-aware network environment attempts to optimize network resource utilization while maintaining the expected QoS and QoE, respectively. For this purpose:

- It establishes virtual networks on top of the physical infrastructure that feature inherent content awareness, e.g., by dynamically providing network resources appropriate for different content types.
- It provides in-network media caching as well as real-time adaptation, exploiting scalable media coding formats, such as SVC, which are a vital component towards this objective thanks to their compression efficiency and flexibility [7].

Both functions are provided by enhanced network nodes, the MANEs, which feature virtualization support, content-awareness, and media processing, as well as buffering and caching.

The aim of this section is to describe the role of scalable media coding formats – such as SVC – in Content-Aware Networks and to propose new solutions for some use cases. Therefore, we will describe a set of use cases (Section 5.2.1) and provide an analysis thereof regarding a selection of CAN challenges (Section 5.2.2), specifically flow processing, caching/buffering, and QoS/QoE management.

5.2.1 Use Cases

In this section we will illustrate use cases highlighting the benefits of using SVC in CAN ranging from *unicast* and *multicast* to *P2P* and adaptive *HTTP streaming*.

A simplified and generic high-level system overview for the use cases in question is depicted in Figure 51 comprising the following entities: two senders (*S1*, *S2*), two MANEs (*MANE1*, *MANE2*), and three receivers (*R1*, *R2*, *R3*) with different terminal and (potentially) network capabilities, to which three end users (*U1*, *U2*, *U3*) are connected. Our discussion of the use cases addresses streaming of non-live content (e.g., Video on Demand), unless noted otherwise. Please note that in more complex scenarios, more senders, even more receivers, and additional MANEs distributed over multiple autonomous network domains may be deployed. These use cases are subsequently analyzed in Section 5.2.2 with respect to content-aware networking aspects.

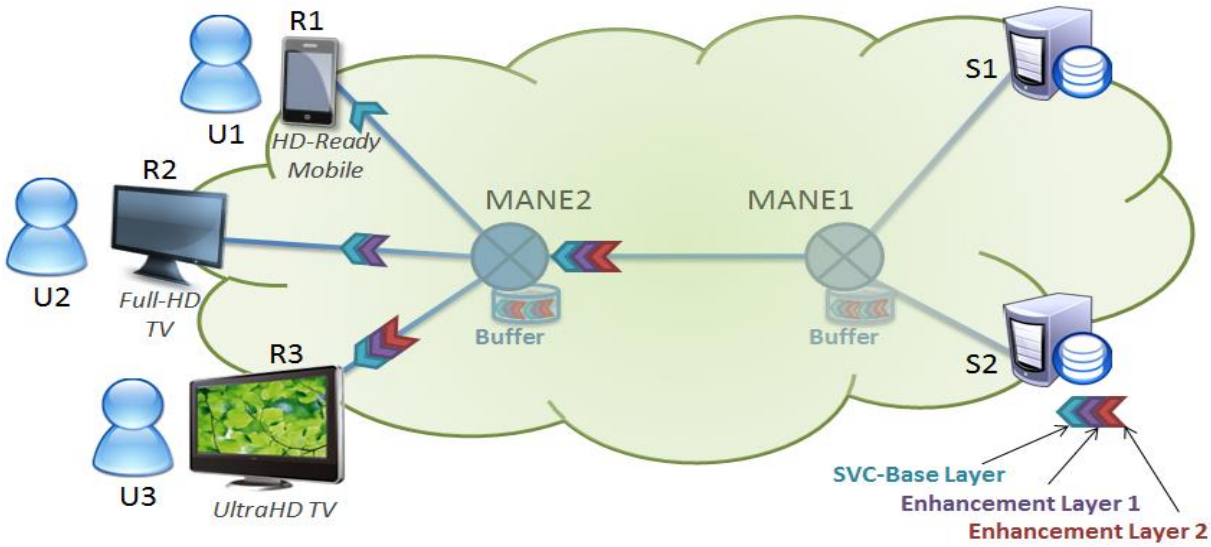


Figure 51: High-level system overview, adopted from [10].

5.2.1.1 Unicast Streaming

For the unicast use case we have only one sender (e.g., *S1*) that streams the scalable video content to a single receiver (e.g., *R3*), as in a traditional Video on Demand application (see Figure 52). This layered media coding approach enables MANEs along the path to perform content-aware operations such as in-network content adaptation. For example, a MANE can react to changing network conditions (based on information provided by a network monitoring system) by dropping enhancement layers of the SVC stream. In current deployments, RTP is typically used as the transport protocol and the Real Time Streaming Protocol (RTSP) [177] is used for session control. Note that in the unicast use case the SVC stream is typically sent via single-session transmission mode over RTP, i.e., all SVC layers are packed into one RTP session.

5.2.1.2 Multicast Streaming

The second use case is multicast streaming, which is characterized by a single sender providing the same content to multiple receivers. In this case, one sender (e.g., *S2* in Figure 51) is streaming the content to heterogeneous trees of MANEs and subsequently to multiple receivers (e.g., *R1*, *R2*, *R3*). The term *heterogeneous trees* denotes a set of trees, allocated for different SVC layers. All trees have the same root (e.g., *S2*) but different leaves, depending on the transported SVC layer (e.g., the SVC base layer is delivered to all receivers, while the highest SVC layer is only received by *R3*), as shown in Figure 53.

Scalable media formats enable the realization of this use case via receiver-driven layered multicast (RDLM) [48] and with SVC this approach is becoming efficient enough to surpass simulcast [7]. In RDLM, different layers are transmitted over

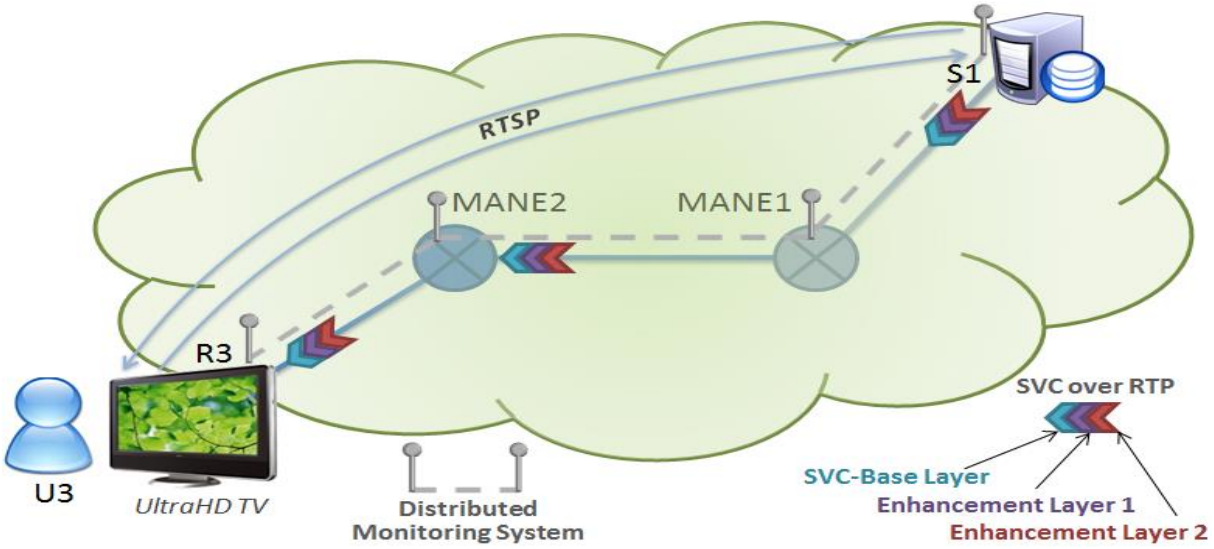


Figure 52: Unicast streaming in Content-Aware Networks, adopted from [10].

separate multicast groups. RTP realizes this via the multi-session transmission mode, where SVC layers are separated into multiple RTP sessions at the sender side, and rearranged to the proper SVC bitstream at the receiver side. Each receiver subscribes only to those layers that it supports and that its network link can handle.

Again, a MANE can react to changing network conditions by adjusting the number of layers to which it is subscribed. Such an approach simplifies the adaptation operations. MANEs can transparently neglect the video header information, since the mapping of SVC layers to multicast groups is realized at a lower level, simplifying the process of content adaptation. In other words, a MANE simply adjusts the number of subscribed RTP sessions without having to inspect each and every RTP packet header.

5.2.1.3 Peer-to-Peer Streaming

In a P2P streaming use case, multiple senders exist and every sender provides some parts of the content called chunks or pieces, while one or possibly more receivers consume the content. A scalable media format enables each receiver to request only the layers that are supported by its media player [178].

In contrast to conventional P2P content distribution, P2P streaming has a timing constraint that every piece must arrive before its playout deadline expires. P2P streaming systems typically use a sliding window of pieces which are currently relevant for the receivers. Within this sliding window, a piece-picking algorithm at the receiver side takes care of downloading those pieces that provide the highest quality to the end user. The piece-picking algorithm ensures that the base layer is always received before the deadline, determines enhancement layers that can be downloaded under the current network conditions, and takes care of the peer selection for each piece [179].

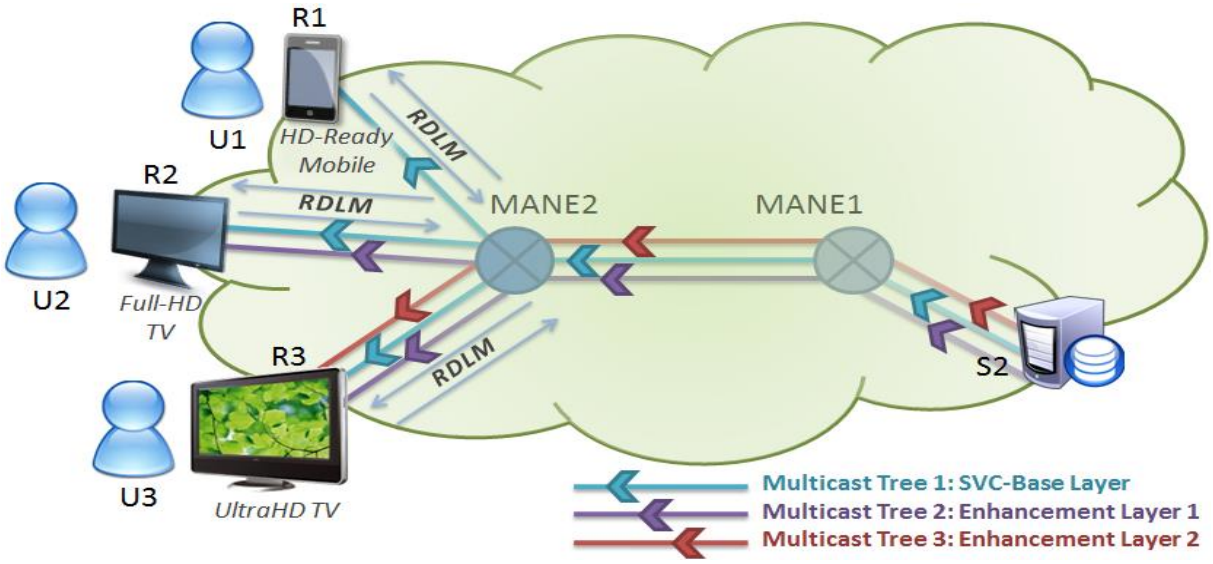


Figure 53: Multicast streaming in Content-Aware Networks, adopted from [10].

While a P2P system is traditionally organized as an overlay network that is transparent to the core network, a content-aware network will allow MANEs to participate in the streaming process in several ways. Figure 54 shows an outline of this use case, showing senders, receivers, and the supporting MANEs.

A MANE can participate in P2P streaming by caching pieces in a content-aware manner or by acting as a peer itself as discussed later in Section 5.2.2.1).

5.2.1.4 Adaptive HTTP Streaming

The previous use cases have shown streaming scenarios with various numbers of senders and receivers. In order to overcome common shortcomings of RTP-based streaming such as network address translation (NAT) and firewall issues, this use case introduces adaptive HTTP streaming (e.g., DASH) in the context of CAN. In HTTP streaming, the content is typically fragmented into segments that are downloaded by the receiver via individual HTTP (partial) GET requests. This approach allows for a stateless sender and enables at the same time caching at the MANEs and dynamic content adaptation at the client. Based on several industry solutions, MPEG has recently standardized DASH [50][180]. For the sake of generality, this discussion uses the term *Adaptive HTTP Streaming* instead of DASH.

HTTP streaming is typically used in unicast mode, but multicast or even P2P streaming modes are also possible.

In unicast mode, the sender provides a manifest file of the content that describes the structure of the media segments and the available media representations. A media representation denotes a particular encoding configuration of the content, e.g., bitrate or resolution [180]. For layered coding formats such as SVC, those representations can define either the individual layers or even subsets of bitstream layers. The receiver selects the appropriate representation based on its processing and

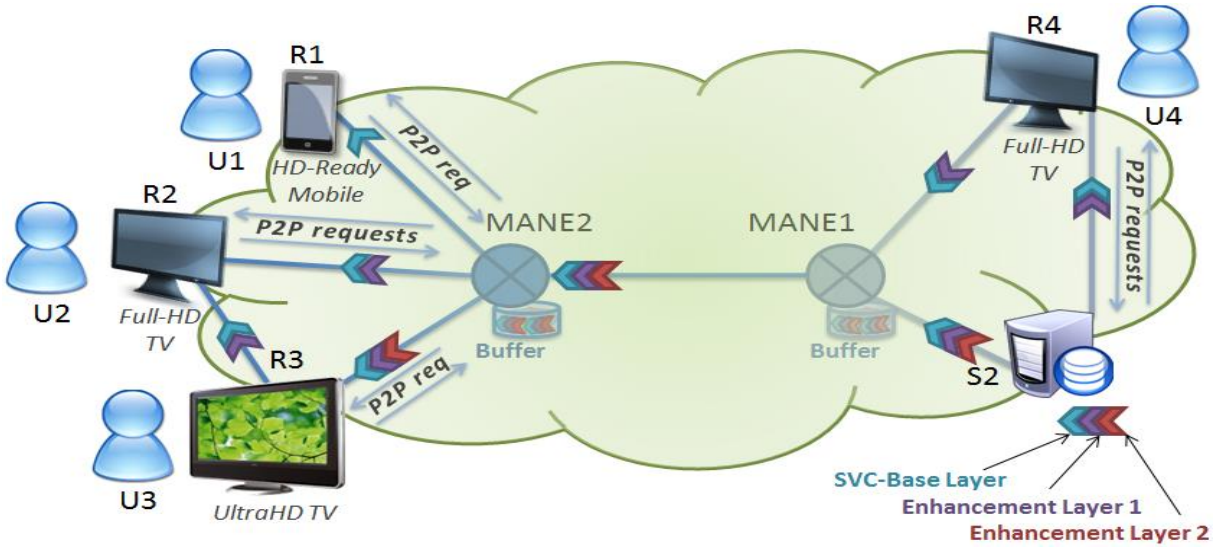


Figure 54: P2P streaming in Content-Aware Networks, adopted from [10].

rendering capabilities and starts requesting continuous segments of the content from the sender. MANEs along the network path can act as caches or as content delivery network (CDN) nodes, as shown in Figure 55.

Although HTTP is a unicast protocol, the concept of HTTP streaming can also be applied to multicast streaming. If MANEs along the network path between the sender and receivers cache the content segments for subsequent requests by other receivers, the result will be a multicast-like tree. The technical considerations of this approach are discussed later in Section 5.2.2.2.

The concept of HTTP streaming can even be applied to multisource streaming scenarios similar to P2P streaming. The manifest file can contain multiple sources for each segment including dynamic updates thereof. The receiver may select any of them to download the segments, thus, balancing the load among the senders.

5.2.2 Analysis of Use Cases

We have described different use cases for multimedia streaming and how they can be applied in content-aware networks. In this section we will provide an analysis concerning content-aware network operations, such as *flow processing*, *caching and buffering*, and *QoS/QoE management* for the use cases in question and present some recent scientific advances.

5.2.2.1 Flow Processing

The term *flow processing* denotes adaptation operations as well as any forwarding behaviour that differs from traditional content-unaware forwarding mechanisms. The goal of *flow processing* is the reduction of overall traffic in order to serve a maximum number of users with the best QoE.

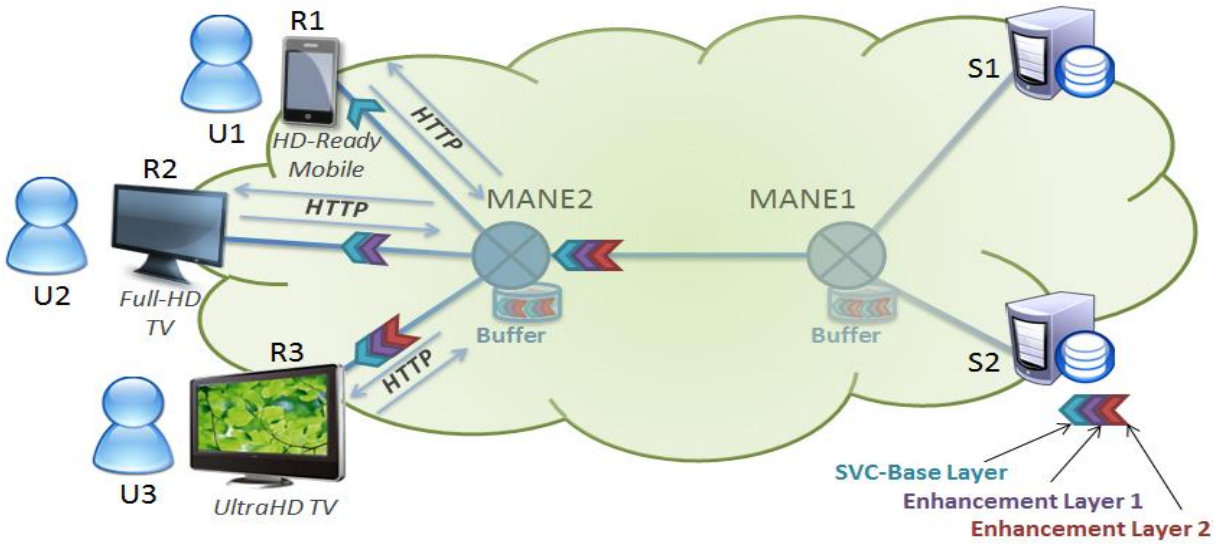


Figure 55: Adaptive HTTP streaming in Content-Aware Networks, adopted from [10].

In the **unicast use case**, the use of scalable media formats such as SVC in a content-aware network brings three main advantages.

First, the sender can easily adapt the content to the receiver's capabilities by only sending those layers that are actually supported by the receiver (e.g., in terms of spatial resolution).

Second, a MANE can perform efficient in-network adaptation of the content in reaction to network fluctuations. That is, when a MANE detects a decrease in available downstream bandwidth that prevents the entire content from being transmitted, it can drop some higher layers of the media stream, assuring continuous playout of at least the base quality at the receiver. Since the dropping of SVC layers in a unicast stream requires adjustments of RTP sequence numbers, we refer to this as *explicit adaptation*. That is, the MANE has to actively interfere with the RTP stream. Although the end user receives the content at a lower bitrate, the actual QoE may increase compared to the alternative which would cause the playout either to stall or to show too many visual artifacts due to high packet loss rate. As soon as the network conditions return to normal, the MANE can re-increase the number of forwarded layers. Each decision about dropping or forwarding SVC layers is triggered by a distributed network monitoring system, which detects network fluctuations and raises appropriate alarms.

The choice which SVC layers to drop or to forward is solved by an *Adaptation Decision-Taking Engine* (ADTE) as detailed later on in Section 5.3.1. The ADTE is actually not specific to SVC adaptation but is used for steering any adaptation of content – be it at the MANE or outside the network at the sender or receiver. Based on context parameters and the description of possible adaptation options, the ADTE runs an optimization algorithm that finds the best-suited choice for the current situation. In the case of in-network SVC adaptation, the set of context parameters is reduced to the network parameters and possible adaptations are limited to SVC layers, making this task rather simple and fast to compute.

Third, a MANE can signal its monitoring information about the network condition upstream to the sender, allowing for sender-side adaptation. While in-network adaptation is a good solution for mitigating short-term network fluctuations, it wastes bandwidth between the sender and the MANE in case of longer periods of decreased available bandwidth. In other words, if a higher layer packet is to be discarded at a MANE anyway, it is useless to transmit it to that MANE in the first place. Note, however, that network-aware adaptation at the sender needs at least one round-trip time (from MANE to sender) to take effect.

In the **multicast use case**, MANEs can adapt to changing network conditions by subscribing to or unsubscribing from multicast groups containing SVC enhancement layers. Conventional layered multicast is receiver-driven [48], i.e., the receivers control the subscriptions to multicast groups. Hence, in-network adaptation is achieved *implicitly* as the receiver controls it through subscription to appropriate SVC layers. MANEs aggregate and combine subscriptions from downstream entities – both receivers and MANEs – using them for subscribing to appropriate SVC layers upstream. ALICANTE adopts and extends the RDLM approach for the distribution of video content in multicast-based scenarios.

There are two possible ways for MANEs to assist the network-aware adaptation of multicast streaming. Either, downstream forwarding of one or more SVC layers can be temporarily truncated in case of congestion at an outgoing link as discussed in [181], or a MANE can control multicast group subscriptions by sending prune or graft messages to upstream neighbors as defined in RFC 3973 [182].

MANEs can also improve multicast functionalities of existing network infrastructures by enabling a *hybrid multicast* infrastructure. If native multicast is not supported, MANEs may perform overlay multicast with adjacent MANEs, so that they become bridges between native and overlay multicast, as it is done in ALICANTE [41]. Furthermore, ALICANTE supports traffic engineering as well as content and service classification and differentiation mechanisms (i.e., DiffServ and MPLS) that enable selective treatment of SVC layers, e.g., increasing priority and robustness of the base layer.

For the **P2P streaming use case**, a MANE may act as a peer, autonomously requesting pieces which it deems relevant for any connected receivers. Running a P2P engine on a MANE increases the processing requirements for this entity but it also offers a flexible and powerful way to participate in P2P streaming. The MANEs thus form a *P2P overlay network* (at the CAN layer) that may closely cooperate with the overlay network at the application layer.

The aforementioned flow-processing policies are also applicable to **adaptive HTTP streaming** with some noticeable differences. TCP uses reliable transmission that is unsuitable for in-network adaptation achieved through enhancement layer dropping. If a MANE simply drops TCP packets of an enhancement layer to avoid network congestion, it would trigger the sender to retransmit the packets after TCP timeout. For the streaming session, the retransmission of the packet wastes bandwidth and even if the packet reached the receiver eventually, it would probably arrive after the

playout deadline. Thus, for HTTP streaming a MANE shall act as a *transparent proxy cache* in combination with CDN functionality as will be described in the following section. As the adaptation logic is entirely located at the receiver side, in-network adaptation is achieved implicitly – similar to the multicast use case – by means of HTTP requests for layers that are supported by the receiver. Requests for individual SVC layers can be answered by different network nodes (or by the sender), depending on where these layers are buffered. Hence, adaptation occurs within the network, but without active participation by the MANEs.

The aforementioned in-network adaptation mechanisms – implicit or explicit – provide a powerful tool for mitigating the effects of network fluctuations. Furthermore, the adaptation decision-taking (i.e., the selection of which SVC layers to forward) has to be performed in a distributed manner. That is, each MANE computes its local adaptation decision and coordinates it with the other nodes in the network. Efficient, scalable signaling and coordination of adaptation decisions is still an open research challenge [7].

5.2.2.2 Caching and Buffering

MANEs can buffer previously requested content and may even act as CDN caches, i.e., proactively moving the content closer to the receivers. Note that the storage requirements for CDN-enabled MANEs are considerably higher than for mere buffering support. In this context, we use the term *buffering* to denote very temporary storage of data (e.g., within the sliding window of a live streaming scenario), whereas *caching* denotes storage for a longer, though limited, duration.

In the **unicast use case**, a CDN-enabled MANE can proactively perform caching of popular content. In particular, *prefix caching* decreases start-up delay while also reducing network traffic. When a receiver requests the content, the MANE starts streaming from its cache while requesting the suffix of the content from the sender [183].

The usage of SVC offers a trade-off between quality and availability to the MANE. The prefix cache may contain only the base layer for less popular content. Thus, the end user starts receiving only the base layer, but with a low start-up delay, and later the enhancement layers from the sender are added.

Proactive caching can also be used in the **multicast use case** to reduce mainly start-up delay but also network traffic, e.g., for IPTV-like services. Note that proactive caching is not applicable to live streaming sessions. Moreover, all receivers are served simultaneously via multicast RTP streams, abolishing the need for buffering at MANEs.

In the **P2P streaming use case**, a MANE can aggregate requests for a piece and buffer downloaded pieces for subsequent requests. Especially in live scenarios, almost all receivers share the same time window for the content; thus, each piece will be highly popular for a short time span. By buffering a piece during this time frame,

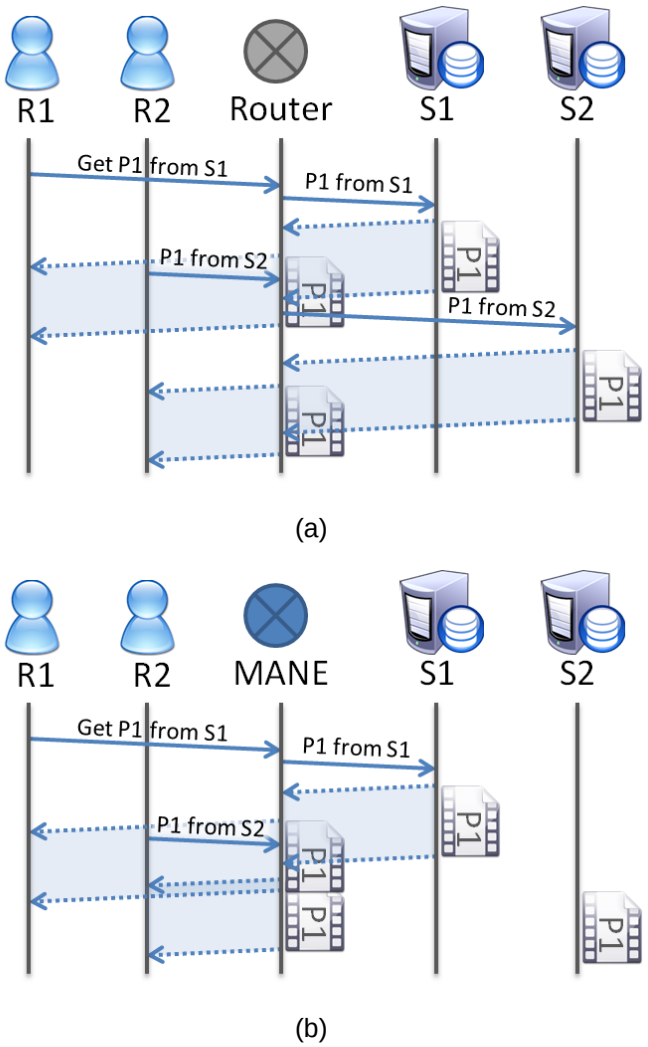


Figure 56: Request aggregation for P2P streaming for (a) conventional router and (b) MANE.

the MANE will be able to reduce network utilization and latency even with a limited buffer size. In most cases, such behavior is transparent to the peers within the traditional, application layer P2P overlay network.

Additionally, the MANE may also aggregate requests for the same piece to different senders and only forward one request which we call *content-aware buffering* as illustrated in Figure 56. For example, receiver *R1* selects sender *S1* for downloading a piece. The request passes through the MANE, which remembers the request and buffers the piece. Soon afterwards, receiver *R2* selects sender *S2* for the same piece. Unlike conventional routers (Figure 56 (a)), the MANE may intercept requests and transmit a buffered piece instead of forwarding the requests (Figure 56 (b)). This approach would constitute an evolutionary implementation of the CCN functionality [174]. However, a small drawback of this approach is that the peer selection of the first receiver might not always be the optimal selection. But once the MANE has downloaded and buffered the entire piece, the issue is alleviated.

A MANE might also act as a peer, proactively requesting pieces that may be needed in the near future by any receivers connected to it. Thus, the MANE increases the

replication of the content and moves it closer to the receivers. However, this puts some additional performance and storage requirements on the MANE.

Caching and buffering are integral parts of the **adaptive HTTP streaming use case**. In unicast mode, a MANE can provide CDN functionalities similar to the unicast use case discussed above. In contrast to RTP-based streaming, HTTP streaming immediately benefits from existing HTTP caching infrastructures [59][60] that may be deployed on top of content-aware networks. The multicast mode relies on *buffering* and *request aggregation* at the MANE for bandwidth-efficient streaming. As mentioned before, intelligent buffering at MANEs along the network path between sender and receivers constructs a bandwidth-efficient multicast tree. In order for the buffer size at the MANE to remain inside a reasonable limit, two requirements must be met. On the one hand, all receivers must share the same time window so that the popularity of a segment is temporarily limited. This time window can be signaled in the manifest file, as it is typically the case for live streaming services [50]. On the other hand, the MANE has to be aware of the streaming session in order to buffer the segments accordingly. The straightforward solution is for the MANE to parse the manifest file and to retrieve such information from there. An alternative solution would be that the MANE learns about the best buffering policy from a statistical analysis of the stream.

In the multisource mode of HTTP streaming, buffering at MANEs has similar effects as in P2P streaming. That is, MANEs aggregate requests (even to different senders) and perform content-aware buffering of downloaded segments for the duration of the sliding window of the streaming session. An open research challenge is the impact of the discussed request aggregation on the load balancing strategies between the senders.

In a recent study, Lederer et al. have proposed a peer-assisted HTTP streaming architecture compliant with DASH [184]. For each segment, the server lists a selection of possible peers in the manifest file. Those peers have already downloaded the segment and provide it through local HTTP servers. Other clients download segments from those peers if their buffer fill level guarantees smooth playback. Even under the consideration that clients have asymmetric Internet connections with significantly lower uplink bandwidth than downlink bandwidth, the solution reduces server bandwidth by up to 25%. While that work [184] focuses on conventional client peers, MANEs can act as peers just as well. Since MANEs are usually not limited by asymmetric connection speeds, server bandwidth can be further reduced. To validate this assumption we performed simulations with the same setup as [184], except that MANEs acting as peers had symmetric connection speeds (15 peers with 16 Mbps and 25 peers with 8 Mbps). Like in the original evaluation, the maximum bitrate of the content was set to 1,400 kbps. The simulation results of server bandwidth requirements over time are shown in Figure 57. Original server bandwidth for asymmetric connection speeds of peers is labeled *Peer Assisted*, server bandwidth for symmetric connection speeds is labeled *Peer Assisted (MANE)*. MANEs acting as peers in this HTTP streaming scenario were able to reduce server bandwidth by up to 29.5%. It should be noted that the simulation did

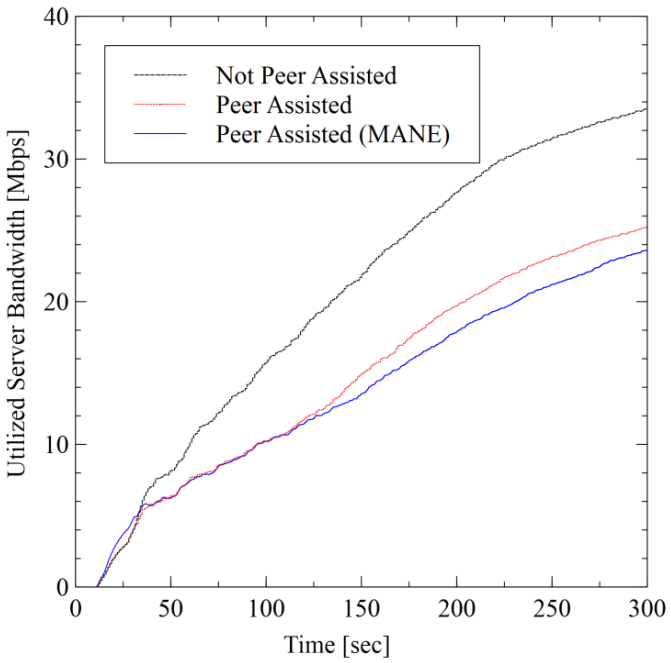


Figure 57: Simulation of peer-assisted HTTP streaming with MANEs as peers, adopted from [10].

not consider frequent updates of the manifest file, which contains the current list of peers. Updating the manifest file every 60 or 120 seconds would bring further performance gains.

The deployment of SVC in HTTP streaming also brings benefits to caching and buffering mechanisms. While HTTP streaming of non-layered media formats requires switching between different content representations (e.g., frame rate, resolution, quality) for adaptation, SVC-based adaptation is performed by adding/removing enhancement layers. Thus, the MANE only has to cache one SVC stream instead of multiple streams for different representations. This both reduces storage requirements and increases cache performance. Simulations conducted by Sánchez et al. compared the combination of SVC-based HTTP streaming and a streaming-optimized caching strategy to AVC-based streaming under Least Recently Used (LRU) strategy [59]. Their results show that, thanks to SVC and the optimized caching strategy, the cache hit ratio can be increased by up to 11.5 percentage points (from 52.8% to 64.3%) for congestion in the cache feeder link (i.e., the link between the sender and the cache) and by up to 25.7 percentage points (from 30.9% to 56.6%) for congestion in the access links.

5.2.2.3 QoS/QoE Management

A primary goal of content-aware networking is to manage and optimize the QoS and consequently QoE at the application level.

The term *QoS* describes properties of the network that influence the transport of media flows. Metrics like delay, packet loss, and jitter are used to measure QoS. The

more recently coined term *QoE* targets the degree of delight or annoyance of the user about an application or service. Besides QoS parameters, also user-related factors (e.g., expectations) as well as terminal capability and performance play a role in QoE. QoE is typically measured as MOS based on user ratings. More information on QoS and QoE can be found in [185].

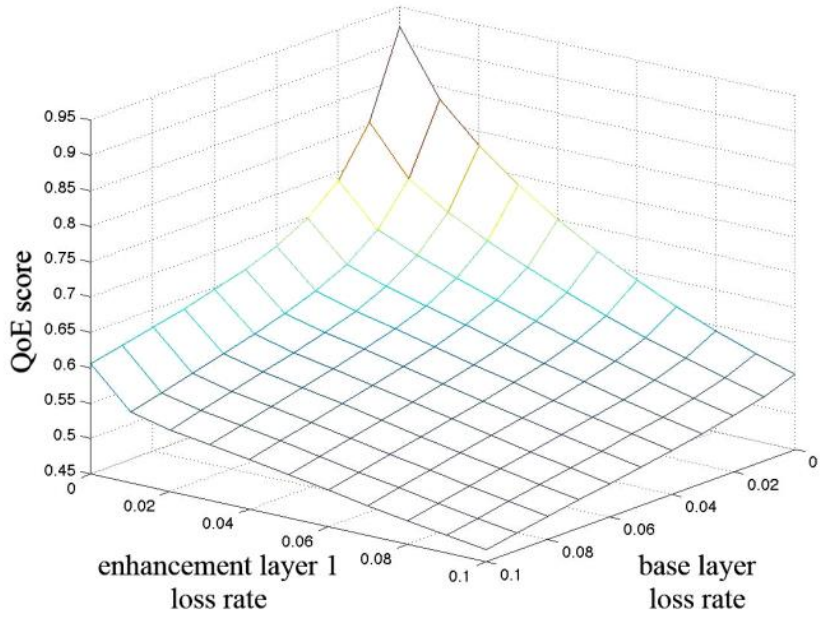
QoS/QoE optimization can be achieved through context-aware mechanisms both at the end-user side and within the (core) network. At the end-user side, several aspects of the usage environment (such as terminal capabilities) can be taken into account during content request and consumption. Other aspects, such as user preferences and the current status of the end-user terminal, may dynamically affect the configuration of the requested SVC stream.

Within the (core) network, context-awareness relates to the current condition of the network. Network monitoring enables MANEs to react to network fluctuations by performing in-network adaptation of SVC content. Monitoring information is used *locally* and is *aggregated* at the CAN level for managing the network behavior and establishing long-term adaptation policies [176].

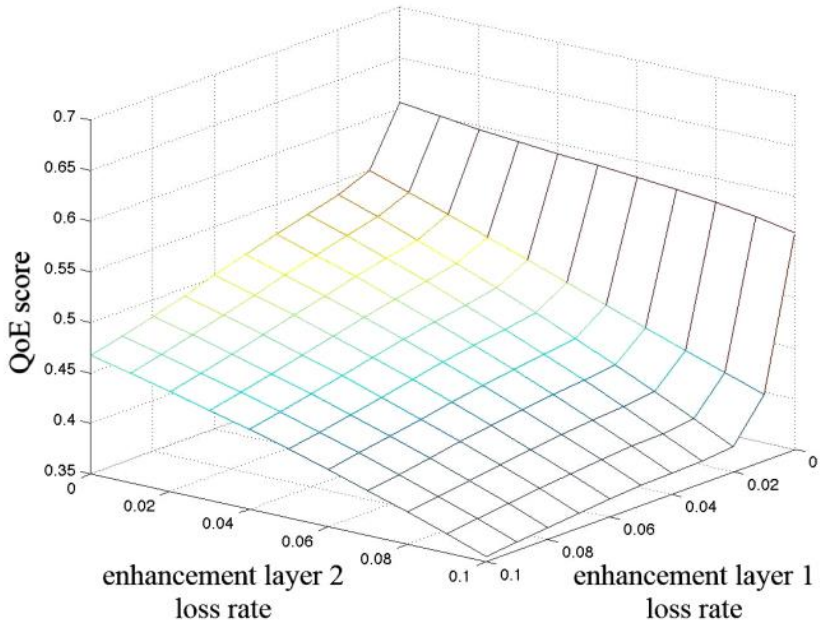
One important aspect is the appropriate media encoding configuration. In the ALICANTE project, we have developed encoding guidelines for SVC that facilitate distributed adaptation as discussed in Chapter 3. Those guidelines comprise a description of typical resolutions, which and how many bitrates to use for each resolution, appropriate scalability modes (spatial, SNR, etc.), how to combine these modes, differences among use cases, and more. On the other end of the media delivery chain, the project investigates the video quality at the client when there have been packet losses in any of the SVC layers. Evaluations are performed using a no-reference QoE tool called *ALICANTE Pseudo-Subjective Quality Assessment* (A_PSQA) [186], which uses a continuous QoE score ranging from 1 (excellent) to 0 (bad) to estimate video quality based on packet loss characteristics. The SVC streams used in the experimental setup comprised three layers. Figure 58 shows how the quality of a video degrades for packet loss at any of these layers.

The QoE scores are subsequently used for triggering the adaptation and enhancing the granularity by which the system reacts to context variations. Thus, QoE evaluations are a vital part of advanced adaptive media delivery systems.

As already mentioned, SVC enables a fine-grained control over the QoE at the network level. A non-scalable media format will suffer from severe QoE degradation if not all packets of the stream are transmitted. With SVC, lower layers can be prioritized, maintaining *smooth and undistorted playout* with controlled QoE degradation. SVC can also be conveniently combined with error recovery techniques at the decoding side, in order to further enhance the QoE perceived by the user. It should be noted that we do not address issues related to wireless transmission of SVC. For more information on that topic, the interested reader is referred to [187].



(a)



(b)

Figure 58: QoE scores vs. (a) loss rate at SVC base layer and enhancement layer 1, and (b) loss rate at enhancement layer 1 and enhancement layer 2 with base layer loss rate of 10%, adopted from [10].

As a conclusion, Table 16 summarizes the discussed CAN-related challenges for each of the described use cases. Note that for *QoS/QoE management* we make no explicit distinction between the use cases.

Table 16: Summary of CAN-related challenges addressed by the presented use cases, adopted from [10].

Use case	CAN challenge		
	Flow processing	***Caching & buffering***	***QoS/QoE management***
Unicast	• explicit adaptation • signaling adaptation decision to sender	• SVC-based prefix caching for low start-up delay	• local & aggregated monitoring of network conditions • smooth, undistorted playout via SVC
Multicast	• implicit adaptation • multicast bridges for hybrid multicast • differentiated forwarding of SVC layers	• SVC-based prefix caching for low start-up delay • no buffering	
P2P streaming	• explicit adaptation • peer for CAN P2P overlay network	• aggregating requests • content-aware buffering of pieces within the sliding window	
Adaptive HTTP streaming	• implicit adaptation • transparent proxy cache	• SVC-based prefix caching • *multicast:* buffering within the sliding window • *multisource streaming:* aggregating requests & content-aware buffering within the sliding window	

5.2.3 Conclusions

Scalable media coding formats (such as SVC) in combination with in-network adaptation – and, as a consequence, its capabilities in terms of flow processing, caching/buffering, and QoS/QoE – are becoming promising concepts towards enabling content-awareness within the (core) network. This concept is referred to as Content-Aware Networking and provides an elegant, powerful, and flexible tool to accommodate existing and imminent challenges for a variety of traditional and emerging use case scenarios in the context of multimedia delivery within the Future Internet.

We have argued that sender-driven use cases such as unicast and multicast streaming greatly benefit from content-awareness for routing and forwarding. In P2P streaming, the combination of enhanced forwarding and buffering techniques may allow MANEs to collaborate with receivers within the P2P network. Content- and context-aware caching/buffering are furthermore important aspects in the adaptive HTTP streaming use case.

Interesting challenges remain, such as the integration of on-the-fly QoE evaluation of SVC content for adaptive media streaming or the further improvements to the involvement of MANEs into P2P streaming. As future trends indicate more advanced video compression technologies targeting resolutions beyond 1080p, (e.g., a new scalable extension for HEVC [57][188]), efficient and reliable caching and buffering at MANEs becomes increasingly important in order to reduce overall network loads. Furthermore, adaptive HTTP streaming becomes increasingly popular due to its relatively easy deployment. Therefore, future work will focus on how MANEs can further improve the existing HTTP infrastructure.

5.3 Distributed Adaptation Framework

5.3.1 Adaptation Framework Architecture

Existing and future media ecosystems need to cope with the ever-increasing heterogeneity of networks, devices, and user characteristics collectively referred to as (usage) context. The key to address this problem is media adaptation to various and dynamically changing contexts in order to provide a service quality that is regarded as satisfactory by the end user. The adaptation can be performed in many ways and at different locations, e.g., at the edge and within the network resulting in a substantial number of issues to be integrated within a media ecosystem.

An important aspect towards Universal Multimedia Access (UMA) [189] and Universal Multimedia Experience (UME) [190] is the adoption of scalable media coding formats such as SVC enabling edge and in-network adaptation. In this section, we discuss the exploitation of these scalable media formats within the (core) network – featuring in-network adaptation – in order to optimize the network resources utilization, and at the edge of the network, for the adaptation from/to heterogeneous formats, devices, and platforms. This is achieved by means of overlay networks, where the adaptation is coordinated in a distributed fashion.

The ALICANTE system architecture introduces two new virtual layers, i.e., HB and CAN layers, on top of the existing network infrastructure (cf. Section 2.3.1). This approach brings both content-awareness to the network layer and context-awareness to the user environment. This section focuses on the adaptation framework of that architecture. Content delivery in the core network relies on scalable media formats such as SVC. This enables content-aware adaptation according to the network conditions at the CAN layer, i.e., within the MANEs.

Home-Boxes are enhanced home-gateways with media processing capabilities. They can serve as home media servers, enable users to act as content providers, and keep track of the capabilities of connected terminals. Home-Boxes form a virtual HB layer that enables context-aware adaptation towards end-user terminals and user preferences. For example, screen resolution and decoding capabilities are taken into

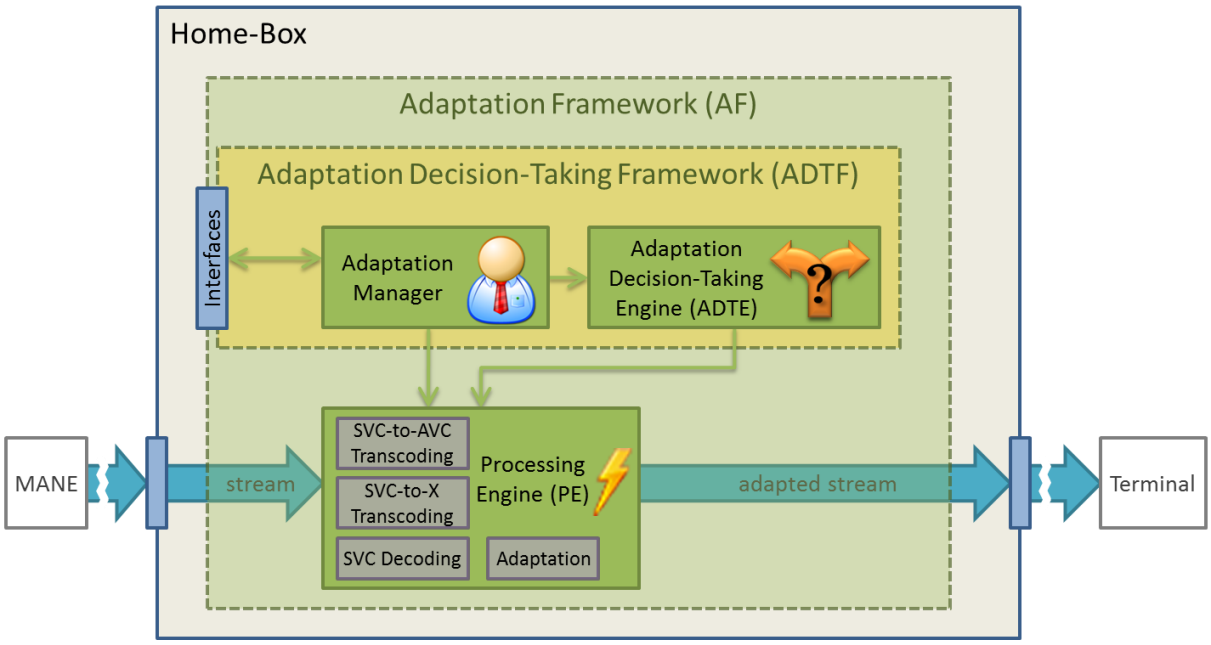


Figure 59: Modules of the Adaptation Framework at the Home-Box, adopted from [9].

account at content request time. For legacy terminals that do not support SVC, Home-Boxes are able to transcode content to non-scalable media formats (e.g., MPEG-2, MPEG-4 AVC). On the server side, corresponding HB layer entities are implemented as software modules.

Figure 59 shows the conceptual architecture of the Adaptation Framework (AF) at the Home-Box. The same conceptual architecture applies to the MANE (with the exception of transcoding components). The AF comprises the ADTF and the Processing Engine (PE). Inside the ADTF, the ADTE computes the best-suited adaptation decision for a stream, while the Adaptation Manager module coordinates all active streams and collects relevant information (e.g., network monitoring). The adaptation decision is fed into the PE, which performs adaptation (and transcoding at the Home-Box). For a detailed description of the ALICANTE adaptation architecture, the interested reader is referred to [8] and [9].

5.3.1.1 Adaptation Decision-Taking

Local adaptation decisions are taken based on an optimization algorithm, determining the most suitable adaptation for a given content as described in Section 5.4.2 later on. The various local adaptation decisions have different purposes, depending on the location they are performed in. For example, adaptation decisions in the network focus on dynamic adaptation towards network conditions, while adaptation decisions at the Home-Box mainly target the capabilities of the user terminal and the QoE. The distribution of adaptation decisions also depends on the streaming mechanism, as, e.g., RTP multicast streaming is handled differently from HTTP streaming.

5.3.1.2 Coordination of Adaptation Decisions

The adaptation decisions at the aforementioned different locations are taken locally. Nevertheless, some degree of coordination between those adaptation decisions is needed in order to assure a stable end-to-end quality. The coordination can be performed between the adaptation nodes as well as based on a central entity.

Between adaptation nodes, adaptation decisions can be signaled upstream to optimize network resource utilization. If a MANE decides to adapt to a lower SVC layer, this information can be signaled to its upstream neighbor (e.g., via layer un-subscription in multicast scenarios) to avoid unnecessary traffic (cf. Section 5.2.2.1).

Within the ALICANTE architecture, a content-aware network domain is administered by an entity called CAN Manager. This CAN Manager coordinates the MANEs by distributing adaptation policies to them [9] that have been negotiated as Service-Level Agreements (SLAs) [191][176] between network, CAN, and service providers. The policies describe which classes of media services are allowed to be adapted and to which extent, the minimum bandwidth that shall be allocated for each media flow, as well as rules for minimum and maximum aggregated bandwidth for different traffic classes. The adaptation logic of each MANE is configured based on these policies as discussed later on in Section 5.4.2. The policies can be updated during operation to accommodate changes in the network infrastructure.

5.3.1.3 SVC Tunneling

As discussed in Chapter 4, our distributed adaptation framework relies on SVC (layered-multicast) tunneling, inspired by IPv6-over-IPv4 tunnels. That is, within the CAN only scalable media resources – such as SVC – are delivered adopting a layered-multicast approach [48]. This allows the adaptation of scalable media resources by MANEs [44], implementing the concept of distributed adaptation [192][193]. At the border to the user (Home-Box), adaptation modules are deployed enabling device-independent access to the SVC-encoded content by providing X-to-SVC and SVC-to-X transcoding/rewriting functions with X={MPEG-2, MPEG-4 Part 2, MPEG-4 Part 10 (AVC) etc.}. An advantage of this approach is the reduction of the load on the network (i.e., no duplicates), making it free for other data (e.g., more enhancement layers).

Note that SVC tunneling is also applicable to unicast scenarios due to dynamic SVC-based adaptation, although multicast scenarios bring higher gains in terms of network resource utilization.

5.3.2 Related Work

Similar to ALICANTE, several other research projects target media adaptation and content-aware networks. The FP7 Project ENVISION [194] proposes a multi-layered

content distribution approach, targeting optimized end-to-end performance and content adaptation during distribution. However, it does not focus on QoE aspects on the client side. Dynamic and distributed adaptation of scalable multimedia content has been proposed by the FP6 Project DANAE [195]. With a focus on the MPEG-21 standard, it pioneered in the area of interoperable adaptation approaches [196]. The FP7 Project MEDIEVAL [197] aims at evolving the Internet architecture for efficient video transport. In particular, it targets cross-layer SVC adaptation ranging from the application layer down to the physical layer [198]. The FP6 Project ENTHRONE [199] developed a system architecture to cover the entire multimedia distribution chain, focusing on end-to-end QoS performance and network management. These projects tackle important aspects of media-aware adaptation along the delivery path. In the following we discuss several adaptation-related features of the ALICANTE architecture.

5.3.3 Adaptation at Network Edges

Home-Boxes are enhanced home-gateways deployed within the ALICANTE architecture at the end users' premises at the border to the network. They feature advanced adaptation capabilities and are interconnected to form a virtual HB layer that facilitates caching and P2P streaming [200]. To the end user, a Home-Box acts as a multimedia server that adjusts media services to the connected devices. A similar concept for a home media server has been recently introduced by a major industry player [201].

In the context of the ALICANTE adaptation framework, the Home-Box establishes streaming sessions upon the end user's request, receives a scalable media stream, adapts and transcodes it, and ultimately sends the resulting stream through the home network to the end user's device. In the following, we briefly explain the technical realization of the media processing chain on the Home-Box for RTP streaming, adaptive HTTP streaming, and P2P streaming.

5.3.3.1 RTP Streaming

RTP streaming of SVC supports two different modes [202]. Single-session transmission (SST) mode transports the entire SVC bitstream in a single RTP session and is typically used for unicast streaming, whereas multi-session transmission (MST) mode transports each SVC layer on a different RTP session and is thus better suited for multicast streaming. For MST mode, the receiving Home-Box rearranges (or *multiplexes*) the data from the SVC layers into a single SVC bitstream. To ensure the synchronization between the packets of the RTP sessions, the timestamps of RTP packets or special packets signaling the cross-session decoding order number (CS-DON) can be used. At this stage, any undesired SVC layers that have not been removed by the server or the MANEs already are discarded. If the end-user terminal supports SVC, the bitstream is piped into an RTP unicast

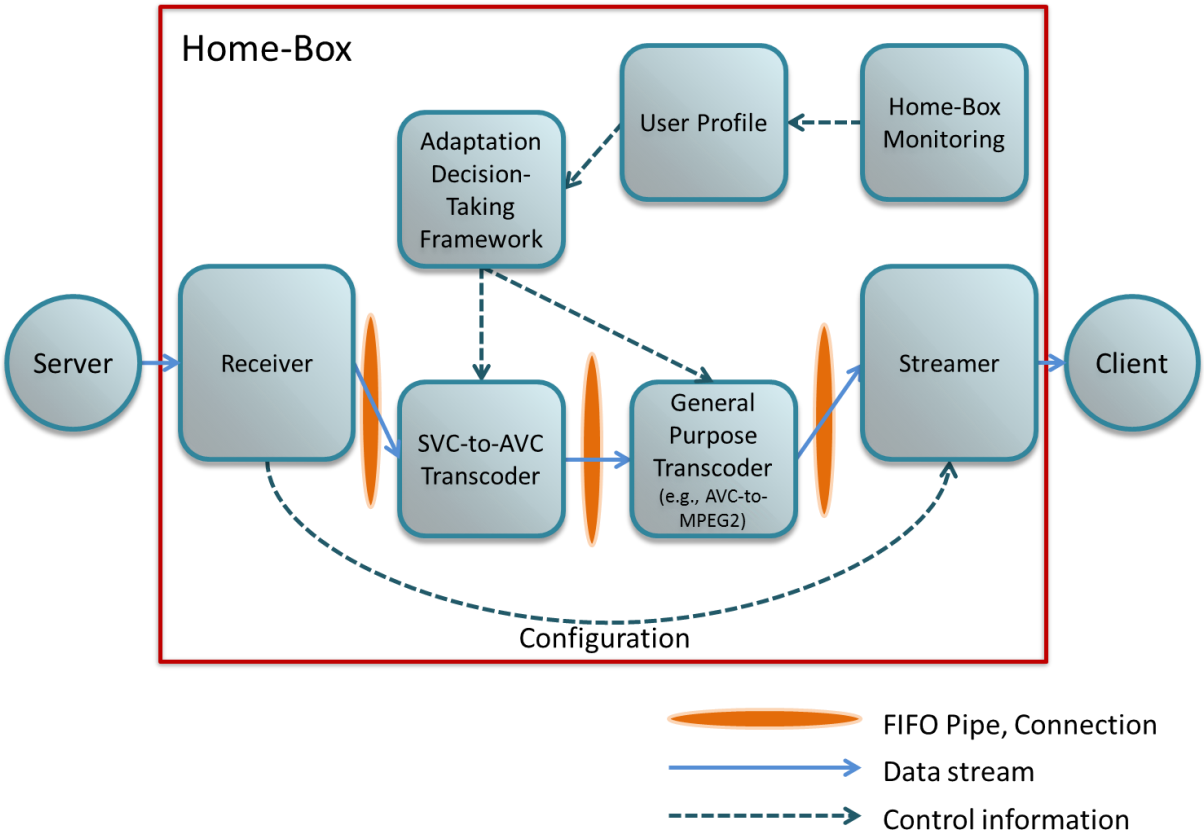


Figure 60: Home-Box adaptation tool chain for RTP streaming, adopted from [9].

streaming module for transmission to the terminal. Otherwise, the received bitstream is piped into an on-the-fly transcoding module. In order to optimize the processing performance for AVC support, the SVC bitstream is first fed into a fast SVC-to-AVC transform-domain transcoder. For support of any other legacy coding format (e.g., MPEG-2), the AVC bitstream can be further fed into a *general-purpose transcoder* (GPT). The GPT is based on FFmpeg [165] and thus currently supports more than 50 video coding formats [203]. The GPT also readily adjusts resolution and bitrate to the device's profile [204]. The transcoded bitstream is then streamed to the terminal. The Home-Box adaptation tool chain for RTP streaming is illustrated in Figure 60. Dynamic SVC-based adaptation is performed within the SVC-to-AVC transcoder module, whereas any further adaptation (e.g., adjustment to a specific resolution) is performed at the General-Purpose Transcoder module.

To manage the adaptation to heterogeneous devices, the Home-Box keeps a database to store the devices' capabilities (i.e., supported media formats, display resolution, etc.), associated user preferences, and information on active sessions [205].

Adaptation at the RTP server is typically performed based on clients' QoS reports via the RTP Control Protocol (RTCP) [49]. Future work will investigate to which extent in-network adaptation influences and complements such server-side adaptation.

5.3.3.2 Adaptive HTTP Streaming

For DASH, the Home-Box deploys a DASH proxy module to handle adaptation, transcoding, and re-streaming. The DASH proxy relieves the terminal from its adaptation logic. Thus, a simple, non-adaptive HTTP streaming client can be used at the terminal. In preparation of the streaming session, the DASH proxy generates a *local MPD* describing the transcoded segments that the client at the terminal will be able to request from the Home-Box. This local MPD contains a single representation as all adaptation is already performed on the Home-Box. An example of a local MPD is provided in Annex E. Upon the client's request for a transcoded segment, the DASH proxy downloads and transcodes the corresponding SVC layers from the server. This on-request processing of segments introduces a constant delay consisting of the round-trip time between the Home-Box and the server, and the implementation-dependent delay of the transcoder. Thus, we recommend that the client at the terminal uses HTTP pipelining [206] to request multiple segments simultaneously.

In SVC-based DASH, the SVC layers are provided in multiple representations (cf. Section 3.5). A normal SVC bitstream has enhancement layers located at each frame as shown in Figure 61 (a). For SVC-DASH each temporal segment is split into multiple *chunks*, one per layer. Each of the chunks contains several frames for one layer as depicted in Figure 61 (b).

In order to rearrange the frames and layers into the correct decoding order at the client, some information is needed concerning the location of the NALUs of the frames of each layer in the original stream. One possible solution is the integration of NALU byte ranges in the MPD as proposed by Müller et al. [61]. For each segment, the NALUs belonging to different frames are described by their byte ranges in the segment. The DASH client module has to understand those byte range descriptions and reorder the NALUs into their proper decoding order before passing the stream to the SVC decoder. However, this approach is a custom extension to the standardized MPD format and results in large MPDs.

We implemented a small client module that parses the chunks, locates the NALU boundaries inside them and reorders them in correct decoding order into the multiplexed SVC output bitstream. This *SVC multiplexing* module is located between the DASH client and the SVC decoder and does not require any additional signaling in the MPD. A corresponding *SVC demultiplexing* module splits the original SVC bitstream at the server into chunks. We made the SVC multiplexing and demultiplexing modules available as open-source software at [12].

The received and multiplexed segments are piped into the transcoding module, similar to the RTP case. The transcoded segments are provided for HTTP download by the terminal. For a detailed description of the Home-Box adaptation for DASH, the interested reader is referred to [9].

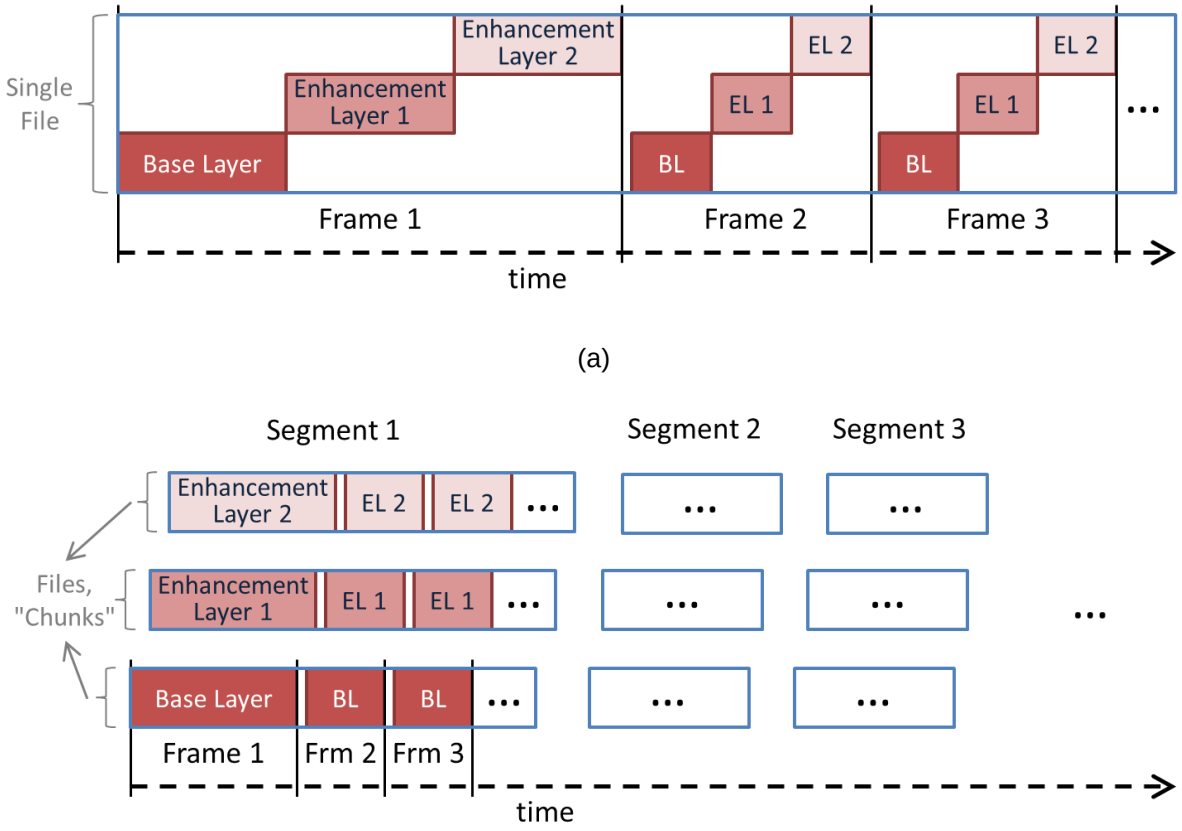


Figure 61: Segmentation of SVC bitstream for DASH. SVC layers in (a) original bitstream and (b) segmentation for DASH.

5.3.3.3 P2P Streaming

ALICANTE deploys the *libswift* implementation [207][208] of the Peer-to-Peer Streaming Peer Protocol (PPSPP) [209] for P2P streaming. For simplicity, the current implementation of P2P streaming on the Home-Box deploys the HTTP gateway (also known as swift-to-HTTP proxy) of the *libswift* implementation [210], which is located before the DASH proxy [9]. The DASH proxy retains the adaptation logic (in contrast to more sophisticated native P2P streaming adaptation mechanisms [179]). Figure 62 illustrates the adaptation tool chain for DASH and P2P streaming at the Home-Box.

5.3.4 In-Network Adaptation

MANEs perform dynamic in-network adaptation to mitigate the effects of network congestion. Each MANE has a local ADTE that computes whether to adapt a media stream. The adaptation processes for multicast and unicast streaming have to be considered separately. Multicast streaming deploys RTP MST mode, where SVC layers are transmitted over separate RTP sessions and are rearranged by the receiver. Thus, multicast trees for the different SVC layers are built. MANEs realize

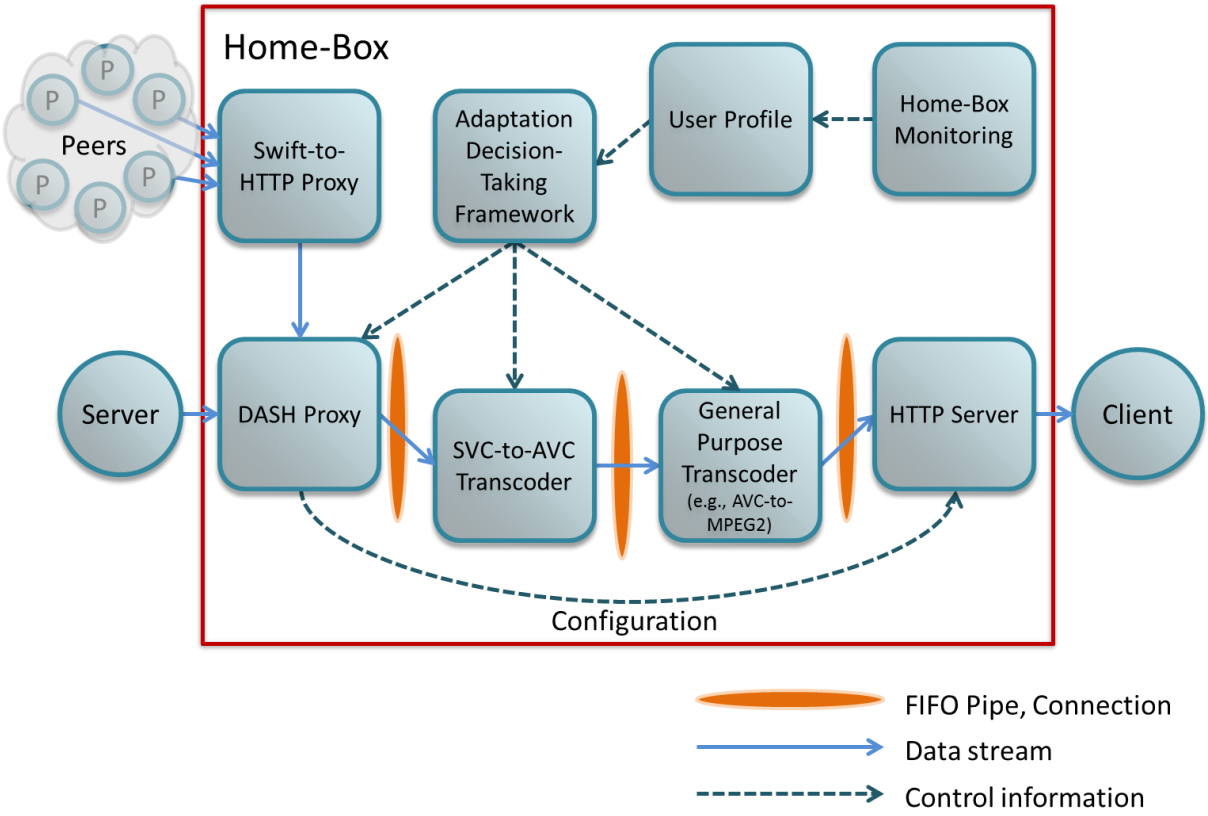


Figure 62: Adaptation tool chain for DASH and P2P streaming, adopted from [9].

dynamic adaptation by pruning (or conversely grafting) the multicast tree corresponding to a specific SVC layer.

RTP-based unicast streaming is typically realized via SST mode, where the entire SVC stream is packed into a single RTP session. In order to perform adaptation, a MANE de-packetizes the RTP stream, analyzes the SVC header, and filters SVC layers accordingly [211]. The RTP re-packetization module updates the sequence number field of the RTP packet headers if needed. Alternatively, unicast streaming could also be realized via MST mode, using separate ports for separate layers.

Due to the issues related to in-network adaptation of TCP streams raised in Section 5.2.2.1, the current implementation of the MANE only performs adaptation for RTP/UDP-based streaming.

5.3.5 Scalability Considerations

The proposed techniques act on a per-flow basis, thus, some scalability considerations (in terms of number of concurrent flows) have to be taken into account. Adaptation decision-taking at a MANE has to handle many different flows in parallel, requiring a very lean and efficient implementation of the ADTE. The processing overhead can be controlled by the update frequency of adaptation decisions. For example, the decision to drop an SVC layer shall be triggered immediately when network monitoring indicates congestion, but the decision to add a

layer back to the stream can be delayed by a scheduler until CPU utilization has declined to a certain threshold. In contrast to the MANE, adaptation decision-taking at the Home-Box has to take more parameters into account, including terminal capabilities and user preferences, but has fewer flows to handle. A Home-Box in a typical household might have to handle up to five concurrent flows. However, any adaptation or transcoding operations have much higher computational complexity and resource demands than the adaptation decision-taking.

Transcoding at the server side and at the Home-Box are computationally expensive parts of SVC tunneling. Transcoding to SVC on the server-side has only to be performed once per video and can be performed offline prior to streaming. Transcoding from SVC to other formats on the Home-Box demands less resources but the Home-Box has to be dimensioned to support a handful of concurrent flows.

In-network adaptation in MST mode relies on receiver-driven layered multicast, thus, the usage of SVC does not put any overhead on this approach. In SST mode, RTP de-packetization and re-packetization limit the number of concurrent flows. A prototype implementation on an off-the-shelf WiFi router supported concurrent adaptation of several flows in 2008 [46], dedicated hardware and improved algorithms may lead to a higher number of possible concurrent flows.

5.4 SVC Adaptation

The layered structure of SVC enables fast and efficient adaptation. For adaptive SVC streaming, the following research questions arise as discussed in Section 2.3.3.1: Where to adapt? When to adapt? How often to adapt? How to adapt?

In this section, we discuss related research on SVC adaptation for RTP-based and HTTP-based streaming, describe the adaptation logic implemented for the ALICANTE streaming system, and propose a technique for enabling smooth transitions between representations.

5.4.1 Related Work

In the recent years, the focus of research on video adaptation has changed from network-centric strategies towards user-centric approaches with increasing involvement of QoE considerations. This trend has also gained momentum through the exploration of adaptive streaming of SVC via DASH. This section provides a discussion of related work on SVC adaptation strategies, the corresponding integration of QoE-awareness and special considerations for different transport modes.

5.4.1.1 Adaptation Strategies

The technical challenges of SVC adaptation are discussed in [212], which also provides an overview of design principles, standard tools, and methods for adaptation decision-taking.

The application of SVC for IPTV and corresponding adaptation is discussed in [75], and an evaluation of bandwidth requirements for SVC in IPTV services was performed in [76] and [77] (cf. Section 3.2.1). Design options for SVC in-network adaptation are discussed in [45]. Kofler et al. [46] have demonstrated SVC adaptation on off-the-shelf routers. For a detailed discussion of in-network SVC adaptation, the interested reader is referred to [213].

The goal of SVC adaptation techniques comes down to the question of which NALUs of the SVC bitstream shall be dropped to meet the bandwidth constraints. Niedermeier et al. [67] have evaluated an optimal extraction path of layers from an SVC stream based on objective and subjective quality evaluations (cf. Section 3.2.1). A similar approach for SVC adaptation was used in a recent study by Li et al. [214].

Eichhorn et al. [128] have conducted a study to assess which scalability dimension (spatial, temporal, or quality) is preferred by viewers on mobile devices. Their results show that viewers clearly prefer a lower quality version of a video over a spatially downscaled version.

As mentioned in Section 3.2.1, Nur et al. [70][71] have proposed an SVC adaptation technique based on a utility function of SVC layers. The utility function ranks SVC extraction points by weighting spatial, temporal, and quality layers. The weights are based on a model devised from subjective evaluations for videos classified by motion intensity and structural features.

A common drawback of those adaptation techniques is that the models for ranking SVC layers depend on the specific SVC encoding configuration, e.g., number of quality layers and QPs of each layer, spatial resolutions and so on. Most models are only validated with the same set of SVC configurations the model was based on.

The study by Nur et al. [70] also explores an adaptation approach with asymmetrical key/non-key frame adaptation for improved flexibility. The approach exploits the fact that SVC combines two techniques for drift control. Frames of the lowest temporal layer are marked as key frames. For these frames, motion compensation is performed at the base quality layer. All non-key frames use the quality layer for motion compensation [27]. With the NALU prioritization scheme proposed by Nur et al., key frames can be adapted to a lower MGS layer than non-key frames without interfering with motion compensation.

During RTP/UDP-based streaming, packet loss significantly influences the resulting video quality [215][216][217]. Even with good error concealment at the SVC decoder [218][219], it is often better to switch to a lower SVC layer to improve the QoE. A model for mapping QoS parameters such as packet loss and jitter and SVC encoding configuration parameters onto expected QoE is presented in [211]. The expected

QoE then guides the video adaptation in the proposed streaming system. Note that unequal error protection can be deployed for improving the robustness of SVC streaming [40].

Building a QoE model to guide adaptation decisions typically requires a large set of subjective ratings. To acquire them is very time-consuming. One way to reduce the number of subjective ratings required for a QoE model is the Pseudo-Subjective Quality Assessment (PSQA) [220][221]. PSQA deploys a Random Neural Network (RNN) [222] to learn a QoE model from few subjective test results. The approach can be used to estimate the quality impairment through packet loss of different types of frames, as implemented in the A_PSQA tool [186]. The deployment of PSQA for SVC is evaluated in [223]. Based on the PSQA model for SVC, Ksentini and Hadjadj-Aoul [224] have developed an adaptation technique. The proposed algorithm measures the packet loss and computes the expected video quality for each SVC layer and selects the one with the highest expected quality.

The perceived video quality also depends on temporal variations of the per-frame quality [225]. This effect is particularly relevant for modeling distortion due to packet loss as reported in [226]. Furthermore, temporal hysteresis effects can be observed for quality changes [227]. That is, continuous subjective quality ratings have shown that viewers react strongly to hard quality decreases (e.g., due to packet loss) by assigning poor scores. But when the video quality resumes to its previous high state, their ratings do not follow this increase immediately, but are still affected by the memory of the period of bad video quality. Based on these observations, Joseph and De Veciana [228] have devised an adaptation algorithm that strives to reduce temporal variations in video quality.

When serving multiple clients at the same time, the rate allocation between the clients has to be considered. Equal rate allocation to all streams is in general suboptimal due to different RD characteristics of the streams. Typically, videos with low RD performances (e.g., due to high Spatial and Temporal Information) should be allocated higher rates than videos with better RD performance. It can be shown that an optimal allocation is one where the slopes of the rate points on the (concave) RD curve of all videos are equal (or near-equal in a practical deployment) [229]. If two slopes were unequal, allocating a higher rate to the video with the higher slope would increase the quality more than that of the video with the lower slope would decrease from that reallocation. Based on this approach, Hansen and Hissam [230] have developed a distributed Quality of Service resource allocation model (D-Q-RAM) for wireless networks. Another framework for multi-video rate allocation incorporating multiple agents (e.g., proxies or MANEs) was proposed by Chakareski in [193].

The two approaches [230] and [193] also show how adaptation can be distributed among multiple network nodes. Given that network nodes know not only the bitrates but also the quality characteristics (i.e., the RD curves) of all transported video streams, each node can decide how to adapt the video streams while maintaining optimal viewing qualities for all end users. In [193], the network nodes are assumed to know the distortions caused by previous nodes. In [230], the network nodes deploy

a feedback mechanism that informs other nodes of the chosen value of the aforementioned slope on the utility curves. The study also investigates different algorithms for handling conflicting slope values. Earlier works on distributed adaptation, such as [231] or [232], have focused on architecture and infrastructure challenges for distributed adaptation.

For P2P streaming, a receiving peer must decide which pieces of the various SVC layers to request from which other peer. Eberhard et al. [179] have evaluated piece-picking algorithms for SVC-based P2P streaming in three different scenarios. A comprehensive description of a QoE-aware SVC-based P2P streaming system is provided in [233].

In addition to the discussed adaptation approaches, more unconventional approaches comprise adaptation decision-taking based on ambient illumination as studied in [234][70] and the influence of sensory effects on the QoE [235][236][237]. Sensory effects are enrichments for multimedia content, such as ambient light, wind effects, vibration effects, or olfactory effects. These effects enable viewers to immerse into the content so that they become less perceptive of visual artifacts of the content.

5.4.1.2 Adaptation for HTTP Streaming

While packet loss is a major adaptation concern of RTP/UDP-based streaming, TCP-based transmission ensures the arrival of packets for HTTP streaming. Thus, the adaptation in HTTP streaming has to avoid playback stalls due to late arrival of video segments. Adaptation techniques typically rely on the bandwidth computed from previously downloaded segments and on the buffer level. An adaptation logic that relies on the measured bandwidth and current buffer level for AVC and SVC streaming is presented in [238] and [61].

Several studies have investigated the factors that influence the QoE in (non-adaptive) HTTP streaming. Hoßfeld et al. [239] have researched the trade-off in terms of QoE between initial delay of HTTP streaming services and stalling during playback. Viewers clearly prefer higher initial delay over stalling. For example, a single 1-second stalling event during playback of a 30-second video was rated worse than 32 seconds of initial delay. The findings also show that the QoE impact of initial delays fits a logarithmic model. That is, the QoE impact of an initial delay t can be modeled as $a \cdot \log(t) + b$ (with model parameters a and b). On the other hand, the QoE impact of playback stalling fits an exponential model (see also [240]). That is, a stalling duration s impacts the QoE following an exponential model $a \cdot e^{-b \cdot s} + c$ (with model parameters a, b, and c).

Similarly, Mok et al. [241] have proposed three application performance metrics that influence the QoE in HTTP streaming: initial buffering time, mean rebuffering duration, and rebuffering frequency. They also devise a linear model to estimate the MOS from these three metrics. But as shown in [239], this linear model is only partially accurate.

In adaptive HTTP streaming another impact factor comes into play: flickering due to switches between representations. Ni et al. [242][243] have evaluated the impact of flickering on the video acceptance by the viewer on mobile devices. They have investigated the effects of changing video qualities (noise flicker), video resolutions (blur flicker), and frame rates (motion flickering) for SVC at various configurations with periodic flickering durations. Periodic flickering means that a switch from the higher to the lower representation, and vice versa, occurred periodically, e.g., every 2 seconds. Their results show that frequent noise flickering between two SNR representations with a period below 2 seconds impairs the viewing quality down to a point where viewers would prefer the lower video representation altogether. For blur flickering, viewers preferred the constant lower representation (at half the original resolution) even compared to longer flickering periods.

When no flickering was involved, participants rated the lower representation of videos with a deltaQP of 8 (i.e., enhancement layer at QP=24, base layer at QP=32) to have roughly the same viewing quality as the lower representation of dyadically downscaled videos (i.e., enhancement layer at 480x320, base layer at 240x160). When flickering was introduced, the viewing quality for blur flickering got rated lower than for noise flickering. These observations underpin our statement from Section 3.5.1 that adaptive video streaming sessions should maintain the same resolution and should only switch between quality layers.

Mok et al. [244] have proposed a QoE-aware DASH system based on AVC. As quality switches of high amplitude (e.g., from highest to lowest representation) are annoying to viewers, the proposed adaptation algorithm inserts intermediate steps to avoid abrupt quality changes. Thus, the reduced amplitude of quality switches seems to outweigh the additional number of quality switches in terms of QoE. This also confirms an earlier study on quality switches by Zink et al. [245] that has evaluated viewers' preferences of various quality switching patterns. General trends in those patterns are that high amplitudes in down-switches should be avoided and that switching up is preferred to switching down (i.e., it is better to start with a low quality and switch up than to start with a high quality and switch down).

Furthermore, Ni et al. have proposed the concept of frequent layer switching (FLS) [246][127]. Their study compares the QoE of a playback that frequently switches between two representations (with either temporal or quality scalability) to the playback of only the lower representation. The study was conducted on mobile devices and HD screens. The motivation of FLS is that the required bandwidth for adaptive streaming could be strategically reduced through layer switches without impacting the QoE. The results show that for mobile devices, temporal switching between 25 and 12 fps (at a quality enhancement layer encoded with QP=28) was preferred over constant playback at only the quality base layer (encoded with QP=36) at full frame rate. Viewers had no clear preference between temporal switching and constant playback at the lower frame rate. For HD screens, the results were less conclusive. For both scenarios, high switching frequencies induced undesirable flickering effects.

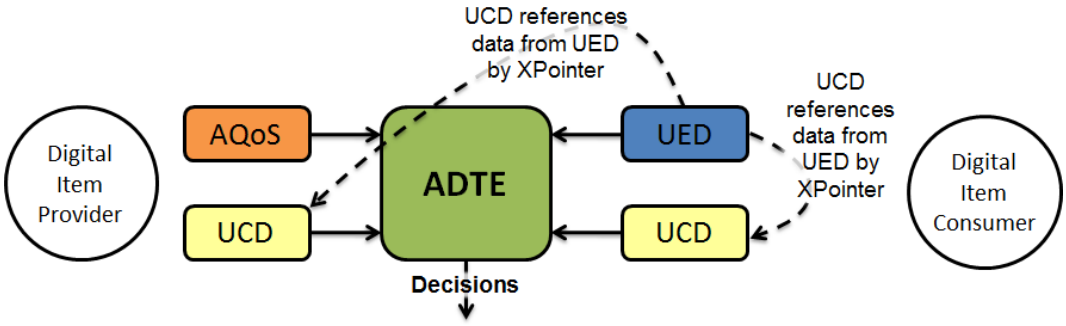


Figure 63: MPEG-21 Digital Item Adaptation architecture, adopted from [248].

In a recent study, Sieber et al. [247] have proposed an SVC adaptation logic that reduces the number of quality switches by striving for a stable buffer level before increasing the number of consumed SVC layers. Their evaluations show a very high and stable overall playback quality of the proposed algorithm compared to other state-of-the-art SVC-DASH adaptation techniques. However, the comparison does not take the amplitude of quality switches into account.

5.4.1.3 Standardization

In the context of the MPEG-21 standards family towards a holistic multimedia framework, MPEG has standardized a multimedia adaptation framework, called MPEG-21 Digital Item Adaptation (DIA) [248]. It specifies a general architecture for adaptation systems and a representation format for adaptation parameters and adaptation logics. The architecture of the MPEG-21 ADTF is depicted in Figure 63. MPEG-21 DIA specifies XML-based formats for the Usage Environment Description (UED), Universal Constraints Description (UCD), and Adaptation QoS (AQoS). The UED describes the current conditions of the system and comprises user characteristics, terminal capabilities, network characteristics, and natural environment characteristics. The UCD allows for the expression of limitation constraints (e.g., a lower bound for the resolution) and optimization constraints (e.g., to maximize the frame rate) for the adaptation. Feasible types of adaptation, their associated utilities, and relationships between constraints are described by the AQoS [8]. In other words, the adaptation parameters are represented by the UED, while the adaptation logic is represented in the UCD and AQoS. For further details, the interested reader is referred to [249].

Recently, MPEG has standardized DASH, formally known as ISO/IEC 23009-1 [250]. Within DASH, multiple representations of the same content (e.g., different bitrates or resolutions) are split into temporal segments that the client can request (cf. Section 3.5.1). Representations and segments are listed in an MPD that is retrieved prior to streaming. The adaptation logic is thus shifted into the client that decides on the representation to be downloaded for a given segment. Note that DASH does not specify the adaptation decision-taking process. It rather provides the means for arbitrary client implementations to retrieve and adapt content from a conventional

HTTP server. Thus, it does not make MPEG-21 DIA obsolete; on the contrary, DIA can be integrated into a DASH client for performing adaptation decisions.

5.4.1.4 Conclusions

Based on the surveyed related work, a holistic adaptation framework should:

- properly encode the content to SVC according to layer configuration recommendations devised in Chapter 3,
- prioritize NALUs (i.e., set NALU priority IDs) to allow high flexibility in adaptation [70],
- allocate rates for multi-video streaming [229][230][193],
- adapt either at the client (for HTTP/TCP-based streaming [61]) or in a distributed manner at the network edges and inside the network (for RTP/UDP-based streaming),
- dynamically adapt content, taking network status monitoring, expected QoE based on video quality indications of SVC layers [70][71][67][214], buffer level, influence of quality switches (considering switching amplitude and frequency as well as switching patterns) [242][247][245] or temporal quality variations in general [226][227][228], quality impact of initial playback delay [239], risk and impact of playback stalling (or packet loss for RTP/UDP-based streaming [211][223][224]), and user context (e.g., device type, ambient illumination [234], use of sensory effects [235], etc.) into account, while being aware of caching strategies at the network layer [59][60],
- constantly monitor (or at least estimate) the QoE of the adapted stream to readjust configuration parameters of the ADTE [211].

A QoE model enabling such holistic adaptation logic would come with a plethora of intertwined configuration options. Incorporating all these aspects into the adaptation logic would clearly exceed the scope of this thesis. Rather, we focus on integrating our SVC encoding guidelines with an end-to-end distributed adaptation chain in a real-life streaming prototype setup. From the points listed above, our adaptation chain considers proper SVC encoding, adaptation at client side and in the network, and dynamic content adaptation based on the monitored network status.

5.4.2 Adaptation Logic

This section describes the adaptation logic deployed for RTP streaming in the end-to-end distributed adaptation chain test-bed setup of the ALICANTE architecture. It has to be noted that this adaptation logic itself is not intended to extend the state of the art of SVC adaptation discussed in Section 5.4.1. It rather demonstrates the

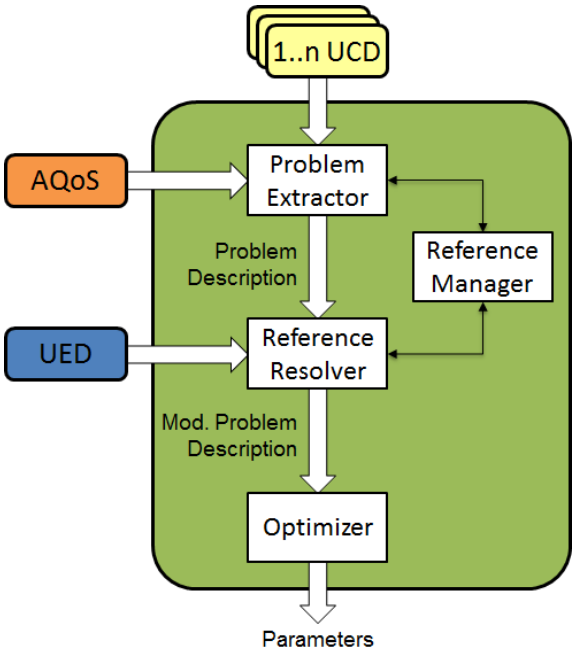


Figure 64: Architecture of the ADTE, adopted from [251].

integration of adaptation within the network (at MANEs) and on the network edges (at Home-Boxes) using an MPEG-21 DIA standard-conforming ADTE.

The ADTE depicted in Figure 64 is the core engine of the ADTF and is typically divided into four building blocks: *Problem Extractor*, *Reference Manager*, *Reference Resolver*, and *Optimizer*. The *Problem Extractor* processes AQoS descriptions and several UCDs and generates an internal mathematical representation of the problem. The *Reference Manager* handles the references in the problem description by completing them with values from the Reference Resolver. The *Reference Resolver* extracts those values from the UED. The Reference Resolver also checks whether all references can be resolved and modifies the problem description accordingly. Finally, the *Optimizer* computes a solution for the modified optimization problem and yields the parameters for adaptation [251][3].

In the following, we document the limitation and optimization constraints of the deployed adaptation logic.

The limitation constraints related to the spatial resolution are given in Equations (3) and (4).

$$HRes_{disp} \geq hRes_{sel\ layer} \geq HRes_{min} \qquad (3)$$

$$VRes_{disp} \geq vRes_{sel\ layer} \geq VRes_{min} \qquad (4)$$

Variables $hRes_{sel\ layer}$ and $vRes_{sel\ layer}$ represent the horizontal and vertical resolution of the selected SVC layer, respectively. The horizontal display resolution

$HRes_{disp}$ and vertical display resolution $VRes_{disp}$ form the upper bounds for the content resolution. The minimum guarantees for the horizontal resolution $HRes_{min}$ and the vertical resolution $VRes_{min}$ specified in the SLA form the lower bounds for the content resolution.

Without loss of generality, we assume that the content is encoded into a single SVC bitstream. Therefore, we speak of SVC layers in this section, even though the adaptation logic can be applied to encodings with multiple SVC bitstreams (cf. Chapter 3) or other coding formats (e.g., AVC) as well.

The limitation constraint related to media bitrate is given in Equation (5).

$$s_i \cdot C_{link} \geq br_{sel\ layer} \geq Bw_{min} \qquad (5)$$

The variable $br_{sel\ layer}$ represents the bitrate of the selected SVC layer. The link capacity is denoted as C_{link}. Multiplied with the maximum bandwidth share s_i of the stream, it forms the upper bound for the content bitrate. The minimum guaranteed bandwidth Bw_{min} specified in the SLA forms the lower bound for the content bitrate.

The bandwidth shares s_i are divided equally between all streams currently being handled by the Home-Box (including cross traffic). The weighting of bandwidth shares can be refined by traffic classification, with each class having a different priority. Within the ALICANTE framework, the Service Priority can be signaled in the Content-Aware Transport Information (CATI) as discussed in [252], the distribution of bandwidth among those classes is described in the SLA. Let n_i be the number of streams in traffic class i. The calculation of bandwidth share s_i for a stream in traffic class i is given in Equation (6).

$$s_i = \frac{w_i}{\sum_i (n_i \cdot w_i)} \qquad (6)$$

Let w_i be the weight assigned to traffic class i based on its Service Priority, such that $\sum_i w_i = 1$.

Since each traffic class may comprise a different number of streams, the equation compensates for w_i by the sum of all weights of all streams in all traffic classes. Thus, every stream gets a fair share of the overall bandwidth based on the weight of its traffic class and the number of streams. Note that the rate allocation could be further improved by taking the RD performances of different streams into account as discussed in Section 5.3.3.1.

In order to enable adaptation decisions based on the measured packet loss, the adaptation logic performs the following estimation. The monitored bitrate and packet loss are used to estimate the upper bound for the bitrate of the stream according to

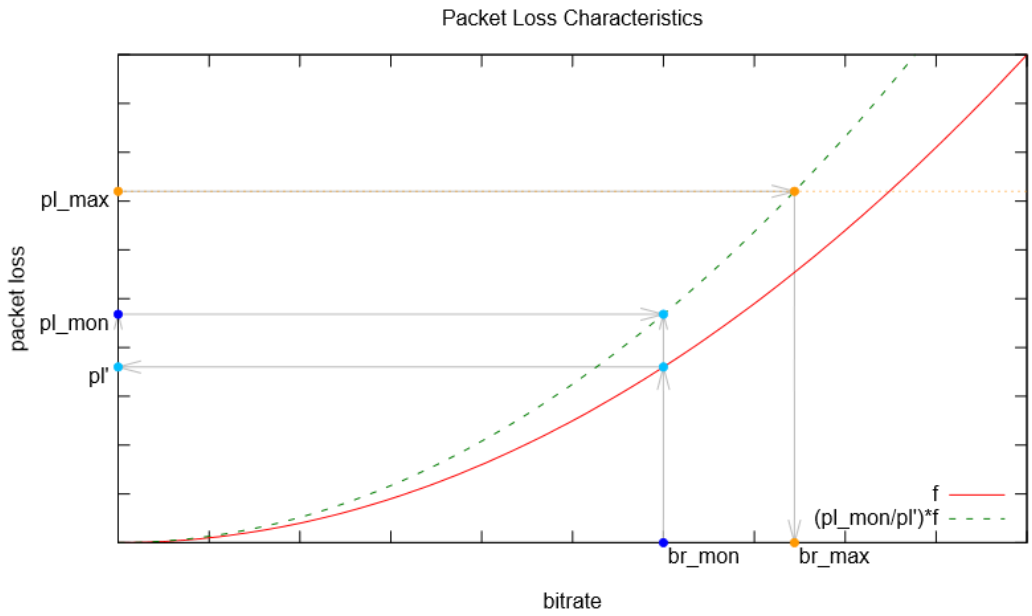


Figure 65: Illustration of estimation of maximum bitrate based on packet loss characteristics.

the packet loss requirements of the SLA. Let $f : BW \rightarrow PL$ be the packet dropping probability as a function of the bandwidth utilization. This mapping can be specified by the configuration of the congestion avoidance algorithm, e.g., Generalized Random Early Detection (GRED). A more accurate mapping is obtained by monitoring the packet dropping characteristics of the network. The monitored bitrate br_{mon} of the stream is used to estimate the packet loss as $pl' = f(br_{mon})$. This estimated packet loss is adjusted by the actual monitored packet loss pl_{mon} and the adjusted mapping $\frac{pl_{mon}}{pl'} \cdot f$ is used to determine the highest bitrate that will not violate the maximum packet loss pl_{max} stated in the SLA. The estimation process is illustrated in Figure 65. The resulting limitation constraint is shown in Equation (7).

$$\frac{pl'}{pl_{mon}} \cdot f^{-1}(pl_{max}) \geq br_{sel\ layer}\,, \quad \text{with } pl' = f(br_{mon}) \qquad (7)$$

The goal of this limitation constraint is to prevent higher packet dropping rates from the congestion avoidance algorithm by proactively switching to a lower layer.

The adaptation logic could be further refined by taking the quality degradation introduced by packet loss into account as described in Section 5.2.2.3. Note that such an approach also requires the initial quality of each layer to be signaled, typically in the media stream itself. For SVC, this information could be signaled in a SEI message by using a *user data unregistered SEI message* [23]. With the initial quality information and the quality degradation characteristics, the layer with the highest estimated QoE could be selected based on a PSQA model [223][224].

The optimization constraints of the adaptation logic are given in Equations (8), (9), and (10).

$$\max\left(hRes_{sel\ layer}\right) \quad (8)$$

$$\max\left(vRes_{sel\ layer}\right) \quad (9)$$

$$\max\left(l_{sel\ layer}\right) \quad (10)$$

The variable $l_{sel\ layer}$ represents the layer number of the selected SVC layer. The optimization constraints are subject to the limitation constraints of Equations (3), (4), (5), and (7). The deployed implementation of the MPEG-21 ADTE uses a simple generate&test approach with priority sorting of optimization constraints as discussed in [251]. Due to the priority sorting, the maximization of the horizontal resolution has precedence over the vertical resolution and the SVC layer number.

The described adaptation logic prefers quality switches over resolution switches due to this priority sorting. That is, the maximization constraints for mainting a high resolution have precedence over the maximization of the layer number. Note that it does not consider temporal layers, except implicitly through the assignment of SVC layer numbers. A quality model for the best extraction path as described in [67] could be used to refine the adaptation logic for temporal adaptation. However, Section 3.2.2 has shown that industrial streaming solutions typically recommend only a single frame rate (24-30 fps). Thus, we argue that temporal scalability can be neglected in our test-bed setup.

The adaptation logic does currently also not take the impact of layer switches on the QoE into account. As shown in [242] and [243], frequent quality switches impair the QoE. Thus, we configured the ADTE to update its adaptation decision at most every 2 seconds. The integration of a QoE estimation technique that considers the impact of quality switches would require the quality of each layer to be signaled in SEI messages as described above, which is subject to future work.

The implementation of the adaptation logic for the MPEG-21 ADTE of [251] is provided in Annex G.

For DASH adaptation, we deploy the adaptation logic by Müller et al. [238][61]. The algorithm uses the bandwidth measured from the previous downloaded segment to estimate the maximum available bandwidth for the current segment. If the buffer level gets below 35% or above 50%, the bandwidth estimate is adjusted accordingly. This adaptation logic is built into the *libdash* [253] client implementation of the DASH standard.

Section 5.5 will present and validate the discussed adaptation logic. Before that, we introduce the concept of smoothing the transition between representations to reduce the flicker that is caused by switching between representations.

5.4.3 Smooth Transition between Representations

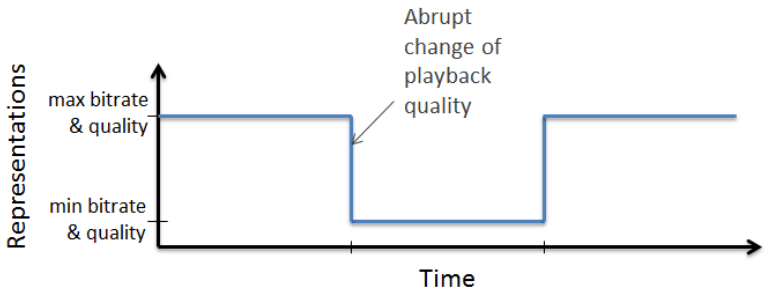
5.4.3.1 Introduction and Concept

Frequent quality switches with high amplitudes during adaptive HTTP streaming sessions – e.g., switching from (very) high to (very) low bitrates – have been shown to annoy viewers and, thus, reduce the QoE [242]. The disturbance can be reduced through intermediate quality levels [244] but in practice only very few levels (3-5) are deployed. Previous work focused only on quality switches at segment boundaries and viewers may still notice abrupt quality changes.

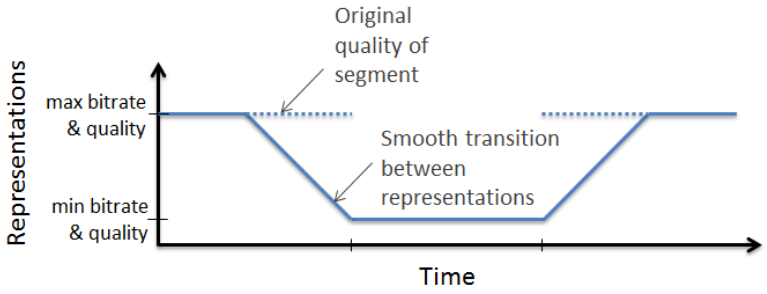
We propose an even more fine-grained approach, a smooth transition between representations, which we subsequently call *representation switch smoothing*. The goal of *representation switch smoothing* is to reduce the annoyance of quality switches. When the receiver is aware of an imminent switch to a lower representation, it can already reduce the playout quality of the current or the subsequent representation, enabling a smooth transition between the two representations. Frame by frame, the playout quality is slightly reduced. Vice versa, the playout quality can be smoothly increased when a higher representation is received. The concept is illustrated in Figure 66.

In client-driven streaming scenarios such as DASH, the adaptation decision is typically known at least one segment duration ahead of playout time. While the current segment is played, the next segment has to be requested to ensure timely arrival. For SVC-DASH, the time frame might be shorter, depending on whether enhancement layers of the segment are downloaded using HTTP pipelining [206][61]. Typical DASH clients already decide to adapt to a lower representation when still three or more 2-second segments are buffered [61]. If the adaptation logic pursues a conservative buffer management (e.g., [247]), the adaptation decision is taken even further ahead. In any case, the receiver is aware of pending representation switches ahead of playout time and can thus react by smoothing the quality transition.

Representation switch smoothing can be realized by an additional component in the decoding chain. This component is notified by the client's adaptation logic whenever the adaptation decision is changed. The amplitude of the switch has to be signaled as well. For SVC with MGS layers, this can be represented as the difference in MGS layers. In a more general system, the bitrates or the video qualities (e.g., PSNR) of the higher and lower representation may be signaled. If the first frame of the lower representation can already be decoded, its quality could be used by the representation switch smoothing component as reference to adjust the amount of



(a)



(b)

Figure 66: Adaptation with (a) traditional representation switching and (b) *representation switch smoothing* [14].

noise it adds to frames of the higher representation. Depending on the amplitude of the representation switch, the smoothing component chooses the duration of the transition; higher amplitudes require longer durations.

In case of down-switching, the component adds increasing noise to the frames of the higher representation as detailed in Section 5.4.3.2 until it matches the quality of the lower representation just before the switching. In case of up-switching, the component adds noise with temporally decreasing intensity to the frames of the higher representation, such that the transition between representations becomes seamless.

5.4.3.2 Implementation Options

There are three options for implementing the smooth reduction of quality: either before, within, or after the decoder. As smoothing for up-switching is performed analogously to down-switching, we only consider down-switching in the following discussion.

The first option, denoted *pre-decoder* implementation, is to add a filter component before the decoder. This component alters the encoded bitstream by removing certain picture fidelity data. For SVC with MGS enhancement layers, a straight-forward implementation is to remove transform coefficients (i.e., set them to 0) from the enhancement layer. For the i^{th} frame in the smooth transition, n transform coefficients are removed as calculated in Equation (11).

$$\mathrm{n} = \left\lfloor \frac{i \cdot \Delta C}{d} \right\rfloor \qquad (11)$$

Let d be the duration of the smooth transition in frames and ΔC be the total difference of the number of transform coefficients between the higher and the lower representation.

This approach is easy to implement and independent of the decoder. However, a drawback is that changes from one frame are propagated within the GOP due to motion compensation drift [27], causing unwanted artifacts.

The second option is an implementation inside the decoder (i.e., *in-decoder* implementation). Again, some picture fidelity data is removed from the coded frames, but without affecting the motion compensation of other frames. For SVC, this implies that the inverse transform of the residual data has to be performed twice. The number of transform coefficients to be removed per frame is the same as in the first implementation option. A simplified block diagram of the decoding process is given in Figure 67.

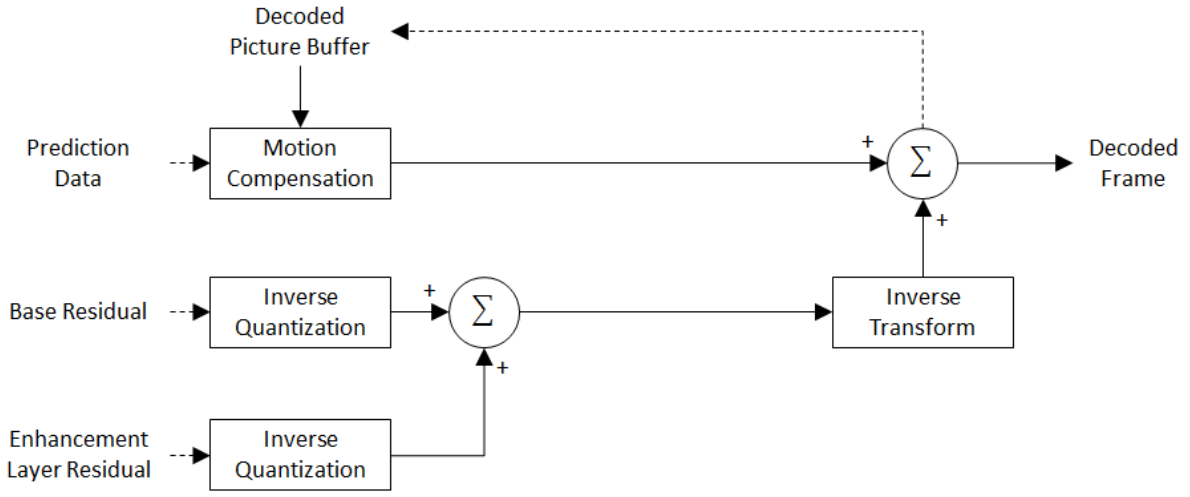
Figure 67 (a) shows the original SVC decoder structure adopted from [158] with handling of base layer and enhancement layer residual data. Figure 67 (b) highlights the additional steps necessary for maintaining the original decoded picture buffer when performing representation switch smoothing. In contrast to the first implementation option, representation switch smoothing is performed after the inverse quantization. The operations are commutative; setting a transform coefficient to 0 has the same result before and after inverse quantization.

Since motion compensation is still based on the original, unimpaired coded video data, we expect the reconstructed frame to be slightly different from the case where the respective transform coefficients had been set to 0 in the encoding process. The assessment of the resulting video quality is subject to future work.

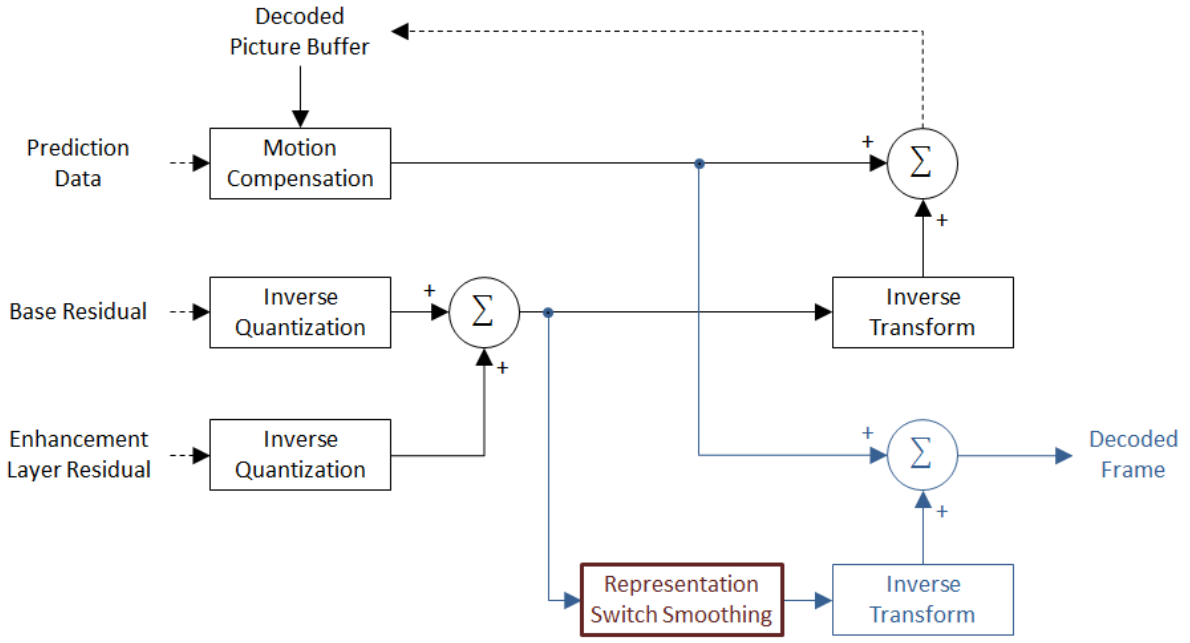
Nevertheless, an implementation within the decoder is more accurate and robust than the first implementation option since it avoids error propagation. Of course, it requires a specialized decoder, which might limit interoperability (cf. Section 3.3.2).

Note that the first two implementation options will have to consider that SVC allows for custom scaling matrices, which even may change between frames. The scaling matrix provides the values by which the transform coefficients of a macroblock are inversely quantized. Full support for custom scaling matrices might increase the computational complexity of the implementation.

The third implementation option is to add a video filter component after the decoder for inserting additional noise into the decoded frames. We denote this as *post-decoder* implementation. This noise mimics the degrading quality to enable a smooth transition to the lower representation. As AVC (and SVC) deploys a low-complexity integer transform [24], this can be achieved with low computational overhead.



(a)



(b)

Figure 67: Simplified block diagram of the SVC decoding process for (a) traditional decoding, adopted from [158] and (b) decoding with representation switch smoothing [14].

Nevertheless, the computational complexity is still slightly higher than for the first two implementation options. That is, in the pre-decoder implementation option, the transform coefficients can be altered directly in the bitstream (after reverting the entropy coding); in the in-decoder implementation option, the inverse transformation has to be duplicated; in the post-decoder implementation option, transformation into the frequency domain also has to be added.

This third implementation option is independent of the decoder and the video coding format and also avoids drift.

Table 17: Characteristics of representation switch smoothing component implementation options.

	Implementation Option		
	Pre-decoder Option	**In-decoder Option**	**Post-decoder Option**
Decoder dependent	no	yes	no
Coding format dependent	yes	yes	no
Drift	yes	no	no
Computational complexity	very low	very low	low

While the third implementation option is video coding format independent, it has to know the extent to which the quality changes with the representation switch, and, subsequently, how the new quality can be approximated by the synthetic distortion. Such a general model for video quality approximation remains an open research challenge.

The properties of the three implementation options are summarized in Table 17.

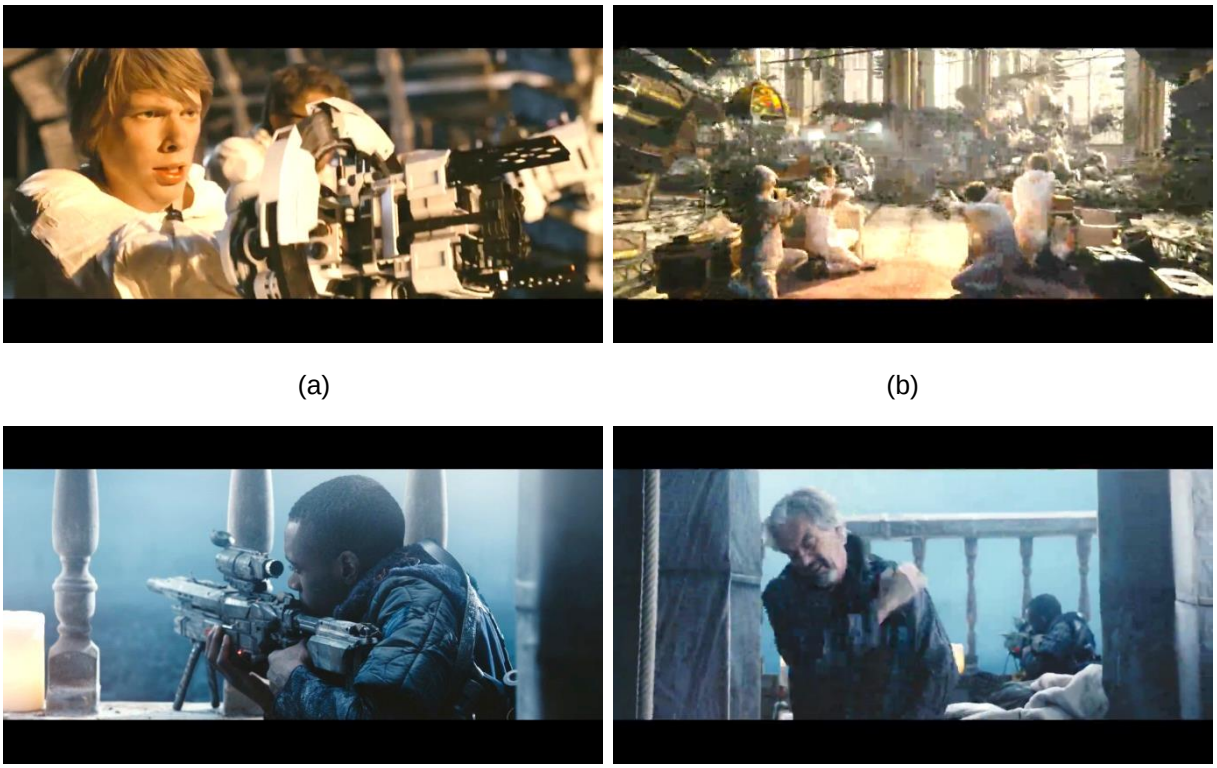
5.4.3.3 Evaluation

We have performed an initial evaluation of the *representation switch smoothing* approach for down-switching scenarios through subjective tests. As up-switching might be perceived differently from down-switching, e.g., viewers might experience a sudden increase in video quality as a positive event, the combination of both up- and down-switching in a single test sequence could bias the results. Thus, we only considered down-switching in our first evaluation in order to decide whether the approach is worth pursuing.

We used two test sequences, both extracted from the open-content short film *Tears of Steel* [254]. Both sequences have durations of 15 seconds at resolution 1280x720 and 24 fps frame rate. *Sequence 1* has high-motion content and was extracted starting at time point 7:43; *Sequence 2*, with low-motion content, was extracted starting at 1:57. The sequences were selected such as to avoid confusing scene changes, although both contain cuts.

The 15-second sequences were split into 5-second segments. We simulated a quality switch from a high bitrate (2,000 kbps) to a low bitrate (400 kbps for *Sequence 1* and 250 kbps for *Sequence 2*) after 10 seconds. As *Sequence 1* has higher temporal information, it was harder to compress for the encoder, causing already strong visible artifacts at 400 kbps. Screenshots of the high and low bitrate encodings of the sequences are shown in Figure 68.

Each sequence was encoded once with a quality switch (after 10 seconds), and once with a smooth downward transition (between seconds 5 and 10). For the purpose of this test, the sequences were encoded to AVC at constant target bitrates with the FFmpeg encoder.



(a) (b)

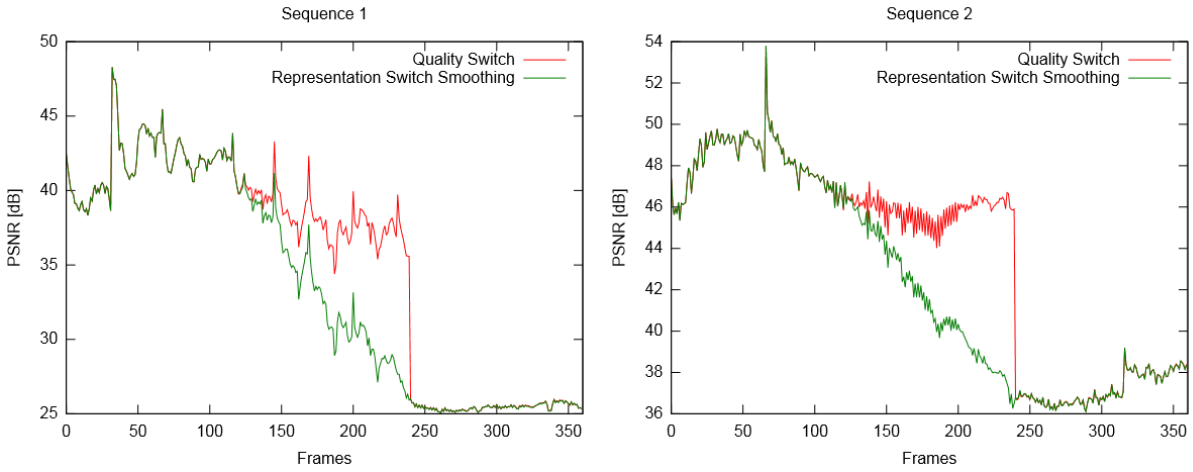
(c) (d)

Figure 68: Snapshots of test sequences; (a) *Sequence 1* at 2,000 kbps, (b) *Sequence 1* at 400 kbps, (c) *Sequence 2* at 2,000 kbps, and (d) *Sequence 2* at 250 kbps [14].

We observed that the encoder badly allocates bitrates for the first few frames, especially at low target bitrates. In per-segment encoding, this caused unwanted distortion at segment boundaries. We thus decided to always encode the entire sequences and to split them into segments after encoding. In the absence of a working implementation of any of the aforementioned options, the smooth transition was realized by encoding the sequences at predetermined target bitrates (one per frame in the transition segment) and stitching the respective frames to a continuous segment. Thus, 120 encodings were used to obtain the 5-second transition.

The bitrates for the smooth transition were determined as follows. The sequence was first encoded at 5 sample bitrates (from 2,000 kbps to the lowest bitrate). The PSNR for the transition segment was calculated to obtain the rate-distortion performance. As the RD performance typically follows a logarithmic curve, a logarithmic curve fitting $f(br)$ was computed as shown in Equation (12) in order to approximate the video quality $q \approx f(br)$ for bitrate br and model parameters a and b.

$$f(br) = a \cdot \ln(br) + b \qquad (12)$$



(a) (b)

Figure 69: Per-frame PSNR results for quality switching and representation switch smoothing for (a) Sequence 1 and (b) Sequence 2 [14].

The inverse function $f^{-1}(q)$ of this curve fitting is shown in Equation (13).

$$f^{-1}(q) = e^{\left(\frac{-b}{a}\right)} \cdot e^{\left(\frac{q}{a}\right)} \qquad (13)$$

Based on this inverse function, the 120 bitrates were calculated that predicted a linear decrease of PSNR over the entire transition duration. The per-frame PSNR results for both versions are shown in Figure 69 for the two test sequences.

One drawback of the applied solution is that the encoder uses different blocks for motion (and intra-) prediction at each bitrate. With low bitrates, blocking artifacts become increasingly visible. Due to the different predictions, the positions of the blocking artifacts change randomly for the extracted frame of each respective bitrate. When stitching the frames from these encodings, this causes some temporal noise. This noise is particularly visible in low-motion areas of the picture. In contrast, the low-bitrate segment at the end of a sequence has blocking artifacts that continuously move through the scene. So, even though the blocking artifacts are clearly visible, their movements correlate with the actual motions in the scene. Due to the temporal noise in the transition, the actual visual quality might be lower than what is reported by PSNR. As this effect was only recognized after time-consuming encoding of the transition segments, and due to the lack of a more accurate short-term solution, the subjective tests were performed with the described transition segments. This means that *representation switch smoothing* based on one of the implementation options discussed in Section 5.4.3.2 may even provide better results than our evaluation.

The subjective tests were performed with 18 participants (13 male, 5 female) of age 23 to 45. The participants were told that the test concerned changes in video quality. No further indication as to the nature of the quality changes was given. The participants were presented with the two versions of each sequence (labeled

Table 18: Subjective test results for evaluation of *representation switch smoothing* [14].

Preferred Version / Sequence	Quality Switching	Representation Switch Smoothing	No Difference
Sequence 1	5	**7**	6
Sequence 2	3	**12**	3

Version a and *Version b*). One version contained the quality switch, the other the smooth transition. The attribution to *Version a* and *Version b* was changed between the two sequences (i.e., *representation switch smoothing* was shown in *Version b* of *Sequence 1* and in *Version a* of *Sequence 2*). The participants were instructed that they may start with either version and may watch each version as often as they wanted. The videos were shown in full-screen mode on a Dell 1907FPc LCD monitor having a native display resolution of 1280x1024. The videos were shown without audio. The participants were asked to rate whether they preferred *Version a*, *Version b*, or saw no difference. The questionnaire is given in Annex F.

The results of the subjective tests are provided in Table 18. We performed the Kruskal Wallis test [255] for both sequences to test for signifance of our results. The Kruskal Wallis test is the non-parametric counterpart of the one-way analysis of variance. For *Sequence 1*, the ρ-value is 0.8479 ($H = 0.33$), which means that the null hypothesis (i.e., viewers voting equally often for each of the three samples, thus being generally indifferent towards the transition technique) cannot be rejected. For *Sequence 2*, the ρ-value is 0.012 ($H = 8.84$), which means that the null hypothesis has to be rejected for $\alpha = 0.05$.

Representation switch smoothing performed significantly better for Sequence 2 than for Sequence 1. Several participants reported that the high motion of Sequence 1 made the two versions look indifferent. Many participants viewed each version at least two or three times before making a decision. There were no significant differences in the test results between male and female participants, although male participants tended to prefer *representation switch smoothing* slightly more than female participants. While the overall results show only a slight preference towards representation switch smoothing, we argue that further tests should be conducted, investigating the effects of smooth transitions on various configurations. Note also that the aforementioned temporal noise in the smooth transitions may have affected the test results.

For future subjective tests, the following evaluations should be performed. Main influence factors to test are the amplitude of the quality switch (e.g., measured as the bitrate difference), the duration of the smooth transition, as well as the amount of Spatial and Temporal Information. Based on our experiences and feedback from test participants, we assume representation switch smoothing to achieve the highest gain for scenes with high Spatial and low Temporal Information. Furthermore, we speculate that longer transition durations (e.g., 10 seconds) will better mask the quality changes.

Other possible influence factors that we identified in our evaluations are the base quality (in contrast to just the bitrate differences), the presence of cuts, the resolution, and the duration for which only low quality segments are available (e.g., only 2 seconds of low quality might not justify two 10-second transitions). It has also to be investigated whether smooth transitions are also useful for up-switching scenarios. As evaluated in [245], viewers prefer to watch low-quality segments followed by high-quality segments rather than the other way around. Thus, we infer that up-switching is perceived to be less annoying than down-switching. Furthermore, Seshadrinathan and Bovik [227] have reported that viewers give poor quality ratings to sharp video quality drops but do not increase ratings as eagerly when the video quality resumes to its previous high state. From those results, we reason that up-switching is noticed less than down-switching. These two effects may diminish the benefits of a smooth transition for up-switching.

For test content generation, the aforementioned temporal noise should be avoided by implementing one of the suggested implementation options from Section 5.4.3.2. Instead of allowing participants to watch versions as often as they like, the test material could contain around 3-5 quality switches and be shown only once to create the same conditions for all participants. Additionally, a 5-point Likert scale could be used to better distinguish preferences between the tested versions.

5.4.3.4 Conclusions

In this section, we have introduced the concept of representation switch smoothing. The approach avoids abrupt quality switches by smoothly reducing the video quality on a per-frame basis. This avoids unnecessary viewer distraction in adaptive HTTP streaming. We have discussed three implementation options for the smoothing component in an SVC-based DASH system.

While down-switching is generally considered annoying, abrupt up-switching might even increase the QoE as viewers might be happy to notice visual improvements in the video quality. It has to be evaluated whether representation switch smoothing is beneficial for up-switching at all.

Our initial evaluations indicate a tendency towards the benefit of *representation switch smoothing* compared to hard quality switches. So far, we have only evaluated down-switching scenarios with very few configurations. Based on these evaluations, we have identified parameters and test methods for future subjective tests on the impact of representation switch smoothing on the QoE. Future work shall derive a model from these evaluations for configuring the duration of a quality transition against the amplitude of the representation switch.

5.5 Validation of End-to-End Adaptation System

This section documents the setup of the integrated end-to-end adaptation system that we used for functional and quantitative validation, and it provides performance evaluations of selected streaming scenarios.

5.5.1 Test-Bed Setup

In order to demonstrate and validate the adaptation capabilities of the ALICANTE streaming system, we created an integrated end-to-end streaming system setup. The setup supports SVC streaming via HTTP and RTP with distributed adaptation and SVC tunneling.

Figure 70 depicts the test-bed setup for SVC streaming via RTP multicast. The source content is encoded to SVC at the server side (denoted *HB1*, as Home-Boxes can also act as content servers) and then streamed via multicast. MANEs only forward SVC layers that are actually used by at least one connected terminal. Additionally, the ADTFs at the MANEs monitor the network conditions and react to congestion by reducing the number of forwarded SVC layers.

At the Home-Boxes, the received streams are handled by the ADTF. Figure 70 shows three different scenarios that demonstrate the adaptation/transcoding features of the Home-Boxes. The first terminal, connected to *HB2*, understands SVC and no transcoding or adaptation is necessary. Hence, the stream is forwarded to the SVC-capable player. The second terminal, a mobile device connected to *HB3*, does not support SVC but AVC. Hence, the SVC stream is transcoded to AVC with necessary adaptations (e.g., reduction of the resolution, bitrate) and then forwarded to the player. The third terminal player, connected to *HB4*, only supports MPEG-2. Hence, the SVC stream needs to be transcoded to AVC and afterwards, an additional transcoding to the supported codec (i.e., MPEG-2) via the General-Purpose Transcoder is necessary. The transcoded and adapted stream is then provided to the player.

For SVC streaming, we developed command line tools for the RTP server and client. The tools perform RTP multicast streaming of SVC in MST mode compliant to [202]. Streaming is performed in non-interleaved single-NALU packetization mode; the decoding order of NALUs is restored via CS-DON (cf. Section 5.3.3.1) at the client. When a packet of a fragmented NALU is lost, the client outputs all fragments of the NALU up to the lost packet. Any further packets of that NALU are discarded. The RTP streaming command line tools are made available as open-source software at [13].

As depicted earlier in Figure 60, the output of the RTP client module is piped into the SVC-to-AVC transcoder. A naïve first-in-first-out (FIFO) pipe can cause the RTP client to block due to a full pipe buffer. That is, if the transcoder cannot process the video fast enough, the buffer of the pipe is filled faster by the RTP client than it is

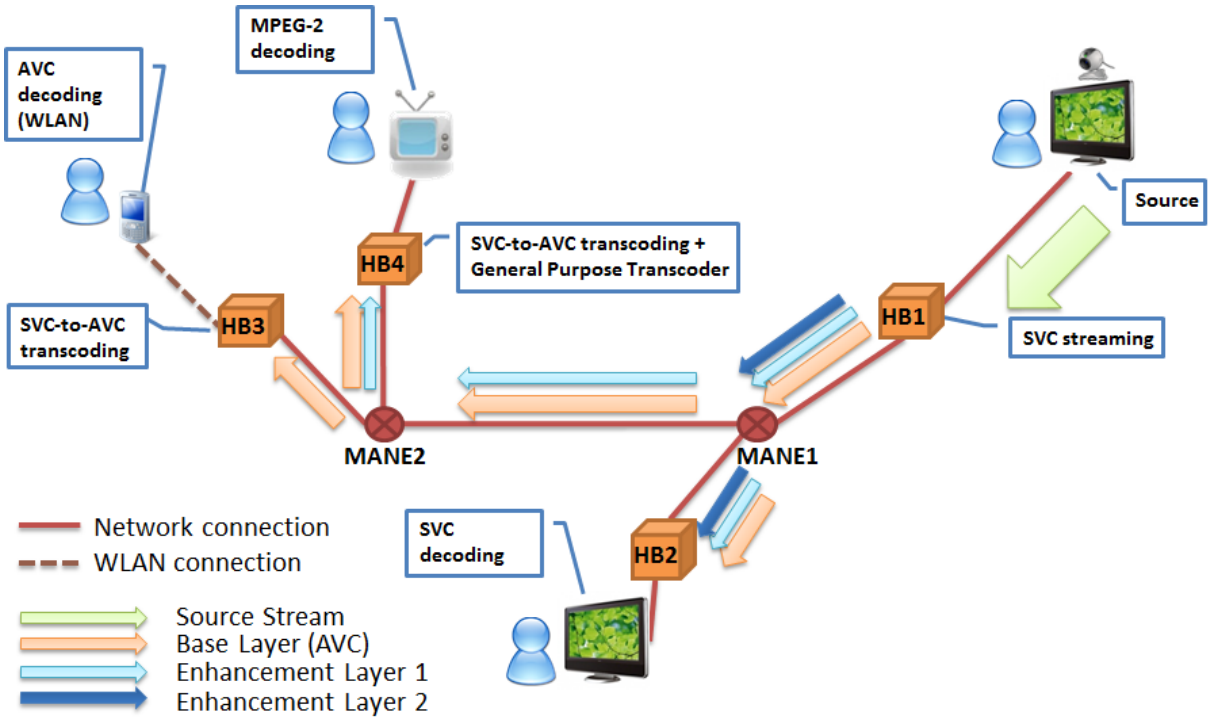


Figure 70: Test-bed setup for adaptive SVC multicast streaming, adopted from [9].

consumed by the transcoder. This causes the output of the RTP client to block until the transcoder has consumed more data. During this time, the RTP client would not be able to receive and process any RTP packets, resulting in high packet loss. To alleviate this problem, the RTP client module writes its output into a temporary file, from which the transcoder continuously consumes the video data.

For this prototype, Home-Boxes and MANEs are realized as virtual machines. The ADTFs at the Home-Boxes and MANEs deploy the adaptation logic described in Section 5.4.2. The SVC-to-AVC transform-domain transcoder was kindly provided by bSoft [103]. Consequently, we also used SVC bitstreams generated by the bSoft encoder to avoid compatibility issues between codecs (cf. Section 3.3.2). The SVC bitstreams were encoded following the encoding guidelines developed in Chapter 3.

The GPT is based on FFmpeg (cf. Section 5.3.3.1). The transcoded bitstreams were streamed via the LIVE555 Media Server [256] to the terminals. On the terminals, the VLC media player [257] was used for video retrieval and playback.

The demonstrator also supports SVC-DASH with a DASH proxy at the receiving Home-Box (cf. Section 5.3.3.2). As we reuse the adaptation logic implemented in libdash, we did not conduct performance evaluations of DASH. Note that due to time constraints, the representation switch smoothing approach described in Section 5.4.3 was not integrated in the test-bed setup.

A video of an early installation of the demonstrator (based on manual adaptation triggering) is provided at [258]. As of the time of writing, the updated video of the integrated demonstrator was still pending. The updated video will be provided on the ALICANTE blog [259] within the period of the project.

5.5.2 Evaluation

In addition to the functional validation of the end-to-end system, we also evaluated its performance in terms of end-to-end delay for streaming and quality impact of packet loss, distributed adaptation, and transcoding.

Due to time constraints, only few rudimentary tests are reported here. Further evaluations of the streaming system will be reported in the upcoming ALICANTE Deliverable D8.3 [260].

5.5.2.1 End-to-end Delay

For an evaluation of the end-to-end delay, the RTP streaming server was located on the same machine as the receiving Home-Box (*HB4*) in order to allow accurate timing measurements. The Wireshark [261] packet analyzer tool was used to capture the timing information of incoming and outgoing RTP packets.

Note that the LIVE555 Media Server used for streaming to the terminal uses RTSP for session control. This means that the VLC media player at the terminal will poll every few seconds for available content via RTSP DESCRIBE requests. As long as no content is available, LIVE555 simply acknowledges the request. Only after LIVE555 has received enough video data to parse the media parameters from the stream, it replies with a Session Description Protocol (SDP) [262] message. After receiving the session description, VLC sends RTSP SETUP and RTSP PLAY requests, whereupon LIVE555 starts the actual RTP session. These steps introduce additional delay, in particular the repeated polling via RTSP DESCRIBE requests from the client causes uncontrollable delay between polling operations. To alleviate this factor in our measurements, we implemented a rudimentary RTP unicast sender that simple sends the video data received from the input pipe.

Figure 71 sketches the adjusted test-bed with timekeeping with two MANEs along the network path. We measured the duration from sending an RTP packet of the SVC stream at the server to the reception of that packet at the Home-Box (denoted *time_network*) and the duration from that reception to the sending of the corresponding packet of the transcoded bitstream towards the terminal (denoted *time_adapt_**). While the entire video was streamed, timestamps were only measured for the first RTP packet conveying video content. For all other packets, the association between incoming and outgoing traffic would be much harder to establish due to the transcoding steps. In order to also measure the delay overhead introduced by the SVC-to-AVC transcoder and the GPT, we tested the Home-Box adaptation with mere restreaming (denoted *time_adapt_svc*), with just SVC-to-AVC transcoding (denoted *time_adapt_avc*), as well as with SVC-to-AVC transcoding plus GPT (denoted *time_adapat_mp2*).

Table 19 shows the delay measurement results. The results are average values from three runs. As the measurements from all three runs were consistent, they were considered sufficient to provide an overall picture of the introduced delays.

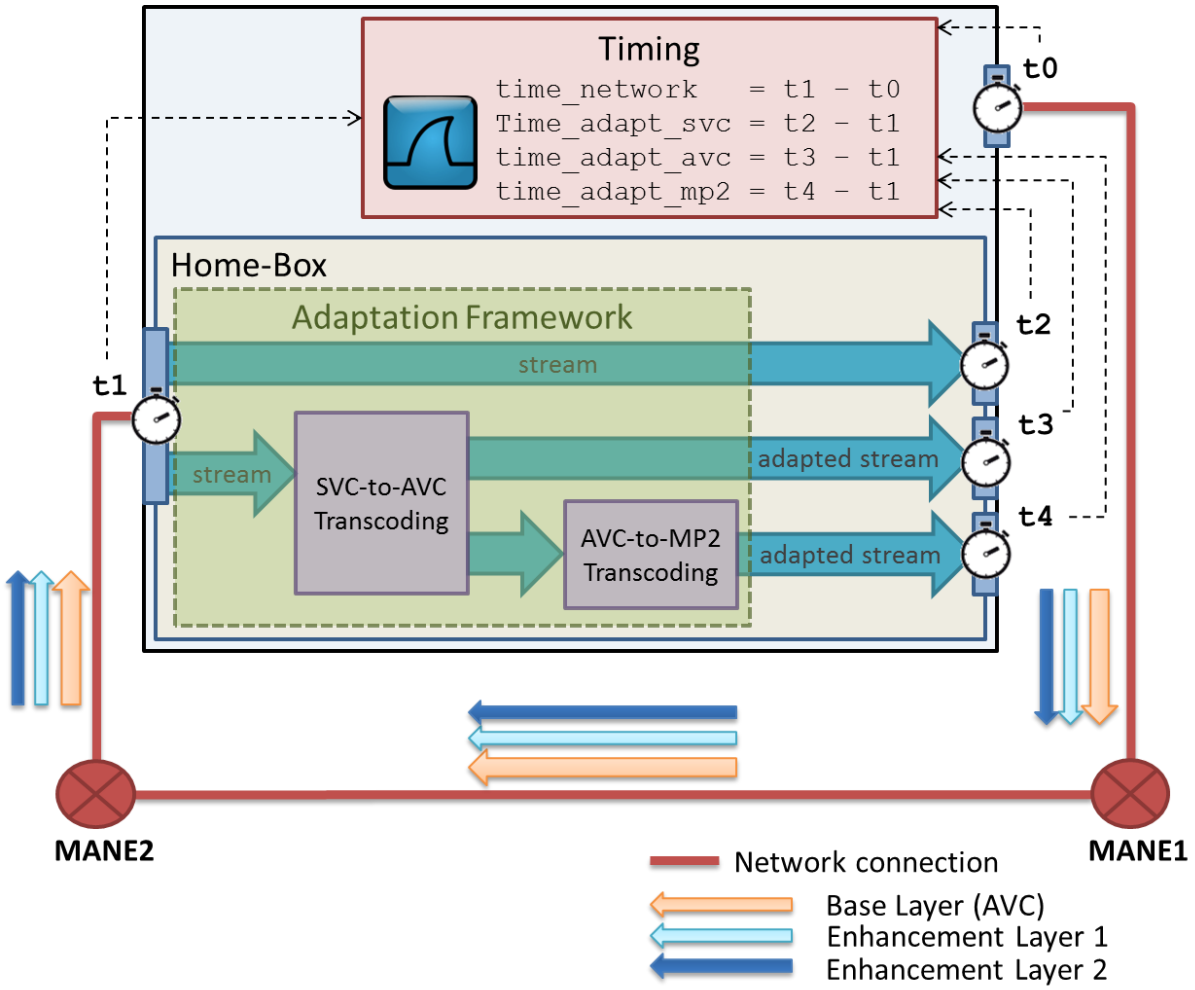


Figure 71: Illustration of delay measurement in the end-to-end streaming system.

The network delay (*time_network*) amounts to 0.17 seconds. This is attributed to the processing at the MANE prototypes. The MANEs perform several content-aware processing steps for RTP packets, such as deep packet inspection (DPI) to retrieve SVC layer information (cf. [263]). For mere ping requests, which are simply forwarded without any processing, the delay is as low as 0.0007 seconds.

For mere restreaming (*time_adapt_svc*), 0.07 seconds of delay are introduced. This is due to the fact that RTP packets are not simply forwarded, but instead handled by the RTP receiver, written into a FIFO pipe, read from the pipe by the RTP unicast sender and finally sent to the terminal.

The SVC-to-AVC transcoder introduces 9.88 seconds of delay, amounting to a total of 9.95 seconds (*time_adapt_avc*). At a 32-frames GOP size and a 25 fps frame rate of the test sequence, this cannot be explained by the structural delay of the video stream alone. Rather, we suspect that the SVC-to-AVC transcoder buffers the video either at the input or the ouput. We are in contact with the software developers and will report any improvements of this latency in the upcoming ALICANTE Deliverable D8.3 [260].

Table 19: End-to-end delay measurements.

Description	Average Timing [seconds]
time_network	0.17
time_adapt_svc	0.07
time_adapt_avc	9.95
time_adapt_mp2	10.31

If the FFmpeg-based GPT is appended, the delay (*time_adapt_mp2*) increases by 0.36 seconds to a total 10.31 seconds. Note that we had to reduce the *probesize* configuration option of FFmpeg, which controls how far FFmpeg peeks into the stream for obtaining media parameters before the start of transcoding. The delay introduced by FFmpeg is far below the GOP size of 32 frames (or conversely 1.28 seconds). Individual tests with frame-by-frame input confirmed the high latency of the bSoft SVC-to-AVC transcoder as compared to FFmpeg.

The end-to-end delay of the streaming system amounts to 10.48 seconds, 94% of which are attributed to the SVC-to-AVC transcoder implementation. If we assume that the latency of the SVC-to-AVC transcoder can be reduced to the same level as the FFmpeg AVC-to-MPEG-2 transcoder, the end-to-end delay could be reduced to below one second.

Note that the GOP structure of SVC bitstreams could be optimized to reduce the delay of the transcoding steps [27]. We deployed the bSoft encoder due to its support of fast SVC-to-AVC transcoding, which is necessary for an actual SVC streaming demonstrator. However, the bSoft encoder does not offer such low-latency encoding.

5.5.2.2 Video Quality Impact

Three factors influence the video quality in the end-to-end streaming system:

- *packet loss* may cause truncated or lost NALUs;
- *in-network adaptation* can help to reduce packet loss by discarding higher SVC layers;
- *transcoding* from SVC to other video coding formats degrades the video quality (cf. Chapter 3).

Extensive evaluations of the impact of packet loss on the video quality of SVC are provided in [215], [216], and [217]. We focus on a small set of tests to assess the impact of packet loss on SVC for the bSoft encoder/decoder. Note that we did not apply error protection schemes for the transmission.

We used the following test sequences: *Foreman*, *Container*, *Hall_Monitor*, *Stefan*. In order to enable reliable streaming measurements, each test sequence was repeated multiple times before encoding, to obtain 900 frames per test sequence. The test sequences were encoded with the bSoft encoder at resolution 352x288 with four

MGS layers with QPs to match the bitrate recommendations from Table 3 (i.e., QP=24 for the *Foreman* sequence, QP=19 for the *Container* sequence, QP=21 for the *Hall_Monitor* sequence, and QP=30 for the *Stefan* sequence). The frame rate was set to 25 fps; the I-frame period was set to 32 frames.

Due to fact that the Home-Box was deployed in a virtual machine, real-time transcoding was only supported up to a resolution of 352x288 (cf. Annex D). Thus, we did not include higher resolutions in our tests. The test scenarios for video quality evaluations in the end-to-end streaming system are illustrated in Figure 72.

During pre-assessment for the evaluation, the bSoft transcoder showed some discrepancies as compared to the bSoft decoder for the handling of packet loss. In particular, the bSoft transcoder introduced additional distortion in case of packet loss and also produced incorrect AVC bitstreams under high packet loss. Therefore, the following evaluations were performed based on the decoder. Note that the bSoft SVC-to-AVC transcoder also has a significant impact on the video quality as detailed in Annex H.

Packet loss can cause NALUs to be distorted to a degree at which the decoder has to discard the frame, or even entire frames can be lost. Thus, special care has to be taken to re-establish the temporal alignment with the original sequence for PSNR calculation. We collaborated with the software developers from bSoft in order to obtain logging information about discarded frames. Temporal adjustment was performed on the decoded YUV sequences. For each lost frame, we repeated the prior frame based on this information. Note that frames that are lost during transmission are never seen by the decoder. Therefore, we also used logging information from our RTP receiver module to identify frames lost in the network.

However, strong distortion in several consecutive frames caused the decoder to produce logging information of discarded frames that were inconsistent with the actual output. In such cases, the output was often completely mangled below any watchable quality. Typically the decoder was not able to recover from such states even after packet loss had decreased. In cases where the output had fewer frames than the original sequence even after our temporal adjustment, we repeated the last frame of the output until obtaining the full sequence length (i.e., 900 frames).

For all streaming tests, three test runs were performed for each test sequence. To account for possible errors in the temporal readjustment, the test run yielding the lowest PSNR was discarded for each sequence.

In an initial test, we streamed the sequences without any artificially induced packet loss. We disabled all adaptation tools, and captured the streamed SVC bitstreams on our test-bed setup. The scenario is depicted in Figure 72 (a). Even without any traffic limitations, small packet loss (typically around 0.2% - 0.9%) was observed in this scenario. Possible sources of error are losses due to packet processing at the MANE (cf. Section 5.5.2.1), the bridging of packets into the virtual machine on which the Home-Box was deployed, or our implementation of the RTP receiver module. As of the time of writing, we are still investigating this issue.

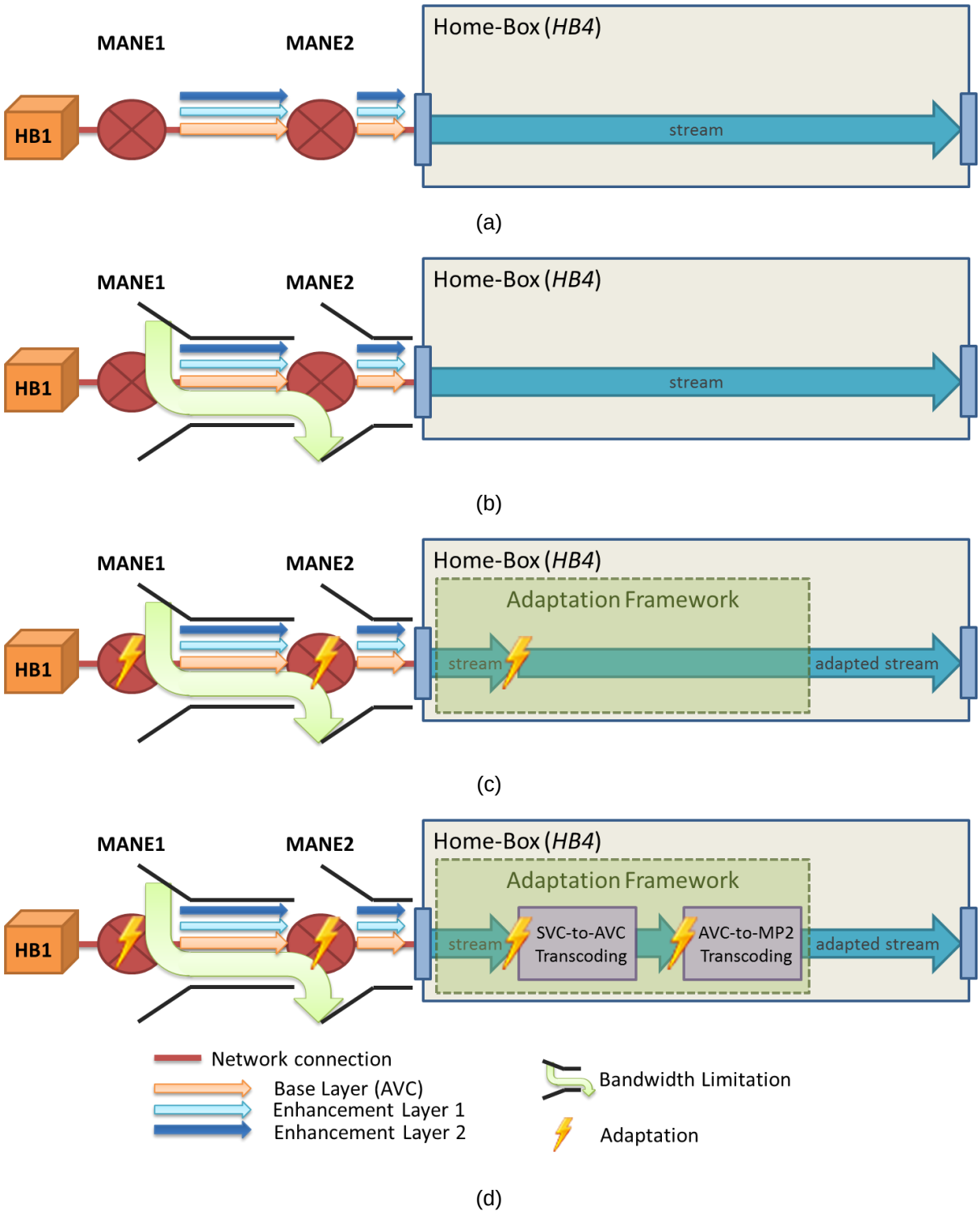


Figure 72: Testing scenarios for video quality evaluations in the end-to-end streaming system.

In a second step, we limited the available bandwidth of the MANEs via the Linux traffic control tool *tc*, again with all adaptation tools disabled, and captured the streamed SVC bitstreams on our test-bed setup. The scenario is depicted in Figure 72 (b). The traffic control tool was configured to limit the available bandwidth to 1,900 kbps, 1,700 kbps, and 1,000 kbps, respectively.

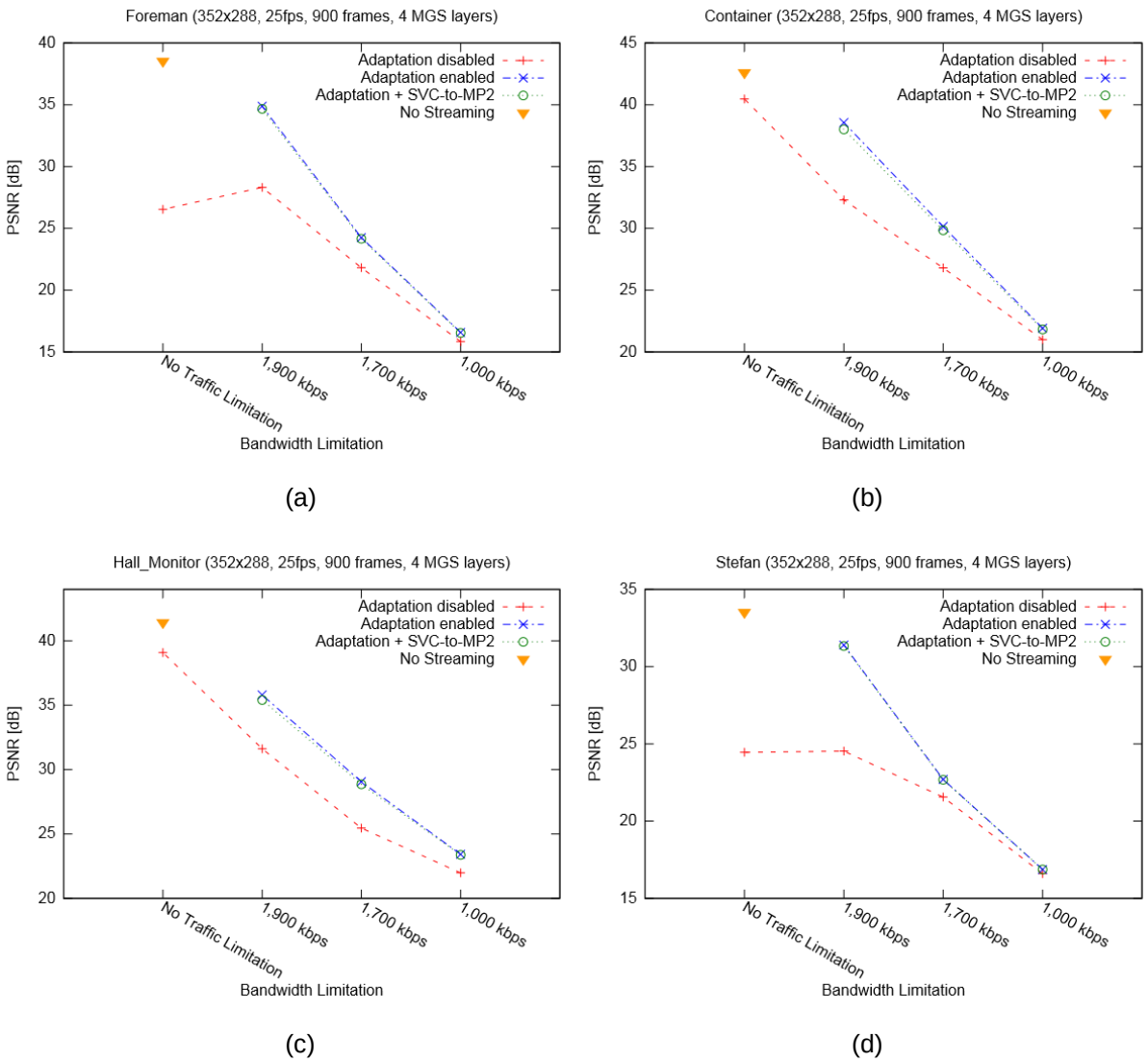


Figure 73: PSNR results for end-to-end streaming under bandwidth limitations for (a) *Foreman*, (b) *Container*, (c) *Hall_Monitor*, and (d) *Stefan* sequences.

In the next step, we compared the impact of packet loss without adaptation to the impact on a transmission with enabled adaptation at MANEs and Home-Boxes. The test scenario is shown in Figure 72 (c). Again, the available bandwidth was limited to 1,900 kbps, 1,700 kbps, and 1,000 kbps, respectively. For adaptation decision-taking, we applied the MPEG-21 ADTE adaptation logic provided in Annex G.

Finally, transcoding to MPEG-2 was added to the adaptive streaming as shown in Figure 72 (d). This configuration corresponds to a typical end-to-end streaming session under limited bandwidth due to cross-traffic.

PSNR results for the above scenarios are provided for all four test sequences in Figure 73. As stated above, the results are the average values from the two better test runs per sequence and scenario. The PSNR was calculated against the original raw YUV sequence. The quality of the SVC bitstream before streaming is shown for reference. The average results across all test sequences are given in Figure 74.

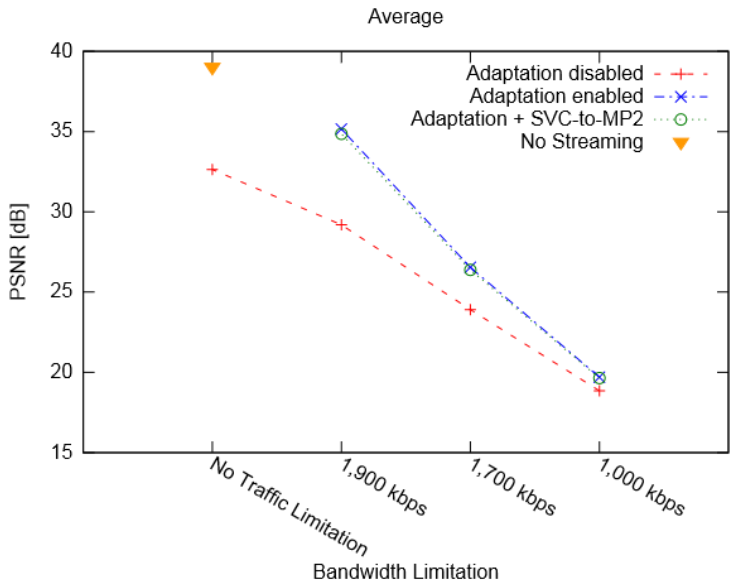


Figure 74: Averaged PSNR results for end-to-end streaming under bandwidth limitations.

It can be observed that adaptation significantly improves the video quality, especially for moderate packet loss. At 1,900 kbps, adaptation improves the quality by 5.97 dB on average, at 1,700 kbps by 2.63 dB, and at 1,000 kbps by only 0.85 dB.

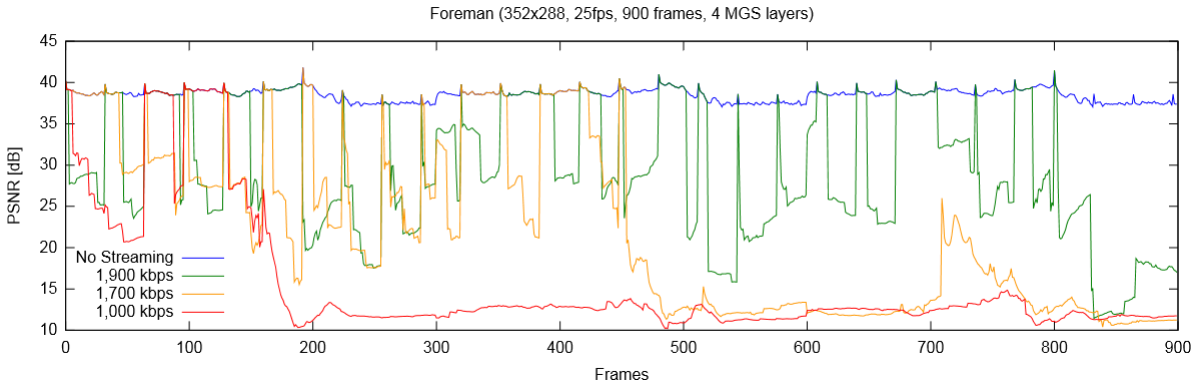
Note that at 1,000 kbps nearly 50% of the packets are lost. The bSoft decoder usually does not handle such high packet loss well. Even after adaptation reduces the stream to a lower MGS layer, the bSoft encoder was often incapable of stabilizing the video quality.

Transcoding to MPEG-2 only reduces the video quality by around 0.17 dB. This is in line with our findings in Section 4.3.3.3. Compared to the distortion due to packet loss, this degradation is negligible.

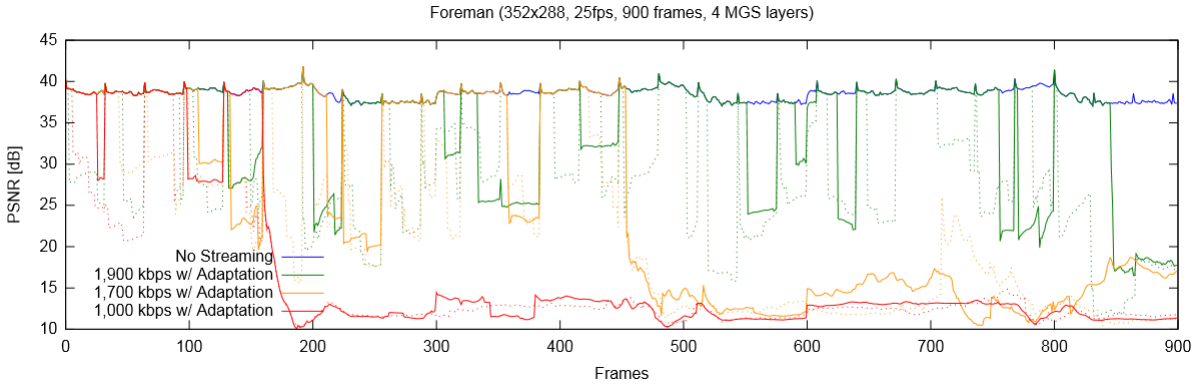
Per-frame PSNR results for the *Foreman* sequence are exemplarily shown in Figure 75. Streaming without adaptation is shown in Figure 75 (a). It can be observed that for higher packet loss, the decoder becomes and remains unstable. Already for a bandwidth limitation to 1,900 kbps, frequent and strong disruptions in quality have to be taken into account. When enabling adaptation, the disruptions are reduced due to down-switching to a lower bitrate as shown in Figure 75 (b). PSNR results for streaming without adaptation are shown in dotted lines for reference. However, for strong packet loss, we again observe instable behavior of the decoder.

From the test runs in Figure 75 (b), we provide two snapshots in Figure 76 to illustrate the distortion around frame 375 for bandwidth limitations of (a) 1,900 kbps and (b) 1,000 kbps.

Throughout these tests, we identified several possibilities for improving the performance of our adaptation logic. For example, the MANE should react faster to high packet loss in order to avoid unstable behavior at the decoder. The adaptation logic is stateless, i.e., it is not aware of previous adaptation decisions. By remembering its previous decision, frequent switches between layers could be



(a)



(b)

Figure 75: Per-frame PSNR results for end-to-end streaming with traffic limitation (a) without adaptation and (b) with adaptation.

reduced. Furthermore, the estimation of the bandwidth-vs.-packet-loss characteristics discussed in Section 5.4.2 should be fine-tuned based on our results.

5.6 Conclusions

In this chapter, we have shown how distributed adaptation can be integrated into an SVC streaming system. Starting from a discussion of use cases for SVC streaming in content-aware networks, we have shown how adaptation is performed on Media-Aware Network Elements and Home-Boxes in the ALICANTE architecture. We have investigated adaptation strategies for SVC and have validated our distributed adaptation approach in an integrated end-to-end test-bed setup.

As this chapter has touched various aspects of adaptation, the following list summarizes the key results:

- RTP-based unicast and multicast streaming of SVC can benefit greatly from content-awareness for routing and forwarding. But also P2P and HTTP streaming scenarios will be able to use MANEs to their advantage for caching/buffering purposes.

(a) (b)

Figure 76: Snapshots for (a) moderate distortion for 1,900 kbps bandwidth limitation and (b) high distortion for 1,000 kbps bandwidth limitation.

- The research questions towards a distributed adaptation decision-taking framework identified in Section 2.3.3.1 can now be answered (cf. [9]):
    - *Where to adapt?*

      In the media ecosystem architecture proposed by the ALICANTE project, distributed adaptation of SVC streams is realized by adaptation at network edges and in-network adaptation at MANEs. Adaptation shall always be performed as early as possible on the delivery path to avoid superfluous transmission of content. On the other hand, terminal capabilities or user preferences should not be propagated to the content-aware network environment.
    - *When to adapt?*

      At the content request phase, the combination of SVC layers has to be decided based on terminal capabilities and user preferences. Whether the decision is performed at the client or server depends on the intended infrastructure scalability, the business model, and deployment scenario rather than on the supported adaptation operations. During streaming, dynamic bitrate adaptation towards network conditions is best performed within the network. At the client side, support of heterogeneous terminals is achieved through SVC tunneling, relying on a smart home-gateway such as the Home-Box.
    - *How often to adapt?*

      Based on available literature, we suggest that the interval between two representation switches should be at least 2 seconds [242][243]. Nevertheless, viewers prefer multiple small quality changes over a single, large switch [244]. In case of network congestion, the adaptation should always be performed immediately; only up-switching to a higher representation should be scheduled accordingly to avoid flickering. We

have proposed a new concept, called *representation switch smoothing*, for further reducing the annoyance of quality switches. For RTP streaming, in-network adaptation avoids packet loss – a lost RTP packet can cause one or more SVC NALUs to be discarded, resulting in distortion and error propagation. For HTTP streaming, the goal of adaptation is to prevent playback stalling – initial delay to fill the client's buffer is generally better tolerated by viewers than any stalling event, no matter how small [239].

- How *to adapt?*

  Within the network, bitrate-based adaptation shall be deployed. While this is a simple and efficient strategy, some studies also have proposed in-network adaptation based on an on-the-fly QoE estimation [224]. However, this will require a careful configuration of input parameters to the QoE estimation algorithm. Client-side adaptation best focuses on resolution and video coding format of the terminal's media player. As no industry streaming solution documents frame rate adaptation (cf. Section 3.2.2), we are skeptic towards its acceptance in real-life streaming systems. In general, we consider quality scalability the most suitable scalability dimension of SVC in adaptive streaming scenarios (cf. also Section 3.5.2.2).

  We have created an integrated test-bed setup to demonstrate the adaptation capabilities of an end-to-end SVC streaming system. Based on this test-bed, we have tested the delay of typical streaming sessions and evaluated the video quality under various conditions. With SVC-to-AVC transcoding at the Home-Box, around 10.1 seconds of delay have to be taken into account, 98% of which are due to the deployed implementation of the SVC-to-AVC transcoder. The high latency was presumably caused by the transcoder's poorly implemented handling of real-time data. Additional AVC-to-MPEG-2 transcoding via our general-purpose transcoding module increases the delay by around 0.36 seconds, which is a more reasonable transcoder latency. We have evaluated the video quality for end-to-end streaming under various bandwidth limitations. Our results show that adaptation at the MANE can increase the video quality by up to 6 dB for moderate packet loss rates. For extreme packet loss scenarios of nearly 50%, the decoder becomes and remains unstable, even if the MANE manages to reduce the packet loss through adaptation. In an SVC tunneling scenario, the transcoding to MPEG-2 reduces the video quality by 0.17 dB on average. Based on our findings, we will continue to improve our streaming setup in the ALICANTE project.

Future work should provide a clearer picture of the coordination for distributed adaptation. While in the ALICANTE architecture, the coordination between MANEs is limited to mere configuration of adaptation policies, further research will be required to understand how multiple points of congestion along the network path can be handled. In our tests, we deployed a simple and straight-forward adaptation logic. More sophisticated adaptation strategies are available in the literature as discussed in Section 5.4.1. However, the proper configuration of adaptation logics at the MANEs needs further study in the context of distributed adaptation. Other research challenges such as efficient on-the-fly QoE estimation and subsequent QoE-based adaptation or the evaluation of representation switch smoothing remain open as well.

6 Conclusions and Future Work

6.1 Summary

The goal of this thesis has been to research and develop mechanisms for streaming and distributed adaptation of scalable media resources. This final chapter will summarize challenges and contributions towards this goal, complemented by an outlook on future work. In the previous chapters, we have covered a framework for SVC encoding, including encoding guidelines and performance evaluations, introduced and evaluated the concept of SVC tunneling, discussed challenges for distributed adaptation in content-aware network environments, and demonstrated our developments in an integrated end-to-end streaming system setup.

The first key aspect to adaptive media streaming of SVC is a proper encoding of the source content. As existing literature in this field often pays little attention to realistic encoding configurations, we developed guidelines for SVC encoding in Chapter 3 that are aligned with the recommendations of industry solutions. Chapter 3 has also introduced the *hybrid SVC-DASH* approach. This approach targets adaptive streaming to different device classes by providing a separate SVC bitstream (consisting of a base layer and several enhancement layers) per device class. In a series of performance evaluations, we have validated our encoding guidelines and the hybrid SVC-DASH approach. The performance evaluations have comprised various configurations for high-definition test content. We have tested the most prominent SVC encoder implementations and highlighted their characteristics under those test conditions. The test results show how scalability configurations can be deployed for efficient SVC streaming.

Traditionally, SVC streaming requires the content to be available in SVC as well as SVC decoding support at the client terminal. In order to enable SVC streaming on legacy systems, we have developed the concept of *SVC tunneling* in Chapter 4. Media resources are transcoded to SVC at the server side and back to a non-scalable target format at the client side. The goal of this approach is to allow efficient in-network adaptation and to enable bandwidth savings in multicast scenarios. The concept is supported by the system architecture of the FP7 ALICANTE project, which places enhanced home-gateways (Home-Boxes) at the edges of the network in order to provide an overlay network for media processing and adaptation. However, the transcoding to and from SVC reduces the video quality. Thus, there is a trade-off between achievable bandwidth efficiency and incurred quality loss. In our tests, we have investigated this trade-off for the example of MPEG-2 as the source and target format. In this course, we have also documented the steps taken to improve the test-bed setup throughout our tests. We have performed our evaluations for a proprietary SVC encoder/decoder and for the JSVM reference software. The proprietary SVC decoder provides real-time capabilities at least to some extent. On the other hand,

the slower reference software exhibits better rate-distortion performance, resulting in a better trade-off between bandwidth savings and quality impact.

The adaptation of scalable media resources in a content-aware network and the distribution of adaptation actions were discussed in Chapter 5. We have investigated the potential impacts of content-aware in-network adaptation of scalable media resources on transport mechanisms, such as RTP-, HTTP-, or P2P-based streaming. Even though many challenges remain open in this context, we argue that scalable media coding in content-aware networks will play a major role in the Future (media) Internet for improving the QoS/QoE management of adaptive media streaming in the long run. Towards a more short-term adoption of distributed adaptation, we have surveyed existing adaptation logics and described the distributed adaptation system and associated adaptation logics that we developed in the course of the ALICANTE project. Furthermore, we have identified a possible issue of the viewing experience in adaptive HTTP streaming. That is, the switch between two (quality) representations may unnecessarily distract the end user. We have thus introduced the concept of *representation switch smoothing*, allowing a gradual transition between the representations. We have realized an end-to-end streaming system prototype of the ALICANTE distributed adaptation framework. We have also documented the performance tests of this streaming system. So, Chapter 5 has combined the research contributions of the previous chapters into an integrated system of a scalable media delivery chain featuring distributed adaptation.

6.2 Findings

The initial research objectives stated in Section 1.2 have been addressed as follows.

(1) to evaluate the performance of *SVC encoding configurations* and scalability features:

We have performed extensive evaluations of SVC encoding configurations, focusing on spatial and quality scalability. We have tested the JSVM reference software and three major proprietary SVC encoders. We carefully selected full HD (1080p) test sequences for our evaluations, considering their amounts of Spatial and Temporal Information. The tests have addressed SVC streaming in general as well as configurations that are particularly interesting for HTTP-based streaming. The video quality has been evaluated via the commonly used PSNR metric as well as VQM, which correlates better with the human visual system. We have found several differences in the reported quality between the metrics.

The general part comprises rate-distortion evaluations of rate control modes, the combination of spatial and quality scalability, the number of SVC quality layers, and requantization of SVC quality layers. The results show that the JSVM reference software is about two orders of magnitude slower than fast proprietary SVC encoders but clearly outperforms all

proprietary encoders in terms of coding efficiency. For the number of SVC quality enhancement layers, each enhancement layer increases the coding overhead by slightly more than the 10% that is often claimed in related literature, with some variations among different encoders.

With a focus on DASH, we extended our evaluations of rate control modes from 2 to 4 SVC quality layers. Based on our evaluations of the combination of spatial and quality scalability, we developed the *hybrid SVC-DASH* approach. This separation of SVC bitstreams per spatial resolution enables better video qualities (around 2.2 dB higher PSNR at the highest resolution) at a moderate increase in storage requirements. Our tests have also shown that the combination of the coarse-grain and medium-grain scalability modes is not useful for increasing the number of SVC quality layers because it is inefficient in terms of supported extraction points for adaptation. In summary, our evaluations have provided a thorough model of the coding efficiencies and characteristics of major SVC encoders.

(2) to develop *guidelines for SVC encoding* in the context of adaptive media streaming:

We have found that existing research literature on SVC performance rarely considers encoding configurations used by industry streaming solutions. We have surveyed AVC encoding recommendations of major streaming solutions. From the multitude of recommendations, we have devised common encoding guidelines for adaptive media streaming of AVC – and subsequently SVC. The investigated seven industry streaming solutions list 26 different resolutions, which often differ from the resolutions commonly used in research literature. The 7 most relevant resolutions have been selected for our devised recommendations. For each resolution, we have formulated bitrate recommendations for two and four bitrates. We have provided those bitrate recommendations for AVC and for SVC. A streaming system would typically use 6 to 12 extraction points (i.e., resolution-bitrate tuples) from that list. Out of the 7 resolutions, 4 have been highlighted in the context of DASH. We have also briefly discussed challenges of segmentation, container formats, and other streaming-related aspects. The devised encoding recommendations provide a common ground for advanced SVC performance studies in the context of adaptive media streaming.

(3) to investigate the feasibility of *SVC tunneling* for device-independent access:

With the availability of an advanced home-gateway (such as the Home-Box in the ALICANTE architecture), universal multimedia experience with heterogeneous devices even becomes possible in combination with the deployment of SVC in the network. We have presented the concept of *SVC tunneling* that allows for efficient in-network adaptation and even bandwidth savings in multicast scenarios by relying on transcoding at network edges.

We have evaluated the trade-off between bandwidth savings and the quality loss due to transcoding for the example of MPEG-2 as the source and target format. In the course of our evaluations, we have developed a model for the selection of quantization parameters for pixel-domain transcoding to and from SVC. SVC-to-MPEG-2 transcoding at the client side can typically assume overprovisioned home-network links, allowing for transcoding to the highest supported bitrates. On the other hand, the QP for MPEG-2-to-SVC transcoding at the server side is the main factor for controlling the quality-versus-bandwidth trade-off. We have evaluated this trade-off for the proprietary bSoft SVC encoder in order to be able to support real-time deployments and for the JSVM reference software in order to optimize rate-distortion performance. For full SVC tunneling with transcoding at the server and client sides with the bSoft encoder, around 2.5 dB PSNR loss has to be considered for SVC tunneling to be more bandwidth efficient than MPEG-2 simulcast. The results for the JSVM significantly improve the trade-off (at the expense of transcoding speed). For example, at a quality loss of 0.33 dB, SVC tunneling requires 41.8% less bandwidth than a comparable MPEG-2 simulcast. We have also evaluated the trade-off for partial SVC tunneling, i.e., if one of the two transcoding steps can be avoided, and have shown that quality loss is further reduced.

(4) to analyze the effects of *scalability features and adaptation configurations* on *content- and context-aware* media delivery:

We have surveyed research literature on adaptation of SVC. The survey addresses the evolution of SVC adaptation, which traditionally has been located at the server side, e.g., for IPTV or RTP-based VoD services, was later extended onto routers performing in-network adaptation, and is now increasingly located at the client side as well due to the advance of HTTP-based media streaming. The overall objective of SVC adaptation is the selection of packets (i.e., NALUs) from the SVC bitstream so that the Quality of Experience for the end user(s) is maximized under constraints of transmission resources while also minimizing the utilization of transmission resources. However, the QoE for video consumption is influenced by a huge number of factors, making it hard to model. It is not only affected by spatial resolution, frame rate, and video SNR, but also by the timing, duration, and pattern of adaptation between representations with different scalability features, the frequency and amplitudes of adaptations, distortion due to packet loss, initial playout delay, playback stalling due to rebuffering and many other factors. Ongoing research in this field strives to understand the impact of these factors. As one contribution, we have researched the perception of switches between representations of a video. We have introduced the approach of smooth transitions between representations, called *representation switch smoothing*, that avoids abrupt changes in video quality through a smooth transition between segments of different bitrates. We have also presented several implementation options of the proposed

approach. First evaluation results indicate that viewers prefer such a smooth transition over a traditional hard switch.

Another factor that influences the adaptation options, and subsequently the achievable QoE, is the choice of encoding configurations. In particular, the number of layers and the bitrates of these layers have to be selected carefully to allow flexible adaptation on the one hand and to avoid excessive coding overhead on the other hand. We have designed our encoding recommendations accordingly as discussed above.

We have also documented and validated the adaptation logic deployed within the ALICANTE adaptation framework. The adaptation logic has been implemented as a standard-compliant MPEG-21 DIA description, enabling interoperability and extensibility.

(5) to investigate the applicability of *distributed adaptation* in *content-aware networks* for different *transport mechanisms*:

We have discussed the challenges and potentials of advanced SVC adaptation at multiple nodes within a content-aware network. In our discussion, we have investigated how different transport mechanisms, such as RTP-, HTTP-, or P2P-based streaming, can benefit from content-aware features of advanced network nodes. We have found that content-aware caching/buffering strategies at those network nodes will play a major role for realizing advanced adaptive streaming in a Future (media) Internet; so will the in-network adaptation of scalable media resources. Among others, we have proposed that advanced network nodes may even collaborate in P2P streaming scenarios by acting as streaming peers themselves. Our analysis has considered several aspects of in-network processing of scalable media resources for improving the QoS/QoE management of adaptive media streaming.

Within the context of the ALICANTE architecture, we have developed an adaptation framework that allows for dynamic adaptation at the network edges (i.e., the Home-Boxes) as well as in the network.

(6) to evaluate the performance of *distributed media adaptation* in an end-to-end streaming system:

Although the integration in an end-to-end streaming system has proven to be quite hard due to many implementation issues, we have managed to demonstrate and validate the adaptation capabilities of our system. This end-to-end streaming system integrates the findings of our SVC encoding recommendations, the deployment of SVC tunneling, and distributed adaptation features. In the course of our evaluations, the end-to-end delay for streaming with (partial) SVC tunneling was measured. Our results show that an end-to-end delay below one second for SVC tunneling should be possible, given an improved SVC-to-AVC transcoder. The PSNR for adaptive streaming with SVC tunneling is up to 6 dB higher than for non-

adaptive streaming in our test-bed setup. Nevertheless, several challenges remain for the configuration and coordination of distributed adaptation in an end-to-end media streaming system.

6.3 Future Work

In the adaptive streaming system presented in Section 5.5, we have so far only tested a small set of the encoding configurations devised in Chapter 3. Future work in this area will test additional configurations for realistic streaming scenarios. The proper deployment of container formats for the integration with audio and other multimedia data (e.g., sensory effects metadata) has to be researched. It also remains to be evaluated how our encoding recommendations for HTTP streaming, in particular the *hybrid SVC-DASH* approach, affect caching performance. Since a scalable extension to the recently ratified HEVC standard is currently under development, extensive performance evaluations will be important for its adoption.

In the context of *SVC tunneling*, future work should target high-definition content, source and target coding formats other than MPEG-2 (or AVC), as well as subjective tests. While we have evaluated SVC tunneling based on pixel-domain transcoding, the evaluation of fast transform-domain transcoders would be an interesting opportunity to improve transcoding speed for higher resolutions.

Our discussion of scalable media coding in content-aware networks has given a broad outlook on the opportunities, but also on the issues of such approaches. The identified challenges will have to be addressed and researched. A key aspect for the deployment will be how well such infrastructures will scale. Thus, efficient algorithms for in-network (and potentially cross-layer) adaptation and advanced caching mechanisms will be needed. But future work should also address security and privacy aspects of the proposed approaches.

We have seen in Section 5.4.1 that various adaptation logics try to solve specific issues in media streaming scenarios. Adaptation no longer just targets the optimization of network resource utilization, but also increasingly addresses the subtle psychological aspects of video perception. One factor is the sensibility to changes in quality. Representation switch smoothing tries to reduce our perception of quality changes. Further evaluations will be required to assess the benefits of this approach. As our understanding of QoE and its influence factors grows, different adaptation approaches will have to be combined.

The efficient distribution of adaptation steps for SVC streaming (and for the delivery of scalable video resources in general) promises further improvements in terms of network resource utilization and QoE management. While we have demonstrated a prototype of an end-to-end streaming system with distributed cross-layer adaptation, there are still many open challenges in this area. Future work shall improve the coordination of adaptation configurations at the involved nodes. For the adaptation framework of the ALICANTE system architecture, the coordination of in-network

adaptation is performed by a domain-management entity (the CAN Manager) that provides adaptation policies to the network nodes. It will have to be evaluated whether these adaptation policies enable optimal adaptation decisions and how different network domains can communicate with each other. There is also research being performed on multi-video rate allocation at the network edges and within the network. Eventually, these aspects will result in a clear overall picture of the coordination and signaling required for efficient distributed adaptation. Another key aspect for future work is the standardization of protocols and interfaces to ensure interoperability of advanced media delivery systems.

Annex A – Abbreviations and Acronyms

4CIF	4x CIF
A_PSQA	ALICANTE PSQA
ABR	Average Bitrate
ACR	Absolute Category Rating
ADTE	Adaptation Decision-Taking Engine
ADTF	Adaptation Decision-Taking Framework
AF	Adaptation Framework
ALICANTE	Media Ecosystem Deployment through Ubiquitous Content-Aware Network Environments
API	Application Programming Interface
AQoS	Adaptation QoS
AVC	Advanced Video Coding
AVI	Audio Video Interleave
B frame	Bi-Predicted frame
BD	Bjontegaard Delta
BL	Base Layer
CABAC	Context-Adaptive Binary Arithmetic Coding
CAN	Content-Aware Network
CATI	Content-Aware Transport Information
CAVLC	Context-Adaptive Variable Length Coding
CBR	Constant Bitrate
CCN	Content-Centric Networking
CDN	Content Delivery Network
CGS	Coarse-Grain Scalability
CIF	Common Intermediate Format
CON	Content-Oriented Networking
CP	Content Provider
CPU	Central Processing Unit
CS-DON	Cross-Session Decoding Order Number

DANAE	Dynamic and Distributed Adaptation of Scalable Multimedia Content in a Context-Aware Environment
dB	decibel
DASH	Dynamic Adaptive Streaming over HTTP
DCT	Discrete Cosine Transform
DIA	Digital Item Adaptation
DID	Dependency Identifier
DiffServ	Differentiated Services
DMOS	Differential MOS
DPI	Deep Packet Inspection
dQP	deltaQP
D-Q-RAM	Distributed Quality of Service Resource Allocation Model
DVD	Digital Versatile Disk or Digital Video Disk
DWT	Discrete Wavelet Transform
EC	European Commision
EL	Enhancement Layer
ENTHRONE	End-to-End QoS through Integrated Management of Content, Networks and Terminals
ES	Elementary Stream
ESS	Extended Spatial Scalability
EU	European Union
FI	Future Internet
FIFO	First-In-First-Out
FLS	Frequent Layer Switching
FP6	Sixth Framework Programme
FP7	Seventh Framework Programme
fps	frames per second
GB	Gigabyte
GHz	Gigahertz
GOP	Group of Pictures
GPT	General-Purpose Transcoder
GRED	Generalized Random Early Detection
H.264/AVC	(see AVC)

HB	Home-Box
HD	High-Definition
HEVC	High Efficiency Video Coding
HLS	HTTP Live Streaming
HT	Hadamard Transform
HTTP	Hypertext Transfer Protocol
I frame	Intra-predicted frame
ICN	Information-Centric Networking
IDR	Instantaneous Decoding Refresh
IEC	International Electrotechnical Commission
IP	Internet Protocol
IPTV	Internet Protocol Television
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISO	International Organization for Standardization
ITU-T	International Telecommunication Standardization Sector
JCT-VC	Joint Collaborative Team on Video Coding
JSVM	Joint Scalable Video Model
JVT	Joint Video Team
kbps	kilobit per second
LCD	Liquid-Crystal Display
LRU	Least Recently Used
MANE	Media-Aware Network Element
Mbps	Megabit per second
MDC	Multiple Description Coding
MDS	Multimedia Description Schemes
MEDIEVAL	MultiMEDia transport for mobIlE Video AppLications
MGS	Media-Grain Scalability
MKV	Matroska Multimedia Container
MOS	Mean Opinion Score
MP4	MPEG-4 Part 14
MPD	Media Presentation Description
MPEG	Moving Picture Experts Group

MPEG-4 Visual	MPEG-4 Part 2
MPLS	Multiprotocol Label Switching
MSE	Mean Squared Error
MSPT	Multimedia Service Platform Technologies
MST	Multi-Session Transmision
NAL	Network Abstraction Layer
NALU	NAL Unit
NAT	Network Address Translation
NTIA	National Telecommunications and Information Administration
P frame	Predicted frame
P2P	Peer-to-Peer
PC	Personal Computer
PDT	Pixel Domain Transcoding
PE	Processing Engine
PPSPP	Peer-to-Peer Streaming Peer Protocol
PSNR	Peak Signal-to-Noise Ratio
PSQA	Pseudo-Subjective Quality Assessment
QCIF	Quarter CIF
QID	Quality Identifier
QoE	Quality of Experience
QoS	Quality of Service
QP	Quantization Parameter
QVGA	Quarter Video Graphics Array
RAM	Random-Access Memory
RD	Rate-Distortion
RDLM	Receiver-Driven Layered Multicast
RNN	Random Neural Network
ROI	Region of Interest
RTCP	RTP Control Protocol
RTMP	Real-Time Messaging Protocol
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
SDP	Session-Description Protocol

SEI	Supplemental Enhancement Information
SHVC	Scalable High Efficiency Video Coding
SI	Spatial Information
SLA	Service-Level Agreement
SNR	Signal-to-Noise Ratio
SP	Service Provider
SST	Single-Session Transmision
SVC	Scalable Video Coding
TCP	Transmission Control Protocol
TDT	Transform Domain Transcoding
TI	Temporal Information
TID	Temporal Identifier
TS	Transport Stream
TV	Television
UCD	Universal Constraints Description
UDP	User Datagram Protocol
UED	Usage Environment Description
UMA	Universal Multimedia Access
UME	Universal Multimedia Experience
US	United States
VBR	Variable Bitrate
VCL	Video Coding Layer
VoD	Video on Demand
VQM	Video Quality Metric
VSS	Vanguard Software Solutions
WSVC	Wavelet-based Scalable Video Coding
XML	eXtensible Markup Language
XSLT	eXtensible Stylesheet Language Transformation

Annex B – Configurations of Tested Encoders

This Annex provides configurations of all tested encoders for the test described in Section 3.4.1.

JSVM encoder:

File main.cfg:

```
# JSVM Main Configuration File

#============================== GENERAL
======================================
OutputFile              $(OUTPUT_FILE)   # Bitstream file
FrameRate               25.0       # Maximum frame rate [Hz]
FramesToBeEncoded       1000        # Number of frames (at
input frame rate)
CgsSnrRefinement        1          # SNR refinement as 1: MGS;
0: CGS
EncodeKeyPictures       1          # Key pics at T=0 (0:none,
1:MGS, 2:all)
MGSControl              2          # ME/MC for non-key
pictures in MGS layers
                                   # (0:std, 1:ME with EL,
2:ME+MC with EL)

#============================== CODING STRUCTURE
==============================
GOPSize                 4          # GOP Size (at maximum
frame rate)
IntraPeriod             32         # Intra Period

#============================== LAYER DEFINITION
==============================
NumLayers               2          # Number of layers
LayerCfg                layer0.cfg # Layer configuration file
LayerCfg                layer1.cfg # Layer configuration file

#============================== MOTION SEARCH
================================
SearchMode              4          # Search mode
(0:BlockSearch, 4:FastSearch)
```

```
SearchFuncFullPel       0           # Search function full pel
                                    #   (0:SAD, 1:SSE,
2:HADAMARD, 3:SAD-YUV)
SearchFuncSubPel        2           # Search function sub pel
                                    #   (0:SAD, 1:SSE,
2:HADAMARD)
SearchRange             32          # Search range (Full Pel)
ELSearchRange           8           # Enh. layer search range
FastBiSearch            1           # Fast bi-directional
search (0:off, 1:on)
BiPredIter              2           # Max iterations for bi-
pred search
IterSearchRange         4           # Search range for
iterations (0: normal)
```

File layer0.cfg:

```
# JSVM Layer 0 Configuration File

#====================== INPUT / OUTPUT
=========================================
SourceWidth          1920           # Input  frame width
SourceHeight         1080           # Input  frame height
FrameRateIn          25             # Input  frame rate [Hz]
FrameRateOut         25             # Output frame rate [Hz]
InputFile            $(INPUT_FILE) # Input  file
ReconFile            rec_layer0.yuv # Reconstructed file
SymbolMode           1              # 0=CAVLC, 1=CABAC

#====================== CODING
================================================
QP                   26.0           # Quantization parameters
# Important: MeQPx should be set to QP - 2.

#====================== CONTROL
================================================
MeQP0                24.00          # QP for mot. est. / mode
decision (stage 0)
MeQP1                24.00          # QP for mot. est. / mode
decision (stage 1)
MeQP2                24.00          # QP for mot. est. / mode
decision (stage 2)
MeQP3                24.00          # QP for mot. est. / mode
decision (stage 3)
```

```
MeQP4                 24.00          # QP for mot. est. / mode
decision (stage 4)
MeQP5                 24.00          # QP for mot. est. / mode
decision (stage 5)
```

File layer1.cfg:

```
# JSVM Layer 1 Configuration File

#====================== INPUT / OUTPUT
=========================================
SourceWidth           1920           # Input  frame width
SourceHeight          1080           # Input  frame height
FrameRateIn           25             # Input  frame rate [Hz]
FrameRateOut          25             # Output frame rate [Hz]
InputFile             $(INPUT_FILE) # Input  file
ReconFile             rec_layer1.yuv # Reconstructed file
SymbolMode            1              # 0=CAVLC, 1=CABAC

#====================== CODING
================================================
QP                    24.0           # Quantization parameters
# Important: MeQPx should be set to QP - 2.

#====================== CONTROL
================================================
MeQP0                 22.00          # QP for mot. est. / mode
decision (stage 0)
MeQP1                 22.00          # QP for mot. est. / mode
decision (stage 1)
MeQP2                 22.00          # QP for mot. est. / mode
decision (stage 2)
MeQP3                 22.00          # QP for mot. est. / mode
decision (stage 3)
MeQP4                 22.00          # QP for mot. est. / mode
decision (stage 4)
MeQP5                 22.00          # QP for mot. est. / mode
decision (stage 5)

InterLayerPred        2              # Inter-layer Pred. (0:no,
1:yes, 2:adap.)
```

MainConcept encoder:

```
#################################
# AVC encoder configuration file#
#################################

[SVC Settings]
num_layers                      = 2
mgs                             = 1
inter_layer_deblocking          = 1

[SVC Layer 0000]
profile_id                      = 3
level_id                        = 51
idr_interval                    = 32
reordering_delay                = 4
use_b_slices                    = 1
interlace_mode                  = 0
def_horizontal_size             = 1920
def_vertical_size               = 1080
frame_rate                      = 25.0000000000
num_reference_frames            = 4
search_range                    = 144
rd_optimization                 = 1
max_l0_active                   = 0
max_l1_active                   = 0
quant_pI                        = 26
quant_pP                        = 26
quant_pB                        = 26
bit_rate_mode                   = 1
bit_rate_buffer_size            = 12000000
bit_rate                        = 0
max_bit_rate                    = 0
inter_search_shape              = 1
entropy_coding_mode             = 1
use_hadamard_transform          = 0
sar_width                       = 1
sar_height                      = 1
video_format                    = 1
video_full_range                = 0
num_units_in_tick               = 1080000
time_scale                      = 27000000
vbv_buffer_fullness             = 0
vbv_buffer_fullness_trg         = 12000000
vbv_buffer_units                = 1
cpb_removal_delay               = 0
```

```
bit_rate_scale                  = 0
cpb_size_scale                  = 0
max_frame_size                  = {0,0,0,0}
hrd_maintain                    = 1
use_deblocking_filter           = 1
deblocking_alphaC0_offset       = -1
deblocking_beta_offset          = -1
adaptive_deblocking             = 0
video_type                      = 0
video_pulldown_flag             = 0
stream_type                     = 2
frame_mbs_mode                  = 0
bit_depth_luma                  = 8
bit_depth_chroma                = 8
chroma_format                   = 2
vui_presentation                = 0
write_au_delimiters             = 0
write_seq_end_code              = 1
write_timestamps                = 1
timestamp_offset                = 0
drop_frame_timecode             = 0
write_single_sei_per_nalu       = 0
write_seq_par_set               = 1
write_pic_par_set               = 1
log2_max_poc                    = 8
log2_max_frame_num              = 16
pic_order_cnt_type              = 0
pic_order_present_flag          = 0
fixed_frame_rate                = 1
frame_based_timing              = 0
vcsd_mode                       = 1
vcsd_sensibility                = 50
slice_mode                      = 1
slice_arg                       = 1
b_slice_reference               = 1
b_slice_pyramid                 = 1
cb_offset                       = 1
cr_offset                       = 1
me_subpel_mode                  = 2
me_weighted_p_mode              = 1
me_weighted_b_mode              = 0
enable_fast_intra_decisions     = 1
enable_fast_inter_decisions     = 1
pic_ar_x                        = -1
pic_ar_y                        = -1
```

```
calc_quality                    = 0
cpu_opt                         = 0
num_threads                     = 0
live_mode                       = 0
buffering                       = 0
min_quant                       = 0
max_quant                       = 51
max_slice_size                  = 0
encoding_buffering              = 0
low_delay                       = 0
air_mode                        = 0
detach_thread                   = 1
min_idr_interval                = 1
adaptive_b_frames               = 0
idr_frequency                   = 1
field_order                     = 0
fixed_i_position                = 0
isolated_gops                   = 0
fast_multi_ref_me               = 1
fast_sub_block_me               = 1
allow_out_of_pic_mvs            = 1
constrained_ref_list            = 1
enable_intra_big                = 1
enable_intra_8x8                = 1
enable_intra_4x4                = 1
enable_intra_pcm                = 0
enable_inter_big                = 1
enable_inter_8x8                = 1
enable_inter_4x4                = 1
enable_inter_pcm                = 0
fast_rd_optimization            = 1
quant_mode                      = 2
grain_mode                      = 0
grain_opt_strength              = 0
adaptive_quant_strength         = {0,0,0,0,0,0,0,0}
denoise_strength_y              = 0
denoise_strength_c              = 0
black_norm_level                = 0
seq_scaling_matrix_present_flag = 0
seq_scaling_list_present_flag   = {0,0,0,0,0,0,0,0}
intra_y_4x4_scaling_list[16]    =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
intra_cb_4x4_scaling_list[16]   =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
```

```
intra_cr_4x4_scaling_list[16]   =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
inter_y_4x4_scaling_list[16]    =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
inter_cb_4x4_scaling_list[16]   =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
inter_cr_4x4_scaling_list[16]   =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
intra_y_8x8_scaling_list[16]    =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0}
inter_y_8x8_scaling_list[16]    =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0}
constrained_intra_pred          = 0
air_split_frequency             = 0
hierar_p_frames                 = 0
aux_format_idc                  = 0
bit_depth_aux                   = 0
alpha_incr_flag                 = 0
alpha_opaque_value              = 0
alpha_transparent_value         = 0

[SVC Layer 0001]
profile_id                      = 3
level_id                        = 51
idr_interval                    = 32
reordering_delay                = 4
use_b_slices                    = 1
interlace_mode                  = 0
def_horizontal_size             = 1920
def_vertical_size               = 1080
frame_rate                      = 25.0000000000
num_reference_frames            = 4
search_range                    = 144
rd_optimization                 = 1
max_l0_active                   = 0
max_l1_active                   = 0
quant_pI                        = 24
quant_pP                        = 24
quant_pB                        = 24
bit_rate_mode                   = 1
bit_rate_buffer_size            = 16000000
```

```
bit_rate                        = 0
max_bit_rate                    = 0
inter_search_shape              = 1
entropy_coding_mode             = 1
use_hadamard_transform          = 0
sar_width                       = 1
sar_height                      = 1
video_format                    = 1
video_full_range                = 0
num_units_in_tick               = 1080000
time_scale                      = 27000000
vbv_buffer_fullness             = 0
vbv_buffer_fullness_trg         = 16000000
vbv_buffer_units                = 1
cpb_removal_delay               = 0
bit_rate_scale                  = 0
cpb_size_scale                  = 0
max_frame_size                  = {0,0,0,0}
hrd_maintain                    = 1
use_deblocking_filter           = 1
deblocking_alphaC0_offset       = -1
deblocking_beta_offset          = -1
adaptive_deblocking             = 0
video_type                      = 0
video_pulldown_flag             = 0
stream_type                     = 2
frame_mbs_mode                  = 0
bit_depth_luma                  = 8
bit_depth_chroma                = 8
chroma_format                   = 2
vui_presentation                = 0
write_au_delimiters             = 0
write_seq_end_code              = 1
write_timestamps                = 1
timestamp_offset                = 0
drop_frame_timecode             = 0
write_single_sei_per_nalu       = 0
write_seq_par_set               = 1
write_pic_par_set               = 1
log2_max_poc                    = 8
log2_max_frame_num              = 16
pic_order_cnt_type              = 0
pic_order_present_flag          = 0
fixed_frame_rate                = 1
frame_based_timing              = 0
```

```
vcsd_mode                       = 1
vcsd_sensibility                = 50
slice_mode                      = 1
slice_arg                       = 1
b_slice_reference               = 1
b_slice_pyramid                 = 1
cb_offset                       = 1
cr_offset                       = 1
me_subpel_mode                  = 2
me_weighted_p_mode              = 1
me_weighted_b_mode              = 0
enable_fast_intra_decisions     = 1
enable_fast_inter_decisions     = 1
pic_ar_x                        = -1
pic_ar_y                        = -1
calc_quality                    = 0
cpu_opt                         = 0
num_threads                     = 0
live_mode                       = 0
buffering                       = 0
min_quant                       = 0
max_quant                       = 51
max_slice_size                  = 0
encoding_buffering              = 0
low_delay                       = 0
air_mode                        = 0
detach_thread                   = 1
min_idr_interval                = 1
adaptive_b_frames               = 0
idr_frequency                   = 1
field_order                     = 0
fixed_i_position                = 0
isolated_gops                   = 0
fast_multi_ref_me               = 1
fast_sub_block_me               = 1
allow_out_of_pic_mvs            = 1
constrained_ref_list            = 1
enable_intra_big                = 1
enable_intra_8x8                = 1
enable_intra_4x4                = 1
enable_intra_pcm                = 0
enable_inter_big                = 1
enable_inter_8x8                = 1
enable_inter_4x4                = 1
enable_inter_pcm                = 0
```

```
fast_rd_optimization            = 1
quant_mode                      = 2
grain_mode                      = 0
grain_opt_strength              = 0
adaptive_quant_strength         = {0,0,0,0,0,0,0,0}
denoise_strength_y              = 0
denoise_strength_c              = 0
black_norm_level                = 0
seq_scaling_matrix_present_flag = 0
seq_scaling_list_present_flag   = {0,0,0,0,0,0,0,0}
intra_y_4x4_scaling_list[16]    =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
intra_cb_4x4_scaling_list[16]   =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
intra_cr_4x4_scaling_list[16]   =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
inter_y_4x4_scaling_list[16]    =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
inter_cb_4x4_scaling_list[16]   =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
inter_cr_4x4_scaling_list[16]   =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
intra_y_8x8_scaling_list[16]    =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0}
inter_y_8x8_scaling_list[16]    =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0}
constrained_intra_pred          = 0
air_split_frequency             = 0
hierar_p_frames                 = 0
aux_format_idc                  = 0
bit_depth_aux                   = 0
alpha_incr_flag                 = 0
alpha_opaque_value              = 0
alpha_transparent_value         = 0
```

VSS encoder:

```
##############################################################
##############
# VSofts H.264 Encoder 4.6
```

```
# Copyright (C) 2002-2012 Vanguard Software Solutions, Inc.
All Rights Reserved.
# Syntax tips:
#  a) ';' and '#' symbols at the start of line mean whole line
comment;
#  b) "//" is a comment till end of line like in "C";
#
################################################################
##############
#
# Here I am, configuration file the size of a planet and they
ask me to encode SVC. Call that job satisfaction? 'Cos I
don't.
#
################################################################
##############

svc.num_layers = 1
rc.qp_intra   = 26  // quant parameter for I-frames (0-51);
svc.layer[0].qp_intra = 24 //qp for intra-frames coding
(qp_delta_p and qp_delta_b is used from main settings)(used
for rc.type = 0)

gop.time_scale = 50000 // fps = time_scale/(2*num_units)

#----------------------- input description, to be set by
application level
input.width  = 1920 //Input frames width in pixels
input.height = 1080 //Input frame height in pixels
input.colorspace = 0  // 0=IYUV,I420; 1=YV12; 2=YUYV,YUY2;
3=YVYU; 4=UYVY; 5=RGB555; 6=RGB565; 7=RGB24; 8=RGB32
                         // 9 = YUV 4:0:0 planar, 10 = YUV
4:2:2 planar
        // if input frames are RGB upside down frames then
input.height must be negative
input.sample_size = 1 // bytes per sample (1)
input.significant_bits = 8  // significant bits per sample (8);

#----------------------- preprocessing
preproc.intra_precision = 2 // 0, 1, 2, 3, 4

#-------------------------------------------------
# chroma format idc valuse:
#      0 = YUV_400, 1 = YUV_420, 2 = YUV_422,  3 = YUV_444
# default value 1
```

```
# acceptable values 0, 1, 2
preproc.chroma_format_idc = 1

#----------------------
preproc.crop.enable = 0
preproc.crop.left   = 0
preproc.crop.top    = 0
preproc.crop.right  = 0
preproc.crop.bottom = 0

#-----------------------------------------------
# 0 = none, 1 = weak, 2 = moderate, 3 = middle, 4 = strong, 5
= very strong, 6 = maximum
preproc.me_denoise.level = 0
preproc.me_denoise.skip_luma   = 0
preproc.me_denoise.skip_chroma = 0

#-----------------------------------------------
# 0 = none, 1 = copy top feild, 2 = copy bottom field, 3 =
blend fields
preproc.deinterlace = 0

#-----------------------------------------------
# steps set (step0, step1, ... step6)
# step parameters set (param0, param1, param2, param3)
# type values
# 0x00 = none, parameters will be set to zero
# filter luma - accepable values 0 or 1 (1 means apply filter)
# filter chroma - acceptable values 0 or 1 (1 means apply
filter)
# 0x10 = BLUR_3x3    (param0=filter luma, param1=filter
chroma),
# 0x11 = BLUR_5x5    (param0=filter luma, param1=filter
chroma),
# 0x20 = SHARPEN_3x3 (param0=filter luma, param1=filter
chroma),
# 0x21 = SHARPEN_5x5 (param0=filter luma, param1=filter
chroma),
# 0x30 = MEDIAN_3x3  (param0=filter luma, param1=filter
chroma),
# 0x31 = MEDIAN_5x3  (param0=filter luma, param1=filter
chroma),

# several resize steps can be used at the same time (using
different slots e.g. step0 and step1)
```

```
# if is used several steps output step0 will be input step1
# but size must be in range (concrete dimension restriction
see in sdk documentation)
# 0x40 = RESIZE (param0=new picture width, param1=new picture
height)

#strength  [0..10]  0 - no filter applied, 1..10 filter
strength, 10 - moved objects and scene change cause artifacts
#buffer length [2..7] color planes counter which will be used
while filtration
# 0x50 = TEMPORAL_DENOISE (param0=luma strength, param1=luma
buffer length, param2=crhoma strength, param3=chroma buffer
length)
#
preproc.step[0].type = 0
preproc.step[0].param0 = 0
preproc.step[0].param1 = 0
preproc.step[0].param2 = 0
preproc.step[0].param3 = 0

################################################################
##############
# svc layers settings
#
# there are layer0, layer1, layer2 and layer3

# number of layers 0..15
# 0 means no SVC
# otherwice count layers counter
#svc.num_layers = 1
svc.key_picture_period = 0  // SVC and AVC key picture period
svc.temporal_mode = 0 // temporal scalability: 0=disabled,
1=enabled;
svc.multistream_mode = 0 // 0=SVC, 1=AVC, 2=MVC

# Bitwise svc/mvc/multistream flags:
#  1=Put MVC prefix-nal units into stream;
#  2=Put MVC picture delimiter into stream;
#       4=Use fast version of ParallelStream
#       8=Generate MVC SEI according to Blu Ray spec
svc.flags = 0

#layer0 description ----------------------------------------
-------------
```

```
# ------------------------ similar descriptions can present
for each layer

# layer extend - spatial extend, comparing to prev layer
# acceptable values
#  SVC_EXTEND_2X2 = 0  - extend twice in both direction
#  SVC_EXTEND_1X1 = 1  - no spatial extend
#  SVC_MGS_EXTEND = 2  - no spatial extend MGS coding
#  SVC_EXTEND_1_5 = 15  - 1.5 extend in both direction
#  SVC_EXTEND_CUSTOM = 100  - custom spatial extend
(dimensions must be set explicitly)
svc.layer[0].extend   = 2

#SVC Encoding tools is bitwise combination of the values
below:
#  SVC_ADAPTIVE_BASEMODE_FLAG = 0x01,
#  SVC_ADAPTIVE_RES_PRED_FLAG = 0x02,
#  SVC_ADAPTIVE_MV_PRED_FLAG = 0x04,
#  SVC_DEFAULT_BASEMODE_FLAG = 0x10,
#  SVC_DEFAULT_RES_PRED_FLAG = 0x20,
#  SVC_DEFAULT_MV_PRED_FLAG = 0x40
svc.layer[0].flags_i  = 0x7  //adaptive usage of all tools for
I-slice
svc.layer[0].flags_p  = 0x7  //adaptive usage of all tools for
P-slice
svc.layer[0].flags_b  = 0x7  //adaptive usage of all tools for
B-slice
svc.layer[0].sym_mode = 1  // select symbol mode: 0=UVLC;
1=CABAC;
svc.layer[0].kbps     = 0  // desided bitrate (for this and
below level) Must be greater then kbps for previous level (for
multistream_mode!=1)
svc.layer[0].max_kbps = 0  // max allowed bitrate in vbr mode
for this layer; default - 0 (means not set)
#svc.layer[0].qp_intra = 42 //qp for intra-frames coding
(qp_delta_p and qp_delta_b is used from main settings)(used
for rc.type = 0)

svc.layer[0].speed.i = 4 // speed for I-frames (0..7):
0==slowest... 7 =fastest;
svc.layer[0].speed.p = 4 // speed for P-frames (0..7):
svc.layer[0].speed.b = 4 // speed for B-frames (0..7):
```

```
svc.layer[0].profile_idc = 86 // SVC: 83= Scalable Baseline,
86= Scalable High; AVC: see profile_idc below; MVC: 118=
Multiview High, 128= Stereo High

svc.layer[0].level_idc = 41 //level_idc; 0 - means, that it
will be calculated from settings

# SVC layer specific VUI parameters. See Standard Annex E
section E.2 for details
# Valid with vui.aspect_ratio_info_present_flag = 1 (see VUI
section description below)
svc.layer[0].vui_aspect_ratio_idc = 0 // 0-auto, 1-16-manual
set from Table E-1, 255-Extended_SAR
svc.layer[0].vui_sar_width        = 0 // Extended_SAR width
svc.layer[0].vui_sar_height       = 0 // Extended_SAR height

#slicing params
svc.layer[0].slice.mode = 0
svc.layer[0].slice.param = 0
svc.layer[0].slice.i_param = 0
svc.layer[0].slice.b_param = 0
svc.layer[0].num_mgs_slices = 1 //number of slices to split
coefs (valid only for MGS extend)
svc.layer[0].mgs_coeffs = 0 //How to split coeffs; 0
automatic. example: 0xB73 - means coefs [0-3] into slice0; [4-
7] - slice1; [8-11] - slice2; [12-15] slice3

svc.layer[0].frame_width = 0 // SVC layer frame width. Must be
set only if custom extend is in use
svc.layer[0].frame_height = 0 //SVC layer frame height. Must
be set only if custom extend is in user

#The same default parameters for all other layers

# end svc layers
#
# Life! Don't talk to me about life.
################################################################
##############

#---------------------- general settings
profile_idc  = 77   // H264 profile selection (66=baseline,
77=main, 100=high, 110=high 10, 122 - High422);
level_idc    = 32   // H264 level selection (12=1.2, 32=3.2,
40=4.0); 0 - means, that it will be calculated from settings
```

```
sym_mode     = 1 // symbol mode (0=UVLC, 1=CABAC);

#--- bit depth parametrers are valid for High-bits enabled
build
bit_depth_luma = 8  //bit depth when encoding luma samples (8-
14)
bit_depth_chroma = 8 //bit depth when encoding luma samples
(8-14)


# ---- Bitwise special encoding flags:
#ENC_DISABLE_VUI =    1, ///< don't put vui infromation in sps
#ENC_SLICE_TYPE_012 = 2, ///< encode slice types as 0,1 or 2
(default is 5,6,7)
#ENC_SPS_ONLY_ONCE  = 4, ///< put SPS only for the first frame
of stream
#ENC_REC_POINT_IDR  = 8,  ///< put recovery point SEI for IDR
picture too
#ENC_FRAME_PACKING  = 0x10, ///< used together with
sei.frame_packing flags. Force encoder to perform actual
packing
#ENC_MBS_DATA       = 0x20  ///< encode macroblocks data (used
together with v4e_get_picture_nal_list_and_mbs_data()
function)
enc_flags = 0

#----------------------- SEI flags
sei.pic_timing_flag = 0  // Picture timing and buffering
period SEIs control (0/1/2); 0 - disable; 1 - put all picture
SEIs in one NAL unit; 2 - Put each SEI in separate NAL unit
sei.film_grain_flag = 0 //Calculate parameters and add film-
grain SEI (0/1/2)
sei.post_filter_flag = 0 //Calculate parameters and add
postfiltering SEI (0/1/2)
sei.rec_point_flag = 0 //add recovery point SEI (0/1/2)
sei.frame_packing_flag = 0 //Add frame packing arrangement SEI
(0/1/2)

#----------------------- Film-grain SEI settings
sei.film_grain_mode = 0 // 0 - automatic; 4 -manual, 1,2,3 -
reserved
#next settings are used in manual mode only
sei.film_grain_luma_noise_level = 0
sei.film_grain_luma_max_frequency = 0
sei.film_grain_chroma_noise_level = 0
```

```
sei.film_grain_chroma_max_frequency = 0

#----------------------- Post filter SEI settings
sei.post_filter_mode = 0 //(0/1/2) 0 - 2D filter, 1 -1D
filters, 2 - cross-correlation matrix
sei.post_filter_size = 0 //(0,1,2,3) Actual size is odd and
calculated as (1+2*post_filter_size)

#----------------------- Frame Packing SEI settings
sei.frame_packing_type = 3 //(3/4/5) 3 - side-by-side, 4 -
top-bottom, 5 - temporal-interleaving arrangement

frame_width = 0 // Base layer frame width. Must be set only if
custom extend is in use
frame_height = 0 //Base layer frame height. Must be set only
if custom extend is in use

//////////////////////// interlace coding mode, will be disabled
for baseline profile
// 0 = disabled;
// 1 = all fields, top field first;
// 2 = all fields, bottom field first;
// 3 = MBAFF (mb-level adaptive frame/field coding)
interlace_mode   = 0

//////////////////////// interlace flags
// 0x01 = disable motion estimation from bottom field to top
one;
// 0x02 = encode both fields as intra (only top field is intra
by default);
// 0x04 = show bottom field first when mbaff of frame coding
// 0x08 = force decoder to play frame-encoded stream as
interlaced
// 0x10 = put zero POC offsets for both top & bottom fields
(for mbaff coding)
// 0x20 = RD-opt MBAFF decision
// 0x40 = disable preprocessing for bottom field
// 0x80 = add telecine picture structure for frame-encoded
video
// 0x100 = "3-2" start with 3. Together with INT_BOTTOM_FIRST
defines start position of telecine

interlace_flags = 0 //
```

```
direct_mode     = 0 // direct mode for B frames (0=temporal,
1=spatial)
constrained_intra_pred = 0 //constrained intra prediction flag
(0/1)
chroma_qp_offset = 0 //offset for chroma QP (-26, +26)

weighted_pred_flag = 0 // use weighted prediciton (0/1)

poc_type = 0 //poc_type (see standard). Encoder support only 0
or 2 for Baseline profile

gpu_acceleration = 0 // Use Nvideo GPU : 0 -off; 1 - On

avc_intra_class = 0 // AVC-Intra class encoding 0 - off; 50 -
class 50; 100 - class 100

# Bitwise combination of avc-intra encoding flags:
# AVC_I_FORCE_PANASONIC = 1   // force all features for
Panasonic compatibility
avc_intra_flags = 1 // bitwise combination of avc coding
flags; @see avc_intra_flags_e

#----------------------- Group of pictures (GOP) settings
gop.idr_period   = 1   // period of IDR frames (0=only first,
N=on every N-th I-frame);
gop.keyframes   = 32  // period of I-frames in number of
frames (0=only first, 1=all);
gop.bframes     = 2 // number of B-frames between P (0=no B-
frames);
gop.min_bframes = 0 // minimum number of B-frames for adaptive
mode (if equal to gop.bframes adaptive mode is disable)
gop.emulate_b   = 0 // put non-reference P frames instead of B,
requires non-zero "gop.bframes" value; (0=not used; 1=B-frames
order; 2=natural order);
gop.aud = 0   // enable Access Unit Delimiters (0=disable,
1=enable AUD+PPS, 2=enable AUD only);
gop.num_units = 1000 //together with time_scale define Frame
per Seconds (fps)
#gop.time_scale = 50000 // fps = time_scale/(2*num_units)
gop.min_intra_period = 4 // minimal distance between intra
frames during continuous scene changes (number of frames).
gop.sps_period = 0 ///< How often SPS/PPS is included (used
for coding with keyframes = 0)

# Bitwise gop flag:
```

```
#  1 - encode Hierarchical B-frames (if 3 or more B frames
specified)
#  2 - don't set IDR-slice on schene changes
#  4 - force B frames in odd positions
gop.flags = 0

#----------------------- High-profile settings
frext.transform_8x8 = 0     //using 8x8 transform (0 - off; 1
- adaptive; 2 - 8x8 only; 3 - 8x8 only without Intra16x16)
frext.second_qp_offset = 0  //offset for V-chroma QP (-26,
+26)
frext.scaling_matrix = 0 //Switch on using alternative scaling
matrix
#Use default alternative scaling matrix or custom matrix, if
it is set explicitly.

#----------------------- Rate Control settings
rc.type       = 0   // type of rate control (0=fixed QP, 1=VBR,
2=CBR, 3=CBR+filler);
rc.kbps       = 0   // desired bitrate, kbps;
rc.auto_qp    = 0   // 1=automatic first qp and range
selection, 0=use manual settings;
#rc.qp_intra   = 44 // quant parameter for I-frames (0-51);
rc.qp_delta_p = 0   // base qp delta between I and P (0-51);
rc.qp_delta_b = 0   // base qp delta between P and B (0-51);
rc.qp_min     = 1   // minimum allowed QP for rate control (1-
51);
rc.qp_max     = 51  // maximum allowed QP for rate control (1-
51);
rc.scene_detect   = 0 // scene change detection threshold (0-
100);
rc.vbv_length  = 0  // rate control buffer length in msec; will
be set to default depending on type if 0
rc.qp_modulation = 0  // enable QP variation between
macroblocks (0/1);
rc.mb_update = 0 // enable mb-level rate-control (0/1);
rc.look_ahead = 1       // number of look-ahead frames (0-8)
//Currently only 0 or 1 are used
rc.max_kbps   = 0       // max allowed bitrate in vbr mode;
default - 0 (means not set)
rc.initial_cpb_removal_delay = -1 // Initial fullness of CBR
buffer in 1/90000 sec; default: -1 (means calculated as
90*vbv_length/2)
rc.dual_pass_param = 256 // dual-pass behavior parameter /< 0
- 256; 0 - CBR-like; 256 - "fixed qp"-like
```

```
rc.max_intra_frame_bytes = 0 ///< maximum size of intra frames
in bytes (0 means no restriction)
rc.min_intra_frame_bytes = 0 ///< minimum size of intra frames
in bytes (0 means no restriction)
rc.gop_bytes = 0    ///< size of GOP in bytes (used with
RC_FIXED_GOP_SIZE flag; 0 means will be calulated from kbps or
max_kbps)

# Bitwise rc flag:
#  1 - ignore buffer overflow for VBR coding
#  2 - use qp_delta_b as max for automatically calulated
delta_b
#  4 - use qp_delta_b as min for automatically calulated
delta_b
#    0x10 - add filler NALs (can be set in vbr mode if
max_kbps is specified)
#    0x20 - put cbr_flag into sps (effective only with flag
above)
#    0x40 - support fixed number of bytes in GOP
rc.flags = 0

#----------------------- Motion estimation settings
me.subdiv   = 7     // macroblock subdivision mask (1=16x16,
2=16x8, 4=8x16,8=8x8, 16=8x4, 32=4x8, 64=4x4); small subdivs
currently not used
me.search_range = -1 // maximum search range in full pels (1-
64); -1 means, that it will be calculated from picture size
me.max_refs = 1  // number of pictures (frames or fields) used
for ``motion search (1-5);

# Bitwise gop flags:
#  0x10 = Set num_refs for B-frames to (1,1) even if max_refs >
1
#  0x20 = disable preproc motion estimation by reduced picture
(experimental, not recomended)
#  0x40 = Disable preproc complexity calculation
(experimental, not recomended)
#  0x1000 = Use more detailed motion estimation for P frames
#  0x2000 = Use more detailed motion estimation for B frames
me.flags = 0

#----------------------- speed mode selection
speed.i = 4 // speed for I-frames (0..8): 0==slowest... 8
=fastest;
speed.p = 4 // speed for P-frames (0..8):
```

```
speed.b = 4 // speed for B-frames (0..8):
speed.automatic = 0 // enables automatic real time control (for
capture) (0/1)

#----------------------- Slicing settings
slice.mode  = 0     // select slice mode (0=none, 1=#mbs per
slice, 2=#bytes per slice; 3=#slices)
slice.param = 0     // provide appropriate number for
slice.mode;
slice.i_param = 0     // provide appropriate number for
slice.mode for I-slices. If 0 slice.param is used;
slice.b_param = 0     // provide appropriate number for
slice.mode for B-slices. If 0 slice.param is used;

#----------------------- Deblocking filter settings
deblock.flag     = 0  // Configure loop filter (0=parameter
below ingored, 1=parameters sent)
deblock.disable   = 0  // Disable loop filter in slice header
(0=Filter, 1=No Filter)
deblock.alpha_c0 = 0  // Alpha & C0 offset div. 2, {-6, -5,
... 0, +1, .. +6}
deblock.beta_c0  = 0  // Beta offset div. 2, {-6, -5, ... 0,
+1, .. +6}

#----------------------- multi-threading settings
mt.disable     = 0  // flag to disable multithreading
mt.num_threads = 0  // select a number of worker threads to
run, 0 means autoconfigure;

//params below will be calculated automatcally if set to -1
mt.max_pict_tasks = -1 // max number of simultaneously coded
picture [0,5]; <= 0 measn that it will be set automatically

# ----- Max value of frames to hold in async-feed encoding
mt.max_raw_frames = 0  //0 - calulate automaticaly; >0 force
this value


// ERROR RESILIENCE
////////////////////////////////////////////////////////////
    // Enable error resilience.
    // (If zero, then no special error resilience features
will be enabled and
    // there will be no possibility to enable error resilience
on-the-fly.)
```

```
////////////////////////////////////////////////////////////////
er.enable = 0

////////////////////////////////////////////////////////////////
    // The initial expected loss rate in percents (no special
error resilience if 0)

////////////////////////////////////////////////////////////////
er.initial_expected_loss_percent = 15

////////////////////////////////////////////////////////////////
    // Intra update method:
    //
    // 0 - don't use intra update
    // 1 - adaptive intra update by distinguishing the motion
areas on picture (recommended)
    // 2 - update by one or more horizontal circular intra
macroblock lines
    // 3 - update by inserting intra mb in random positions
(the closer to the center, the more)
    //
    // Sensative to the value of loss rate - the more is this
value,
    // the more intensively the intra update is performed.

////////////////////////////////////////////////////////////////
er.intra_update_method = 1


////////////////////////////////////////////////////////////////
    // Short temporal period for intra updating the
macroblocks with relatively high motion
    //
    // Parameter for adaptive intra update method based on
distinguishing motion areas.
    // (Ignored if intra_update_method != 1)
    //
    // 0 - don't use short update
    // Recommended (nonzero) values - {1,2,3}.

////////////////////////////////////////////////////////////////
er.fast_motion_update_period = 1
```

```
////////////////////////////////////////////////////////////////
    // Long temporal period for intra updating the macroblocks
with both high and slow motion
    //
    // Parameter for adaptive intra update method based on
distinguishing motion areas.
    // (Ignored if intra_update_method != 1)
    //
    // 0 - don't use long update
    // Recommended (nonzero) values - from 5 to the half of
keyframe interval.

////////////////////////////////////////////////////////////////
er.full_motion_update_period = 6


////////////////////////////////////////////////////////////////
    // The temporal period for picture full intra update
    // (Works only if "er.enable = 1" and
"er.initial_expected_loss_percent > 0"

////////////////////////////////////////////////////////////////
er.total_intra_update_period = 60


#----------------------- Video usability information (VUI) ---
# VUI parameters are placed into SPS.
# See Standard Annex E section E.2 for details
#-----------------------------------------------------------
vui.aspect_ratio_info_present_flag = 0
vui.aspect_ratio_idc = 0
vui.sar_width = 0
vui.sar_height = 0
vui.overscan_info_present_flag = 0
vui.overscan_appropriate_flag = 0
vui.video_signal_type_present_flag = 0
vui.video_format = 0
vui.video_full_range_flag = 0
vui.colour_description_present_flag = 0
vui.colour_primaries = 0
vui.transfer_characteristics = 0
vui.matrix_coefficients = 0
vui.chroma_loc_info_present_flag = 0
vui.chroma_sample_loc_type_top_field = 0
```

```
vui.chroma_sample_loc_type_bottom_field = 0
vui.timing_info_present_flag = 0
vui.fixed_frame_rate_flag = 0
vui.nal_hrd_parameters_present_flag = 0
vui.vcl_hrd_parameters_present_flag = 0
vui.low_delay_hrd_flag = 0
vui.pic_struct_present_flag = 0
vui.bitstream_restriction_flag = 0
vui.motion_vectors_over_pic_boundaries_flag = 255
vui.max_bytes_per_pic_denom = 255
vui.max_bits_per_mb_denom = 255
vui.log2_max_mv_length_vertical = 255
vui.log2_max_mv_length_horizontal = 255
vui.num_reorder_frames = 255
vui.max_dec_frame_buffering = 255
```

bSoft encoder:

```
Vx0Enc.exe H264 MOTION ON OFF OFF 1 2 1080P 1080P 1080P
1080P 25 1 250 32 1 ON 29 29 29 29 0 0 0 0 HIGH HIGH
"file.yuv" "file.yuv" "file.yuv" "file.yuv" "file.vh4" 0 0
0 0 "rec_L0.yuv" "rec_L1.yuv" "rec_L2.yuv" "rec_L3.yuv" OFF
```

The command line parameters are to be interpreted as:

```
Vx0Enc Standard CodingMode SVC Bidir Slice
SpatialLayersNumber MgsLayersNumber FramesizeDid0
FramesizeDid1 FramesizeDid2 FramesizeDid3 Framerate
Frameskip Pictures Intrarate GOPSize
ResidualUpsampling QuantCgsLayer0 QuantCgsLayer1
QuantCgsLayer2 QuantCgsLayer3 BitrateCgsLayer0
BitrateCgsLayer1 BitrateCgsLayer2 BitrateCgsLayer3
FullPixel HalfPixel FilePicOrigDid0 FilePicOrigDid1
FilePicOrigDid2 FilePicOrigDid3 FileBit FilterFlag
FilterIdc AlfaC0Offset BetaOffset FilePicEncDid0
FilePicEncDid1 FilePicEncDid2 FilePicEncDid3 Dash
```

Annex C – Additional SVC Rate-Distortion Performance Results

This Annex provides additional RD performance results for various SVC configurations discussed in Sections 3.4 and 3.5.

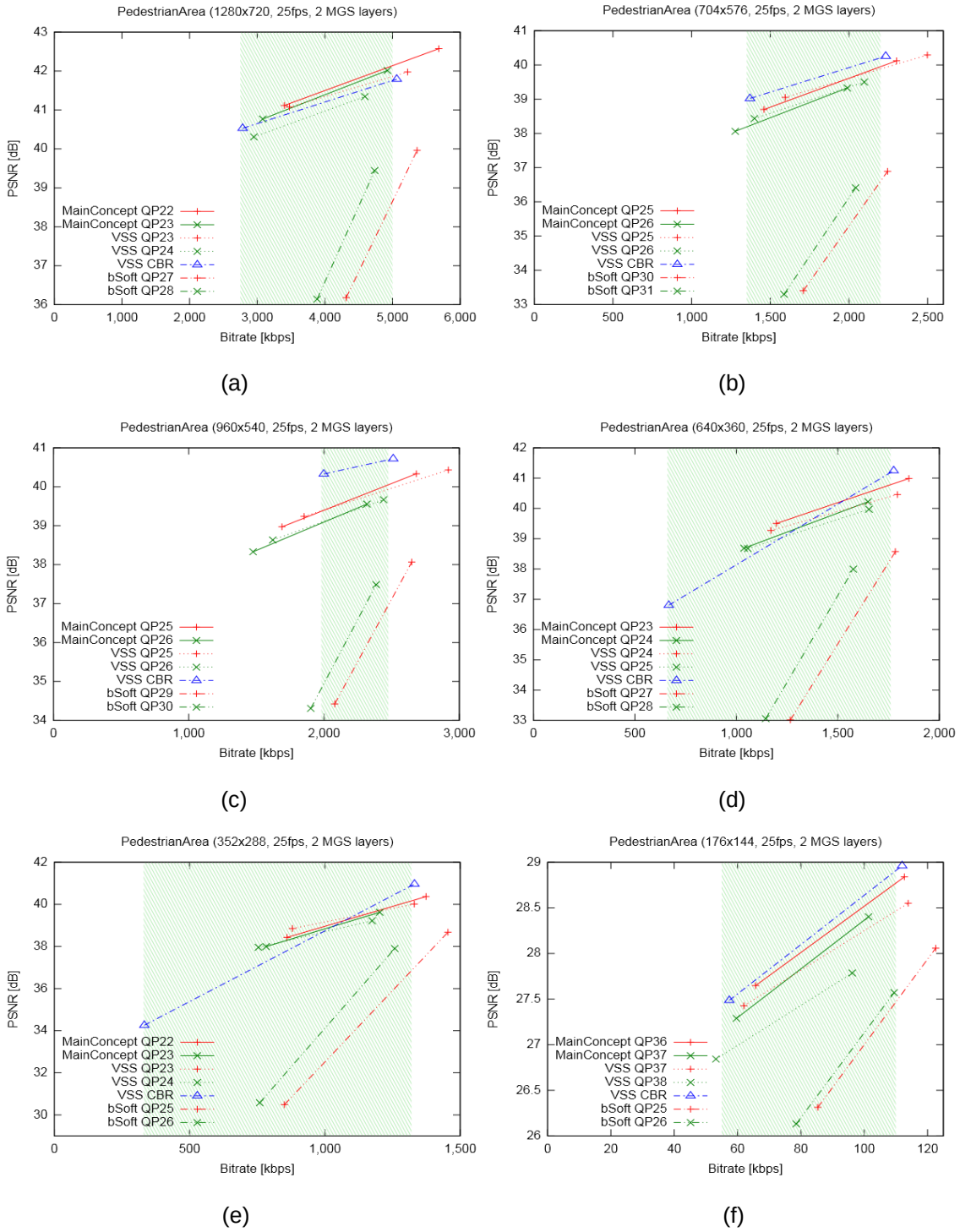


Figure 77: PSNR results of rate control modes for different encoders for the *PedestrianArea* sequence at (a) 1280x720, (b) 704x576, (c) 960x540, (d) 640x360, (e) 352x288, and (f) 176x144 resolutions.

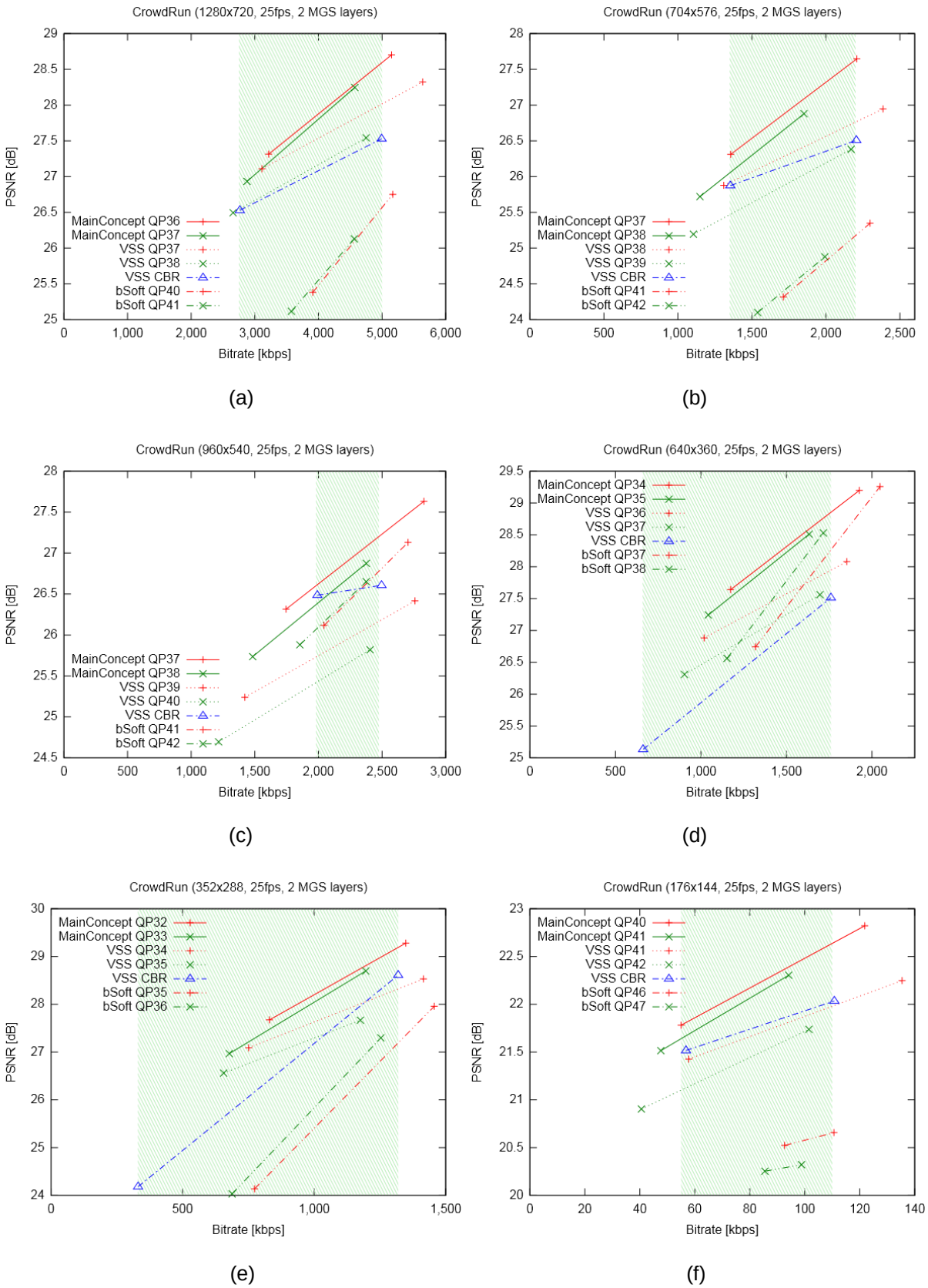


Figure 78: PSNR results of rate control modes for different encoders for the *CrowdRun* sequence at (a) 1280x720, (b) 704x576, (c) 960x540, (d) 640x360, (e) 352x288, and (f) 176x144 resolutions.

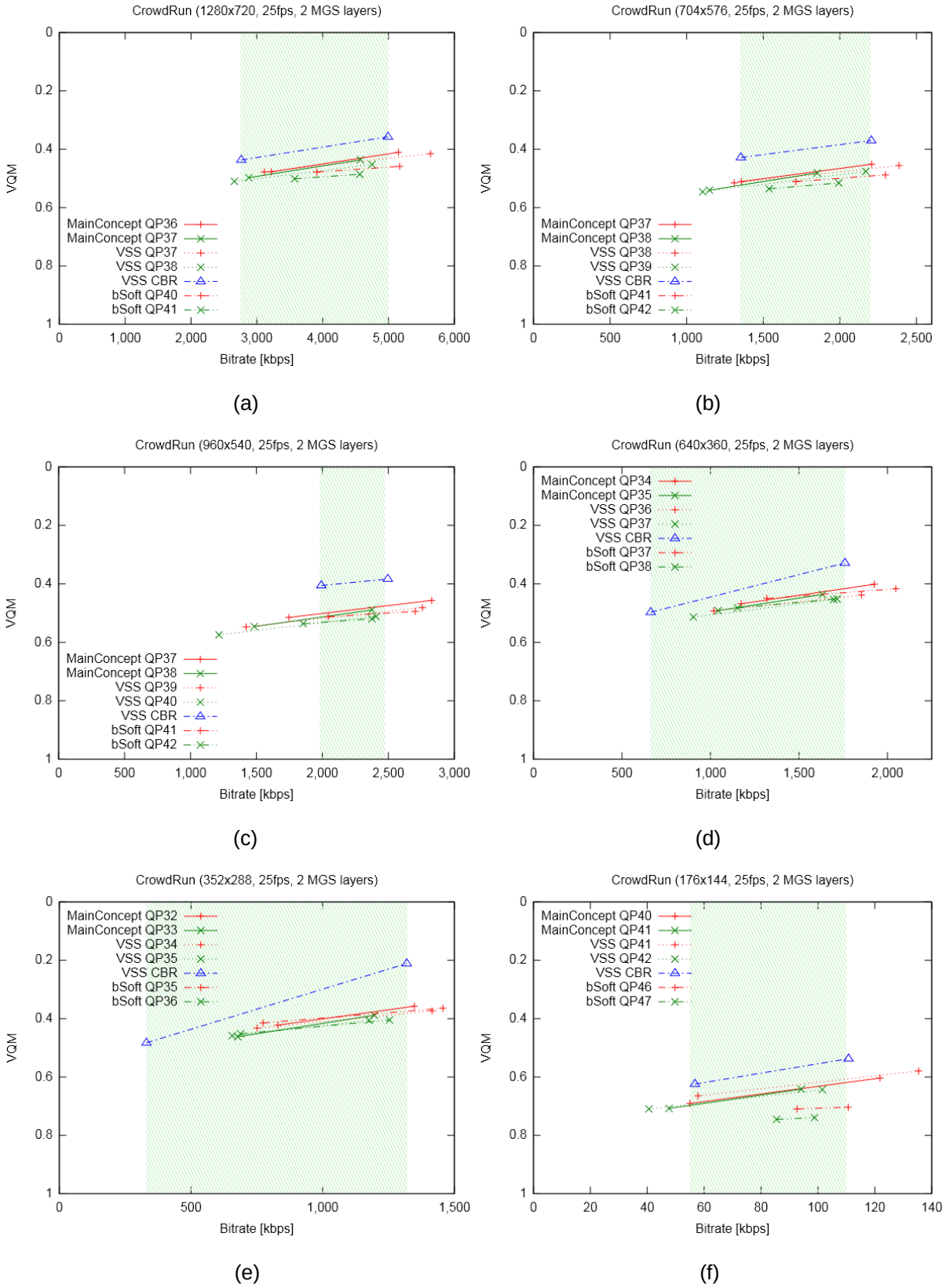


Figure 79: VQM results of rate control modes for different encoders for the *CrowdRun* sequence at (a) 1280x720, (b) 704x576, (c) 960x540, (d) 640x360, (e) 352x288, and (f) 176x144 resolutions.

Annex D – SVC Decoding and Transcoding Speeds

This Annex provides test results for decoding and transcoding speeds. The tests were performed on a ThinkPad T510 notebook with an Intel Core i7-620M 2.67 GHz dual core processor and 4 GB RAM running Ubuntu 11.04. SVC decoding and SVC-to-AVC transcoding for the bSoft decoder/transcoder was tested. Additionally, the decoder output was piped into FFmpeg for encoding to MPEG-2. (Due to software issues, the combination of the SVC-to-AVC transcoding with FFmpeg-based AVC-to-MPEG-2 transcoding is not included in the test results.) The *PedestrianArea* sequence at a frame rate of 25 fps with 4 MGS quality layers was used at several resolutions ranging from 1920x1080 to 176x144.

Test results are shown in Figure 80. Note that the y-axis is in log-scale. Resolutions 4CIF, CIF and QCIF are shown as coupled due to their dyadic spatial relation. Decoding speeds above 37.5 fps (i.e., 150% of the native 25 fps frame rate) are expected to provide robust real-time decoding/transcoding (indicated by green background). The range between 25 and 37.5 fps is considered unstable (indicated by yellow background) as fluctuations in coding complexity or interfering processes can easily cause disruptions.

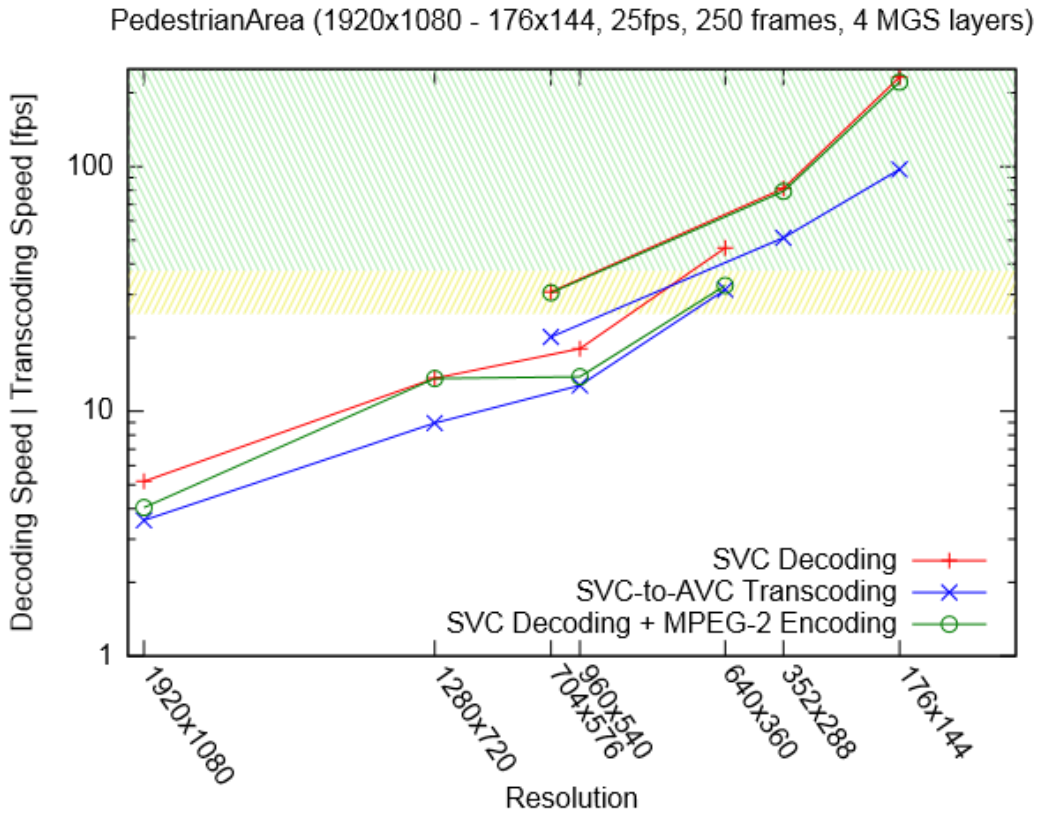


Figure 80: Decoding speeds for the bSoft decoder/transcoder in combination with the FFmpeg encoder.

Annex E – Generation of Local MPD

This Annex describes the generation of a *local MPD* by the DASH proxy on an ALICANTE Home-Box. For further details, the interested reader is referred to [9].

The session initialization for DASH is performed as follows:

1. The end-user terminal sends an HTTP request for the *local MPD* to the Home-Box.
2. The DASH proxy at the Home-Box downloads the *remote MPD* (shown in Listing 3) from the server.
3. The DASH proxy transforms the remote MPD into a local MPD via XSLT. An example of a local MPD is shown in Listing 4. The `BaseUrl` element of the local MPD points to the Home-Box ("192.168.0.2" in the example) and contains an identifier of the streaming session ("42" in the example).
4. The DASH proxy responds to the initial HTTP request by sending the local MPD to the end-user terminal.

The *local MPD* comprises only one representation. The DASH proxy performs SNR adaptation transparent for terminal.

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:mpeg:dash:schema:mpd:2011"
xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011
http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
DASH_schema_files/DASH-MPD.xsd" minBufferTime="PT10.00S"
mediaPresentationDuration="PT3256S" type="static"
profiles="urn:mpeg:dash:profile:isoff-main:2011">
  <BaseURL>http://example.com/</BaseURL>
  <Period>
    <AdaptationSet>
      <Representation mimeType="video/H264-SVC"
codecs="avc1.644028, svc1" width="352" height="288"
frameRate="25" id="0" bandwidth="128000">
        <SegmentList duration="10">
          <Initialization sourceURL="seg-L0-init.svc"/>
          <SegmentURL media="seg-L0-1.svc"/>
          <SegmentURL media="seg-L0-2.svc"/>
          <SegmentURL media="seg-L0-3.svc"/>
        </SegmentList>
      </Representation>
      <Representation mimeType="video/H264-SVC"
codecs="avc1.644028, svc1" width="352" height="288"
frameRate="25" id="1" dependencyId="0" bandwidth="256000">
```

```
        <SegmentList duration="10">
          <Initialization sourceURL="seg-L1-init.svc"/>
          <SegmentURL media="seg-L1-1.svc"/>
          <SegmentURL media="seg-L1-2.svc"/>
          <SegmentURL media="seg-L1-3.svc"/>
        </SegmentList>
      </Representation>
      <Representation mimeType="video/H264-SVC"
codecs="avc1.644028, svc1" width="352" height="288"
frameRate="25" id="2" dependencyId="0 1" bandwidth="512000">
        <SegmentList duration="10">
          <Initialization sourceURL="seg-L2-init.svc"/>
          <SegmentURL media="seg-L2-1.svc"/>
          <SegmentURL media="seg-L2-2.svc"/>
          <SegmentURL media="seg-L2-3.svc"/>
        </SegmentList>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Listing 3: Example of remote MPD with 3 SVC layers, adopted from [9].

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:mpeg:dash:schema:mpd:2011"
xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011
http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
DASH_schema_files/DASH-MPD.xsd" minBufferTime="PT10.00S"
mediaPresentationDuration="PT3256S" type="static"
profiles="urn:mpeg:dash:profile:isoff-main:2011">
  <BaseURL>http://192.168.0.2/session/42/</BaseURL>
  <Period>
    <AdaptationSet>
      <Representation mimeType="video/H264" codecs="avc1"
id="0" bandwidth="512000">
        <SegmentList duration="10">
          <Initialization sourceURL="seg-init.264"/>
          <SegmentURL media="seg-1.264"/>
          <SegmentURL media="seg-2.264"/>
          <SegmentURL media="seg-3.264"/>
        </SegmentList>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Listing 4: Example of generated local MPD with AVC segments, adopted from [9].

Annex F – Questionnaire for the Subjective Evaluation of Representation Switch Smoothing

The following questionnaire was given to the participants of the subjective tests on *representation switch smoothing* described in Section 5.4.3.3.

Questionnaire

Date: _____

Participant Id: ____ Sex: ☐ male ☐ female Age: ____

Thank you for participating in this study on the perception of quality changes in videos.

You will be shown two different video sequences. Each sequence is given in two versions (denoted *a* and *b*).

For each video sequence, please state below which version you prefer.

You may start with either version. You may watch each version as often as you wish.

Sequence 1:

Preferred version:

☐ Version *a* ☐ Version *b* ☐ No difference

Sequence 2:

Preferred version:

☐ Version *a* ☐ Version *b* ☐ No difference

Thank you for your participation!

Annex G – Adaptation Logic Implementation for MPEG-21 ADTE

This Annex provides the MPEG-21 DIA standard-conforming implementation of the adaptation logic discussed in Section 5.4.2. The adaptation logic is implemented via MPEG-21 AQoS and UCD XML documents and has been adjusted to work with the MPEG-21 ADTE implementation of [251]. The AQoS and UCD files are shown in Listing 5 and Listing 6 respectively. An example of the UED is shown in Listing 7.

```
<?xml version="1.0" encoding="UTF-8"?>
<DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mpeg:mpeg21:2003:01-DIA-NS
http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
21_schema_files/dia-2nd/AQoS-2nd.xsd">
  <DescriptionMetadata>
    <ClassificationSchemeAlias alias="AQoS"
href="urn:mpeg:mpeg21:2003:01-DIA-AdaptationQoSCS-NS"/>
    <ClassificationSchemeAlias alias="MEI"
href="urn:mpeg:mpeg21:2003:01-DIA-MediaInformationCS-NS"/>
  </DescriptionMetadata>
  <Description xsi:type="AdaptationQoSType">
    <!-- SVC adaptation parameters -->
    <Module xsi:type="UtilityFunctionType">
      <Constraint iOPinRef="Bandwidth">
        <Values xsi:type="IntegerVectorType">
          <Vector>270000 500000 1080000 1950000</Vector>
        </Values>
      </Constraint>
      <AdaptationOperator iOPinRef="QualityLayer">
        <Values xsi:type="IntegerVectorType">
          <Vector>0 1 2 3</Vector>
        </Values>
      </AdaptationOperator>
      <AdaptationOperator iOPinRef="SpatialLayer">
        <Values xsi:type="IntegerVectorType">
          <Vector>0 0 0 0</Vector>
        </Values>
      </AdaptationOperator>
      <AdaptationOperator iOPinRef="TemporalLayer">
        <Values xsi:type="IntegerVectorType">
          <Vector>0 0 0 0</Vector>
        </Values>
      </AdaptationOperator>
      <AdaptationOperator iOPinRef="PriorityID">
        <Values xsi:type="IntegerVectorType">
```

```
          <Vector>0 0 0 0</Vector>
        </Values>
      </AdaptationOperator>
      <AdaptationOperator iOPinRef="ResWidth">
        <Values xsi:type="IntegerVectorType">
          <Vector>352 352 352 352</Vector>
        </Values>
      </AdaptationOperator>
      <AdaptationOperator iOPinRef="ResHeight">
        <Values xsi:type="IntegerVectorType">
          <Vector>288 288 288 288</Vector>
        </Values>
      </AdaptationOperator>
      <Utility iOPinRef="Layer">
        <Values xsi:type="IntegerVectorType">
          <Vector>0 1 2 3</Vector>
        </Values>
      </Utility>
    </Module>
    <Module xsi:type="LookUpTableType">
      <Axis iOPinRef="PacketLoss">
        <AxisValues xsi:type="FloatVectorType">
          <!-- Note: Lowest value must be 0.0! -->
          <Vector>0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08
0.09 0.1 0.11 0.12 0.13 0.14 0.15 0.16 0.17 0.18 0.19
1.0</Vector>
        </AxisValues>
      </Axis>
      <Content iOPinRef="PacketLossBasedBandwidthEstimate">
        <ContentValues xsi:type="IntegerMatrixType"
mpeg7:dim="27">
          <Matrix>1000000 1050000 1100000 1150000 1200000
1250000 1300000 1350000 1400000 1450000 1500000 1550000
1600000 1650000 1700000 1750000 1800000 1850000 1900000
1950000 2000000</Matrix>
        </ContentValues>
      </Content>
    </Module>
    <!-- Copy of the above LookUpTable, only for max packet
loss. -->
    <Module xsi:type="LookUpTableType">
      <Axis iOPinRef="MaxPacketLoss">
        <AxisValues xsi:type="FloatVectorType">
          <Vector>0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08
0.09 0.1 0.11 0.12 0.13 0.14 0.15 0.16 0.17 0.18 0.19
1.0</Vector>
        </AxisValues>
      </Axis>
      <Content iOPinRef="PacketLossBasedMaxBandwidth">
        <ContentValues xsi:type="FloatMatrixType"
mpeg7:dim="27">
```

```
          <!-- Workaround: Needs to be Float, otherwise the
ADTE fails -->
          <Matrix>1000000 1050000 1100000 1150000 1200000
1250000 1300000 1350000 1400000 1450000 1500000 1550000
1600000 1650000 1700000 1750000 1800000 1850000 1900000
1950000 2000000</Matrix>
        </ContentValues>
      </Content>
    </Module>
    <IOPin id="Layer"/>
    <IOPin id="SpatialLayer" discrete="true"
semantics=":AQoS:1.3.9.1"/>
    <IOPin id="TemporalLayer" discrete="true"
semantics=":AQoS:1.3.9.2"/>
    <IOPin id="QualityLayer" discrete="true"
semantics=":AQoS:1.3.9.4"/>
    <IOPin id="PriorityID" discrete="true"
semantics=":AQoS:1.3.9.5"/>
    <IOPin id="Bandwidth" discrete="true" semantics=":MEI:6"/>
    <IOPin id="ResWidth" discrete="true" semantics=":MEI:17"/>
    <IOPin id="ResHeight" discrete="true"
semantics=":MEI:18"/>
    <IOPin id="MaxPacketLoss"/>
    <IOPin id="PacketLossBasedMaxBandwidth"/>
    <IOPin id="PacketLoss" discrete="false">
      <GetValue xsi:type="SemanticalDataRefType"
semantics=":AQoS:6.6.5.7"/>
    </IOPin>
    <IOPin id="PacketLossBasedBandwidthEstimate"/>
  </Description>
</DIA>
```

Listing 5: Example of AQoS.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
        <DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mpeg:mpeg21:2003:01-DIA-NS
http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
21_schema_files/dia-2nd/UCD-2nd.xsd">
          <DescriptionMetadata>
            <ClassificationSchemeAlias alias="SFO"
href="urn:mpeg:mpeg21:2003:01-DIA-StackFunctionOperatorCS-
NS"/>
            <ClassificationSchemeAlias alias="AQoS"
href="urn:mpeg:mpeg21:2003:01-DIA-AdaptationQoSCS-NS"/>
            <ClassificationSchemeAlias alias="MEI"
href="urn:mpeg:mpeg21:2003:01-DIA-MediaInformationCS-NS"/>
          </DescriptionMetadata>
          <Description xsi:type="UCDType">
            <AdaptationUnitConstraints>
```

```
              <!-- AQoS selected bandwidth &gt;= min bandwidth
-->
              <LimitConstraint>
                <!-- AQoS selected bandwidth -->
                <Argument xsi:type="SemanticalRefType"
semantics=":MEI:6"/>
                <!-- min bandwidth -->
                <Argument xsi:type="ConstantDataType">
                  <Constant xsi:type="IntegerType">
                    <Value>
                      0
                    </Value><!-- Minimum guaranteed bandwidth
from the SLA -->
                  </Constant>
                </Argument>
                <!-- &gt;= -->
                <Operation operator=":SFO:39"/>
              </LimitConstraint>

              <!-- max bandwidth capacity &gt;= AQoS selected
bandwidth  -->
              <LimitConstraint>
                <!-- max bandwidth capacity  -->
                <Argument xsi:type="SemanticalDataRefType"
semantics=":AQoS:6.6.4.1"/>
                <!-- AQoS selected bandwidth -->
                <Argument xsi:type="SemanticalRefType"
semantics=":MEI:6"/>
                <!-- &gt;= -->
                <Operation operator=":SFO:39"/>
              </LimitConstraint>

              <!-- MaxPacketLoss == fixed Value -->
              <LimitConstraint>
                <Argument xsi:type="ExternalIOPinRefType"
iOPinRef="#MaxPacketLoss"/>
                <Argument xsi:type="ConstantDataType">
                  <Constant xsi:type="FloatType">
                    <Value>
                      0.15
                    </Value><!-- The highest value from
document('AQoS.xml')//dia:Axis[@iOPinRef='MaxPacketLoss']//dia
:Vector that is less than or equal maxPacketLoss from the SLA
-->
                    <!-- Max packet loss must be rounded down
to the next tested value -->
                  </Constant>
                </Argument>
                <!-- == -->
                <Operation operator=":SFO:11"/>
              </LimitConstraint>
```

```
              <!-- Packet Loss-based Bandwidth estimation -->
              <!-- f:=(UED measured bandwidth * packet loss
based max bandwidth) / packet loss based bandwidth estimate;
Constraint: max(f, min bandwidth) &gt;= AQoS selected
bandwidth -->
              <LimitConstraint>
                <!-- UED measured bandwidth -->
                <Argument xsi:type="SemanticalDataRefType"
semantics=":AQoS:6.6.5.2"/>
                <!-- avoid edge case where UED measured
bandwidth is 0 -->
                <Argument xsi:type="ConstantDataType">
                  <Constant xsi:type="IntegerType">
                    <Value>
                      1
                    </Value><!-- Assumption: 1 bps is close
enough to nothing. -->
                  </Constant>
                </Argument>
                <!-- max(a,b) -->
                <Operation operator=":SFO:20"/>
                <!--packet loss based max bandwidth -->
                <Argument xsi:type="ExternalIOPinRefType"
iOPinRef="#PacketLossBasedMaxBandwidth"/>
                <!-- * -->
                <Operation operator=":SFO:18"/>
                <!-- packet loss based bandwidth estimate -->
                <Argument xsi:type="ExternalIOPinRefType"
iOPinRef="#PacketLossBasedBandwidthEstimate"/>
                <!-- / -->
                <Operation operator=":SFO:19"/>
                <!-- min bandwidth -->
                <Argument xsi:type="ConstantDataType">
                  <Constant xsi:type="IntegerType">
                    <Value>
                      270000
                    </Value><!-- The lowest value from
document('AQoS.xml')//dia:Constraint[@iOPinRef='Bandwidth']//d
ia:Vector that is greater than or equal min bandwidth from the
SLA -->
                    <!-- Min bandwidth must be rounded up to
the next SVC layer bitrate -->
                  </Constant>
                </Argument>
                <!-- max(a,b) -->
                <Operation operator=":SFO:20"/>
                <!-- AQoS selected bandwidth -->
                <Argument xsi:type="SemanticalRefType"
semantics=":MEI:6"/>
                <!-- &gt;= -->
                <Operation operator=":SFO:39"/>
              </LimitConstraint>
```

```
              <!-- Resolution Horizontal -->
              <!-- media horizontal resolution &gt;=  ucd
display resolution-->
              <LimitConstraint>
                <!-- AQoS - media h-res -->
                <Argument xsi:type="SemanticalRefType"
semantics=":MEI:17"/>
                <Argument xsi:type="ConstantDataType">
                  <Constant xsi:type="IntegerType">
                    <Value>
                      352
                    </Value><!-- Resolution requirements from
the SLA -->
                  </Constant>
                </Argument>
                <!-- &gt;= -->
                <Operation operator=":SFO:39"/>
              </LimitConstraint>

              <!-- UED resolution &gt;= media resolution -->
              <LimitConstraint>
                <!-- UED: available h-res -->
                <Argument xsi:type="SemanticalDataRefType"
semantics=":AQoS:6.5.9.1"/>
                <Argument xsi:type="SemanticalRefType"
semantics=":MEI:17"/>
                <!-- &gt;= -->
                <Operation operator=":SFO:39"/>
              </LimitConstraint>

              <!-- Resolution Vertical -->
              <!-- media vertical resolution &gt;=  ucd
display resolution-->
              <LimitConstraint>
                <!-- AQoS - media v-res -->
                <Argument xsi:type="SemanticalRefType"
semantics=":MEI:18"/>
                <Argument xsi:type="ConstantDataType">
                  <Constant xsi:type="IntegerType">
                    <Value>
                      288
                    </Value><!-- Resolution requirements from
the SLA -->
                  </Constant>
                </Argument>
                <!-- &gt;= -->
                <Operation operator=":SFO:39"/>
              </LimitConstraint>

              <!-- UED resolution &gt;= media resolution -->
              <LimitConstraint>
```

```
                <!-- UED: available v-res -->
                <Argument xsi:type="SemanticalDataRefType"
semantics=":AQoS:6.5.9.2"/>
                <Argument xsi:type="SemanticalRefType"
semantics=":MEI:18"/>
                <!-- &gt;= -->
                <Operation operator=":SFO:39"/>
              </LimitConstraint>

              <OptimizationConstraint optimize="maximize">
                <!-- AQoS selected layer -->
                <Argument xsi:type="ExternalIOPinRefType"
iOPinRef="#Layer"/>
              </OptimizationConstraint>
              <OptimizationConstraint optimize="maximize">
                <!-- AQoS - media h-res -->
                <Argument xsi:type="SemanticalRefType"
semantics=":MEI:17"/>
              </OptimizationConstraint>
              <OptimizationConstraint optimize="maximize">
                <!-- AQoS - media v-res -->
                <Argument xsi:type="SemanticalRefType"
semantics=":MEI:18"/>
              </OptimizationConstraint>
            </AdaptationUnitConstraints>
          </Description>
        </DIA>
```

Listing 6: Example of UCD.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mpeg:mpeg21:2003:01-DIA-NS
http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
21_schema_files/dia-2nd/UED-2nd.xsd">
  <Description xsi:type="UsageEnvironmentType">
    <UsageEnvironmentProperty xsi:type="TerminalsType">
      <Terminal>
        <TerminalCapability xsi:type="CodecCapabilitiesType">
          <Decoding xsi:type="VideoCapabilitiesType">
            <Format href="urn-x:alicante:codec:avc">
              <mpeg7:Name>AVC</mpeg7:Name>
            </Format>
            <!-- AVC -->
          </Decoding>
        </TerminalCapability>
        <TerminalCapability xsi:type="DisplaysType">
          <Display xsi:type="DisplayType">
```

```
            <DisplayCapability
xsi:type="DisplayCapabilityType" colorCapable="true">
              <Mode>
                <Resolution horizontal="352" vertical="288"
activeResolution="true"/>
              </Mode>
            </DisplayCapability>
          </Display>
        </TerminalCapability>
      </Terminal>
    </UsageEnvironmentProperty>
    <UsageEnvironmentProperty xsi:type="NetworksType">
      <Network xsi:type="NetworkType">
        <NetworkCharacteristic
xsi:type="NetworkCapabilityType" maxCapacity="100000000"/>
        <NetworkCharacteristic
xsi:type="NetworkConditionType">
          <AvailableBandwidth maximum="NETWORK_CURMAX"/>
          <!-- updated by monitoring -->
          <Error packetLossRate="NETWORK_CURPLOSS"/>
          <!-- updated by monitoring -->
        </NetworkCharacteristic>
      </Network>
    </UsageEnvironmentProperty>
  </Description>
</DIA>
```

Listing 7: Example of UED.xml.

Annex H – SVC-to-AVC Transcoder Rate-Distortion Performance Results

This Annex provides RD performance results for the bSoft fast SVC-to-AVC transcoder deployed in test-bed setup described in Section 5.5.

The RD performance was evaluated for the *Foreman*, *Container*, *Hall_Monitor*, and *Stefan* test sequences (resolution: 352x288, frame rate: 25 fps). Each sequence was encoded to SVC with the bSoft encoder with 4 MGS layers, fixed-QP rate control, and an I-frame period of 32. The RD results for fast SVC-to-AVC transcoding are shown in Figure 81. For reference, the RD results for the SVC bitstream and the pixel-domain transcoding (i.e., full decoding, full re-encoding to AVC) are also shown.

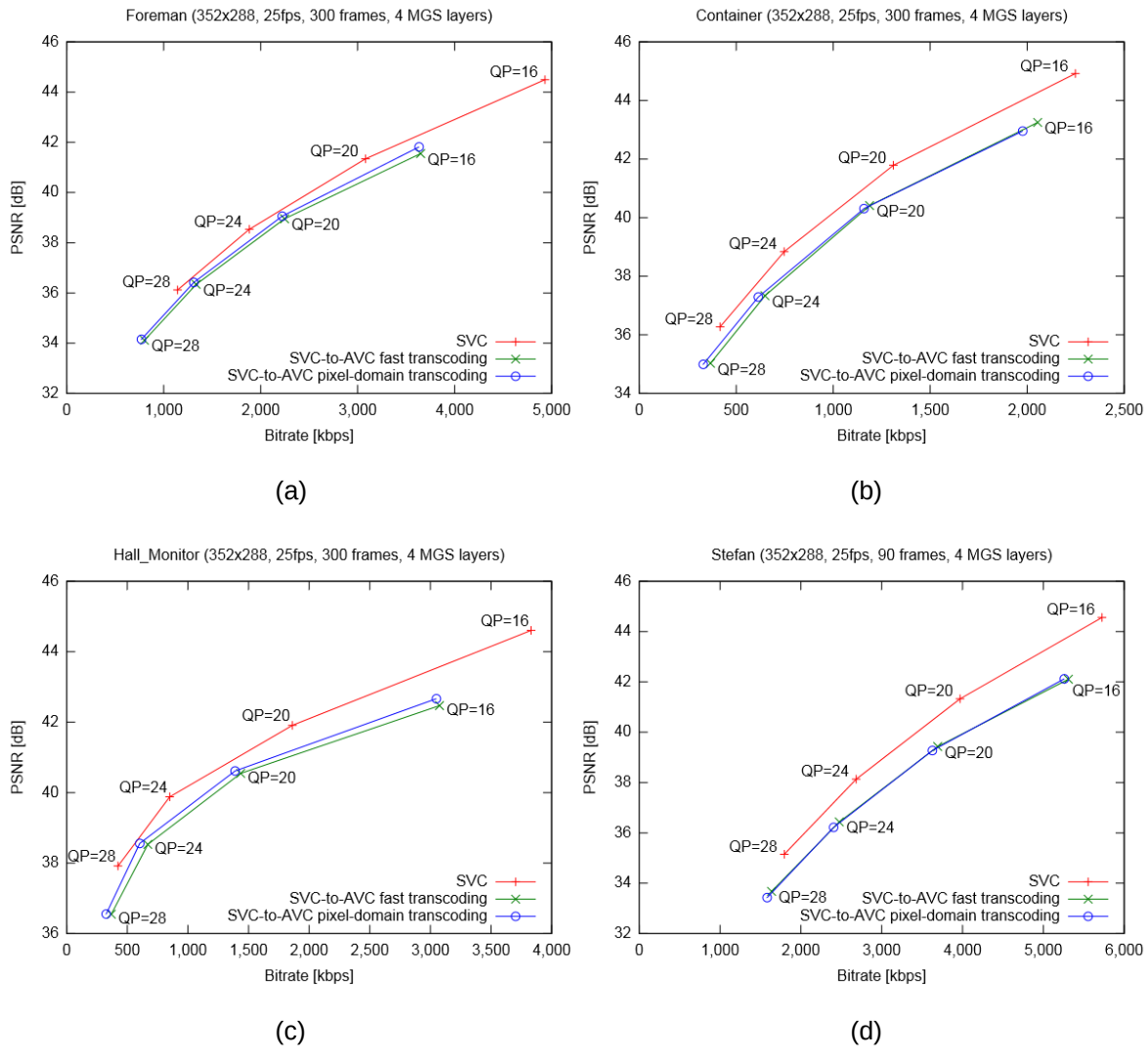


Figure 81: Rate-distortion results for fast SVC-to-AVC transcoding for (a) *Foreman*, (b) *Container*, (c) *Hall_Monitor*, and (d) *Stefan* sequences.

List of Figures

List of Tables

List of Listings

Bibliography

[1] M. Grafl, C. Timmerer, H. Hellwagner, W. Cherif, D. Négru, and S. Battista, "Scalable Video Coding Guidelines and Performance Evaluations for Adaptive Media Delivery of High Definition Content", in *Proceedings of the 18th IEEE International Symposium on Computers and Communication* (ISCC), Split, Croatia, July 2013.

[2] M. Grafl, C. Timmerer. H. Hellwagner, W. Cherif, A. Ksentini, "Hybrid Scalable Video Coding for HTTP-based Adaptive Media Streaming with High-Definition Content", in *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks* (WoWMoM), Madrid, Spain, June 2013.

[3] M. Grafl, C. Timmerer, and H. Hellwagner, "Quality impact of Scalable Video Coding tunneling for Media-Aware content delivery", in *Proceedings of the IEEE International Conference on Multimedia and Expo* (ICME), pp. 1–6, July 2011.

[4] M. Grafl, "SVC tunneling for media-aware content delivery: Impact on video quality", in *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks* (WoWMoM), Lucca, Italy, pp. 1–3, June 2011.

[5] M. Grafl, C. Timmerer, M. Waltl, G. Xilouris, N. Zotos, D. Renzi, S. Battista, and A. Chernilov, "Distributed adaptation decision-taking framework and Scalable Video Coding tunneling for edge and in-network media adaptation", in *Proceedings of the International Conference on Telecommunications and Multimedia* (TEMU 2012), August 2012.

[6] C. Timmerer, M. Grafl, H. Hellwagner, D. Négru, E. Borcoci, D. Renzi, A.-L. Mevel, and A. Chernilov, "Scalable Video Coding in Content-Aware Networks: Research Challenges and Open Issues", in *Proceedings of International Tyrrhenian Workshop on Digital Communications* (ITWDC), Ponza, Italy, September 2010.

[7] M. Grafl, C. Timmerer, H. Hellwagner, D. Négru, E. Borcoci, D. Renzi, A.-L. Mevel, and A. Chernilov, "Scalable Video Coding in Content-Aware Networks: Research Challenges and Open Issues", in *Trustworthy Internet*, L. Salgarelli, G. Bianchi, and N. Blefari-Melazzi, Eds. Milano: Springer Milan, pp. 349–358, 2011.

[8] M. Grafl and C. Timmerer (eds.), "Service/Content Adaptation Definition and Specification", ICT-ALICANTE, Deliverable D2.2, September 2011.

[9] M. Waltl, M. Grafl, C. Timmerer (eds.) et al., "The ALICANTE Adaptation Framework", ICT-ALICANTE, Deliverable D7F, June 2013.

[10] M. Grafl, C. Timmerer, H. Hellwagner, G. Xilouris, G. Gardikis, D. Renzi, S. Battista, E. Borcoci, and D. Négru, "Scalable Media Coding Enabling Content-Aware Networking", *IEEE MultiMedia*, vol. 20, no. 2, pp. 30–41, June 2013.

[11] G. Gardikis, E. Pallis, and M. Grafl, "Media-Aware Networks in Future Internet Media", accepted for publication in *3D Future Internet Media*, A. Kondoz and T. Dagiuklas, Eds., Springer Science+Business Media, LLC, New York, scheduled for publication in 2013.

[12] SVC Demux & Mux, "SVC Demux & Mux | Free software downloads at SourceForge.net", Website, URL: "https://sourceforge.net/projects/svc-demux-mux/". Accessed May 9, 2013.

[13] SVC RTP MST, "SVC RTP MST | Free software downloads at SourceForge.net", Website, URL: "https://sourceforge.net/projects/svc-rtp-mst/". Accessed May 9, 2013.

[14] M. Grafl, C. Timmerer, "Representation Switch Smoothing for Adaptive HTTP Streaming", accepted for publication in *Proceedings of the 4th International Workshop on Perceptual Quality of Systems* (PQS 2013), Vienna, Austria, September 2013.

[15] ISO/IEC 15938-5:2003/Amd 4:2012, "Social metadata", 2012.

[16] ISO/IEC 23006-4:2013, "Information technology – Multimedia service platform technologies – Part 4: Elementary services", 2013.

[17] ISO/IEC 23006-5:2013, "Information technology – Multimedia service platform technologies – Part 5: Service aggregation", 2013.

[18] C. Timmerer, M. Eberhard, M. Grafl, K. Mitchell, S. Dutton, and H. Hellwagner, "A Metadata Model for Peer-to-Peer Media Distribution", in *Proceedings of the Workshop on Interoperable Social Multimedia Applications* (WISMA 2010), Barcelona, Spain, May 2010.

[19] M. Grafl, "Overview of MPEG-M Part 4 (Elementary Services)", MPEG output document ISO/IEC JTC1/SC29/WG11 N11969, Geneva, Switzerland, March 2011. Available online: "http://mpeg.chiariglione.org/standards/mpeg-m/elementary-services", accessed May 18, 2013.

[20] P. Kudumakis, M. Sandler, A.-C. Anadiotis, I. Venieris, A. Difino, X. Wang, G. Tropea, M. Grafl, V. Rodríguez-Doncel, S. Llorente, J. Delgado, "White Paper on MPEG-M: A Digital Media Ecosystem for Interoperable Applications", MPEG output document ISO/IEC JTC1/SC29/WG11 N13952, Incheon, Republic of Korea, April 2013. Available online: "http://mpeg.chiariglione.org/sites/default/files/files/standards/docs/w13952.zip", accessed May 18, 2013.

[21] P. Kudumakis, M. Sandler, A.-C. Anadiotis, I. Venieris, A. Difino, X. Wang, G. Tropea, M. Grafl, V. Rodríguez-Doncel, S. Llorente, J. Delgado, "MPEG-M: A Digital Media Ecosystem for Interoperable Applications", accepted for publication in *Signal Processing: Image Communication*, scheduled for publication in 2013.

[22] W. B. Pennebaker and J. L. Mitchell, "JPEG: Still Image Data Compression Standard", Springer, 1992.

[23] ISO/IEC 14496-10:2012, "Coding of audio-visual objects – Part 10: Advanced Video Coding", 7th edition, 2012.

[24] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.

[25] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with H.264/AVC: tools, performance, and complexity", *IEEE Circuits and Systems Magazine*, vol. 4, no. 1, pp. 7–28, 2004.

[26] ISO/IEC 14496-14:2003, "Information technology – Coding of audio-visual objects – Part 14: MP4 file format", 2003.

[27] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H. 264/AVC Standard." *IEEE Transactions on Circuits and Systems for Video Technology* 17, no. 9, pp. 1103–1120, September 2007.

[28] C. A. Segall and G. J. Sullivan, "Spatial Scalability Within the H.264/AVC Scalable Video Coding Extension", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1121–1135, September 2007.

[29] H. W. Jones, "A Comparison of Theoretical and Experimental Video Compression Designs", *IEEE Transactions on Electromagnetic Compatibility*, vol. EMC-21, no. 1, pp. 50–56, February 1979.

[30] C. Reader, "History of MPEG Video Compression – Ver. 4.0", Joint Video Team (JVT), Doc. JVT-E066, Geneva, Switzerland, October 2002. Available online "http://wftp3.itu.int/av-arch/jvt-site/2002_10_Geneva/JVT-E066.zip", accessed April 14, 2013.

[31] ISO/IEC 13818-2:1996, "Generic Coding of Moving Pictures and Associated Audio – Part 2: Video", 1st edition, 1996.

[32] ISO/IEC 14496-10:2008, "Coding of audio-visual objects – Part 10: Advanced Video Coding", 4th edition, 2008.

[33] M. Wien, H. Schwarz, and T. Oelbaum, "Performance Analysis of SVC", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1194–1203, September 2007.

[34] K. Brandenburg and B. Gill, "First Ideas on Scalable Audio Coding", in *Proceedings of the 97th Audio Engineering Society Convention*, San Francisco, CA, USA, November 1994.

[35] B. Kovesi, D. Massaloux, and A. Sollaud, "A scalable speech and audio coding scheme with continuous bitrate flexibility", in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP '04), Montreal, Canada, vol. 1, pp. I–273–6, May 2004.

[36] M. van der Schaar and P. A. Chou, "Multimedia over IP and Wireless Networks: Compression, Networking, and Systems", Academic Press, 2011.

[37] N. Adami, A. Signoroni, and R. Leonardi, "State-of-the-Art and Trends in Scalable Video Compression With Wavelet-Based Approaches", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1238–1255, September 2007.

[38] V. K. Goyal, "Multiple description coding: compression meets the network", *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 74–93, September 2001.

[39] H. Hellwagner, R. Kuschnig, T. Stütz, and A. Uhl, "Efficient in-network adaptation of encrypted H.264/SVC content", *Signal Processing: Image Communication*, vol. 24, no. 9, pp. 740–758, October 2009.

[40] B. Zhang, M. Wien, and J.-R. Ohm, "A novel framework for robust video streaming based on H.264/AVC MGS coding and unequal error protection", in *Proceedings of the International Symposium on Intelligent Signal Processing and Communication Systems* (ISPACS 2009), Kanazawa, Japan, pp. 107–110, December 2009.

[41] ALICANTE, "ALICANTE - MediA Ecosystem Deployment through Ubiquitous Content-Aware Network Environments - FP7 Project" Home Page, URL: "http://www.ict-alicante.eu/". Accessed September 25, 2012.

[42] G. Tselentis, A. Galis, S. Krco, and V. Lotz, "Towards the Future Internet: Emerging Trends from European Research", IOS Press, 2010.

[43] E. Borcoci, D. Négru, and C. Timmerer, "A Novel Architecture for Multimedia Distribution Based on Content-Aware Networking", in *Proceedings of the 3rd International Conference on Communication Theory, Reliability, and Quality of Service* (CTRQ), Athens, Greece, pp. 162–168, June 2010.

[44] S. Wenger, Ye-Kui Wang, and T. Schierl, "Transport and Signaling of SVC in IP Networks", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1164–1173, September 2007.

[45] R. Kuschnig, I. Kofler, M. Ransburg, and H. Hellwagner, "Design options and comparison of in-network H. 264/SVC adaptation", *Journal of Visual Communication and Image Representation*, vol. 19, no. 8, pp. 529–542, December 2008.

[46] I. Kofler, M. Prangl, R. Kuschnig, and H. Hellwagner, "An H. 264/SVC-based adaptation proxy on a WiFi router", in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Braunschweig, Germany, pp. 63–68, May 2008.

[47] B. Shen, W.-T. Tan, and F. Huve, "Dynamic Video Transcoding in Mobile Environments", *IEEE MultiMedia*, vol. 15, no. 1, pp. 42–51, March 2008.

[48] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast", in *Conference Proceedings on Applications, technologies, architectures, and protocols for computer communications*, Palo Alto, California, United States, pp. 117–130, October 1996.

[49] IETF RFC 3550, "RTP: A Transport Protocol for Real-Time Applications", IETF Request for Comments, July 2003.

[50] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet", *IEEE Multimedia*, vol. 18, no. 4, pp. 62–67, October-December 2011.

[51] Social Sensor, "Social Sensor – Sensing User Generated Input for Improved Media Discovery and Experience", Home Page, URL: "http://www.socialsensor.eu/". Accessed May 19, 2013.

[52] C Müller, S. Lederer, B. Rainer, M. Waltl, M. Grafl, and C. Timmerer, "Open Source Column: Dynamic Adaptive Streaming over HTTP Toolset", *ACM SIGMultimedia Records*, vol. 5, no. 1, March 2013.

[53] P2P-Next, "P2P-Next – Shaping the Next Generation of Internet TV", Home Page, URL: "http://www.p2p-next.org/". Accessed May 19, 2013.

[54] ALICANTE Blog, "Standardisation Activities: Contributions to MPEG-M", blog entry, URL: "http://www.ict-alicante.eu/blog/?p=1401", January 24, 2013. Accessed May 20, 2013.

[55] P. Kudumakis, X. Wang, S. Matone, and M. Sandler, "MPEG-M: Multimedia Service Platform Technologies [Standards in a Nutshell]", *IEEE Signal Processing Magazine*, vol. 28, no. 6, pp. 159–163, November 2011.

[56] ALICANTE Blog, "New Amendment of MPEG-7 Part 5", blog entry, URL: "http://www.ict-alicante.eu/blog/?p=1374", October 1, 2012. Accessed May 19, 2013.

[57] MPEG output document N13158, "Report of Results of the Joint Call for Proposals on Scalable High Efficiency Video Coding (SHVC)", *ISO/IEC JTC 1/SC 29/WG 11/N13158*, Shanghai, China, October 2012. Available online "http://mpeg.chiariglione.org/standards/mpeg-h/high-efficiency-video-coding/report-results-joint-call-proposals-scalable-high", accessed April 15, 2013.

[58] Joint Video Team (JVT), "Joint Scalable Video Model (JSVM)", Version 9.19.15, 2011.

[59] Y. Sánchez, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. De Vleeschauwer, W. Van Leekwijck, and Y. Le Louédec, "iDASH: improved dynamic adaptive streaming over HTTP using scalable video coding", in *Proceedings of the 2nd Annual ACM Conference on Multimedia Systems*, New York, NY, USA, pp. 257–264, February 2011.

[60] Y. Sánchez, C. Hellge, T. Schierl, W. Van Leekwijck, and Y. Le Louédec, "Scalable Video Coding based DASH for efficient usage of network resources", *Position Paper at the 3rd W3C Web and TV Workshop*, September 2011.

[61] C. Müller, D. Renzi, S. Lederer, S. Battista, and C. Timmerer, "Using Scalable Video Coding for Dynamic Adaptive Streaming over HTTP in Mobile Environments", in *Proceedings of the 20th European Signal Processing Conference* (EUSIPCO), Bucharest, Romania, August 2012.

[62] I. Kofler, R. Kuschnig, and H. Hellwagner, "Implications of the ISO Base Media File Format on Adaptive HTTP Streaming of H. 264/SVC", in *Proceedings of the 9th IEEE Consumer Communications and Networking Conference*, Las Vegas, NV, USA, January 2012.

[63] M. Wien and H. Schwarz, "Testing conditions for SVC coding efficiency and JSVM performance evaluation", Joint Video Team (JVT), Doc. JVT-Q205, Poznan, Poland, October 2005. Available online "http://wftp3.itu.int/av-arch/jvt-site/2005_10_Nice/JVT-Q205.doc", accessed April 14, 2013.

[64] H. Schwarz and T. Wiegand, "R-D Optimized Multi-Layer Encoder Control for SVC", in *Proceedings of the IEEE International Conference on Image Processing*, vol. 2, p. II –281 –II –284, October 2007.

[65] T. Oelbaum, H. Schwarz, M. Wien, and T. Wiegand, "Subjective performance evaluation of the SVC extension of H.264/AVC", in *Proceedings of the 15th IEEE International Conference on Image Processing*, pp. 2772–2775, October 2008.

[66] J.-S. Lee, F. De Simone, and T. Ebrahimi, "Subjective quality assessment of scalable video coding: A survey", in *Proceedings of 3rd International Workshop on Quality of Multimedia Experience* (QoMEX), Mechelen, Belgium, pp. 25–30, September 2011.

[67] F. Niedermeier, M. Niedermeier, and H. Kosch, "Quality Assessment of the MPEG-4 Scalable Video CODEC", in *Image Analysis and Processing – ICIAP* 2009, vol. 5716, P. Foggia, C. Sansone, and M. Vento, Eds. Springer Berlin / Heidelberg, pp. 297–306, 2009.

[68] F. Niedermeier, "Objective assessment of the MPEG-4 scalable video CODEC", Diploma Thesis, University of Passau, Department of Informatics and Mathematics, Passau, Germany, 2009.

[69] M. Niedermeier, "Subjective assessment of the MPEG-4 scalable video CODEC", Diploma Thesis, University of Passau, Department of Informatics and Mathematics, Passau, Germany, 2009.

[70] G. Nur, H. K. Arachchi, S. Dogan, and A. M. Kondoz, "Advanced Adaptation Techniques for Improved Video Perception", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 2, pp. 225–240, February 2012.

[71] G. Nur, H. K. Arachchi, S. Dogan, and A. M. Kondoz, "Seamless video access for mobile devices by content-aware utility-based adaptation", *Multimedia Tools and Applications*, pp. 1–31, May 2012.

[72] M. Slanina, M. Ries, and J. Vehkaperä, "Rate Distortion Performance of H.264/SVC in Full HD with Constant Frame Rate and High Granularity", in *Proceedings of the 8th International Conference on Digital Telecommunications* (ICDT 2013), Venice, Italy, pp. 7–13, April 2013.

[73] R. Gupta, A. Pulipaka, P. Seeling, L. J. Karam, and M. Reisslein, "H.264 Coarse Grain Scalable (CGS) and Medium Grain Scalable (MGS) Encoded Video: A

Trace Based Traffic and Quality Evaluation", *IEEE Transactions on Broadcasting*, vol. 58, no. 3, pp. 428–439, September 2012.

[74] S.-H. Yang and W.-L. Tang, "What are Good CGS/MGS Configurations for H.264 Quality Scalable Coding?", in *Proceedings of the International Conference on Signal Processing and Multimedia Applications* (SIGMAP 2011), Seville, Spain, pp. 104–109, July 2011.

[75] T. Wiegand, L. Noblet, and F. Rovati, "Scalable Video Coding for IPTV Services", *IEEE Transactions on Broadcasting*, vol. 55, no. 2, pp. 527–538, June 2009.

[76] Z. Avramova, D. De Vleeschauwer, K. Spaey, S. Wittevrongel, H. Bruneel, and C. Blondia, "Comparison of simulcast and scalable video coding in terms of the required capacity in an IPTV network", in *Proceedings of Packet Video* (PV) Workshop, pp. 113–122, November 2007.

[77] Z. Avramova, D. De Vleeschauwer, P. Debevere, S. Wittevrongel, P. Lambert, R. Van de Walle, and H. Bruneel, "On the performance of scalable video coding for VBR TV channels transport in multiple resolutions and qualities", *Multimedia Tools and Applications*, vol. 57, no. 3, pp. 605–631, April 2012.

[78] P. Lambert, P. Debevere, S. Moens, R. Van de Walle, and J.-F. Macq, "Optimizing IPTV video delivery using SVC spatial scalability", in *Proceedings of the 10th Workshop on Image Analysis for Multimedia Interactive Services* (WIAMIS '09), London, UK, pp. 89–92, May 2009.

[79] Google+ Hangout, "Catch up in a hangout - Google+", Home Page, URL: "http://www.google.com/+/learnmore/hangouts/". Accessed June 29, 2012.

[80] Y. Xu, C. Yu, J. Li, H. Hu, Y. Liu, and Y. Wang, "Measurement Study of Commercial Video Conferencing Systems", Technical Report, Polytechnic Institute of NYU, New York, NY, USA, 2012. Available online "http://eeweb.poly.edu/faculty/yongliu/docs/MPVC-measurement.pdf", accessed April 14, 2013.

[81] Y. Xu, C. Yu, J. Li, and Y. Liu, "Video Telephony for End-consumers: Measurement Study of Google+, iChat, and Skype", in *Proceedings of the ACM Internet measurement conference* (IMC 2012), Boston, Massachusetts, USA, November 2012.

[82] T. Wiegand and G. J. Sullivan, "The picturephone is here. Really", *IEEE Spectrum*, vol. 48, no. 9, pp. 50–54, September 2011.

[83] Apple HTTP Live Streaming, "HTTP Live Streaming Resources - Apple Developer" Home Page, URL: "https://developer.apple.com/resources/http-streaming/". Accessed June 29, 2012.

[84] Adobe Flash Media Streaming, "Streaming server | Adobe Flash Media Streaming Server 4.5", Home Page, URL: "http://www.adobe.com/products/flash-media-streaming.html". Accessed June 29, 2012.

[85] Adobe HTTP Dynamic Streaming, "Live video streaming online | HTTP Dynamic Streaming", Home Page, URL: "http://www.adobe.com/products/hds-dynamic-streaming.html". Accessed June 29, 2012.

[86] Microsoft Smooth Streaming, "Smooth Streaming: The Official Microsoft IIS Site", Home Page, URL: "http://www.iis.net/download/SmoothStreaming". Accessed June 29, 2012.

[87] YouTube, "YouTube - Broadcast Yourself", Home Page, URL: "http://www.youtube.com/". Accessed June 29, 2012.

[88] Netflix, "Netflix - Watch TV Shows Online, Watch Movies Online", Home Page, URL: "http://www.netflix.com/". Accessed June 29, 2012.

[89] Hulu, "Watch TV. Watch Movies. | Online | Free | Hulu", Home Page, URL: "http://www.hulu.com/". Accessed June 29, 2012.

[90] MTV, "New Music Videos, Reality TV Shows, Celebrity News, Pop Culture | MTV", Website, URL: "http://mtv.com/". Accessed February 2, 2013.

[91] Facebook Video Calling, "Facebook Video Calling", Home Page, URL: "http://www.facebook.com/videocalling/". Accessed June 29, 2012.

[92] Skype, "Free Skype internet calls and cheap calls to phones online - Skype", Home Page, URL: "http://www.skype.com/". Accessed June 29, 2012.

[93] iOS Developer Library, "Technical Note TN2224", Website, URL: "http://developer.apple.com/library/ios/#technotes/tn2224/_index.html", August 3, 2011. Accessed June 29, 2012.

[94] OS X Developer Library, "Technical Note TN2218", Website, URL: "http://developer.apple.com/library/mac/#technotes/tn2218/_index.html", May 1, 2008. Accessed June 29, 2012.

[95] M. Levkov, "Video encoding and transcoding recommendations for HTTP Dynamic Streaming on the Adobe® Flash® Platform", Adobe Systems Inc., URL: "http://download.macromedia.com/flashmediaserver/http_encoding_recommendations.pdf", October 2010. Accessed August 29, 2012.

[96] A. Kapoor, "Dynamic streaming on demand with Flash Media Server 3.5 | Adobe Developer Connection", blog entry, URL: "http://www.adobe.com/devnet/flashmediaserver/articles/dynstream_on_demand.html", January 12, 2009. Accessed June 29, 2012.

[97] J. Ozer, "Adaptive Streaming in the Field", *Streaming Media Magazine*, vol. December 2010/January 2011, January 2011.

[98] J. Ozer, "Encoding for Adaptive Streaming", presented at the *Streaming Media West 2011*, Los Angeles, CA, USA, November 2011.

[99] YouTube Help, "Advanced encoding specifications - YouTube Help", Website, URL:

"http://support.google.com/youtube/bin/static.py?hl=en&topic=1728573&guide=1728585&page=guide.cs". Accessed October 1, 2012.

[100] YouTube Help, "Encoding settings for live streaming - YouTube Help", Website, URL: "http://support.google.com/youtube/bin/static.py?hl=en&guide=2474025&topic=2474327&page=guide.cs&answer=1723080". Accessed October 1, 2012.

[101] Apple QuickTime, "Apple - QuickTime - Download and watch videos, movies, and TV shows", Home Page, URL: "http://www.apple.com/quicktime/". Accessed June 29, 2012.

[102] Adobe System Inc., "RTMP Specification 1.0", URL: "http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/rtmp/pdf/rtmp_specification_1.0.pdf", June 2009. Accessed August 29, 2012.

[103] bSoft, "bSoft >> Home", Home Page, URL: "http://bsoft.net/". Accessed September 26, 2012.

[104] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2011-2016 [Visual Networking Index (VNI)]", Website, URL: "http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html", May 30, 2012. Accessed October 2, 2012.

[105] OECD Broadband Portal, "Average advertised download speeds, by technology (Oct. 2009)", Excel file, URL: "http://web.archive.org/web/20110305191656/http://www.oecd.org/dataoecd/10/54/39575095.xls", October 2009. Accessed October 2, 2012.

[106] OECD Broadband Portal, "Average advertised download speeds, by technology (Sept. 2011)", Excel file, URL: "http://www.oecd.org/internet/broadbandandtelecom/BB-Portal_5b_13July_Final.xls", September 2011. Accessed October 2, 2012.

[107] OECD Broadband Portal, "Broadband and telecom - Organisation for Economic Co-operation and Development", Website, URL: "http://www.oecd.org/internet/broadbandandtelecom/oecdbroadbandportal.htm". Accessed October 2, 2012.

[108] MainConcept, "Home: MainConcept", Home Page, URL: "http://mainconcept.com/". Accessed September 25, 2012.

[109] Vanguard Software Solutions, "Vanguard Software Solutions | Scalable Video Coding (SVC)", Website, URL: "http://www.vsofts.com/technology/scalable-video-coding.html". Accessed January 4, 2013.

[110] H. Kirchhoffer, H. Schwarz, and T. Wiegand, "CE1: Simplified FGS", Joint Video Team (JVT), Doc. JVT-W090, San Jose, CA, April 2007. Available online "http://wftp3.itu.int/av-arch/jvt-site/2007_04_SanJose/JVT-W090.zip", accessed April 14, 2013.

[111] J.-R. Ohm, "Bildsignalverarbeitung fuer multimedia-systeme", Skript, Institut für Nachrichtentechnik und theoretische Elektrotechnik der TU Berlin, 1999.

[112] J. Klaue, B. Rathke, and A. Wolisz, "EvalVid – A Framework for Video Transmission and Quality Evaluation", in *Computer Performance Evaluation. Modelling Techniques and Tools*, vol. 2794, P. Kemper and W. Sanders, Eds. Springer Berlin / Heidelberg, pp. 255–272, 2003.

[113] D. Rodrigues, E. Cerqueira, and E. Monteiro, "Quality of Service and Quality of Experience in Video Streaming", in *Proceedings of the International Workshop on Traffic Management and Traffic Engineering for the Future Internet* (FITraMEn2008), EuroNF NoE, Porto, Portugal, pp. 11–12, December 2008.

[114] K. Piamrat, C. Viho, J.-M. Bonnin, and A. Ksentini, "Quality of Experience Measurements for Video Streaming over Wireless Networks", in *Proceedings of the 6th International Conference on Information Technology: New Generations* (ITNG '09), Las Vegas, Nevada, USA, pp. 1184–1189, April 2009.

[115] T. A. Le, H. Nguyen, and H. Zhang, "EvalSVC – An evaluation platform for scalable video coding transmission", in *Proceedings of the 14th IEEE International Symposium on Consumer Electronics* (ISCE), Braunschweig, Germany, pp. 1–6, June 2010.

[116] T. Zinner, O. Abboud, O. Hohlfeld, T. Hoßfeld, and P. Tran-Gia, "Towards QoE Management for Scalable Video Streaming", in *Proceedings of the 21th ITC Specialist Seminar on Multimedia Applications-Traffic, Performance and QoE*, Miyazaki, Japan, March 2010.

[117] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity", *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

[118] NTIA, "ITS | Video Quality Metric (VQM) Software", Website, URL: "http://www.its.bldrdoc.gov/resources/video-quality-research/software.aspx". Accessed October 3, 2012.

[119] ITU-R Rec. BT.1683, "Objective perceptual video quality measurement techniques for standard definition digital broadcast television in the presence of a full reference", 2004.

[120] M. H. Pinson and S. Wolf, "A new standardized method for objectively measuring video quality", *IEEE Transactions on Broadcasting*, vol. 50, no. 3, pp. 312–322, September 2004.

[121] S. Wolf and M. Pinson, "Application of the NTIA general video quality metric (VQM) to HDTV quality monitoring", in *Proceedings of The 3rd International Workshop on Video Processing and Quality Metrics for Consumer Electronics* (VPQM), Scottsdale, AZ, USA, January 2007.

[122] K. Seshadrinathan, R. Soundararajan, A. C. Bovik, and L. K. Cormack, "Study of Subjective and Objective Quality Assessment of Video", *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1427–1441, June 2010.

[123] Y. Wang, "Survey of objective video quality measurements", Technical Report, Worcester Polytechnic Institute, June 2006. Available online "ftp://130.215.28.31/pub/techreports/pdf/06-02.pdf", accessed April 14, 2013.

[124] S. Chikkerur, V. Sundaram, M. Reisslein, and L. J. Karam, "Objective Video Quality Assessment Methods: A Classification, Review, and Performance Comparison", *IEEE Transactions on Broadcasting*, vol. 57, no. 2, pp. 165–182, 2011.

[125] ITU-T Rec. P.910, "Subjective video quality assessment methods for multimedia applications", 2008.

[126] Xiph.Org Foundation, "Xiph.org :: Derf's Test Media Collection", Website, URL: "http://media.xiph.org/video/derf/". Accessed January 3, 2013.

[127] P. Ni, A. Eichhorn, C. Griwodz, and P. Halvorsen, "Frequent layer switching for perceived quality improvements of coarse-grained scalable video", *Multimedia Systems*, vol. 16, no. 3, pp. 171–182, June 2010.

[128] A. Eichhorn and P. Ni, "Pick Your Layers Wisely - A Quality Assessment of H.264 Scalable Video Coding for Mobile Devices", in *Proceedings of the IEEE International Conference on Communications* (ICC '09), Mechelen, Belgium, pp. 1–6, September 2009.

[129] x264, "VideoLAN - x264, the best H.264/AVC encoder", Website, URL: "http://www.videolan.org/developers/x264.html". Accessed January 4, 2013.

[130] H. Riiser, P. Halvorsen, C. Griwodz, and D. Johansen, "Low overhead container format for adaptive streaming", in *Proceedings of the 1st Annual ACM SIGMM Conference on Multimedia Systems*, Scottsdale, Arizona, USA, pp. 193–198, February 2010.

[131] Akamai Technologies Inc., "Akamai HD Network: Encoding Best Practices for the iPhone and iPad", White Paper, URL: "http://www.akamai.com/dl/whitepapers/Akamai_HDNetwork_Encoding_BP_iPhone_iPad.pdf", 2011. Accessed January 31, 2013.

[132] G. Van der Auwera, P. David, and M. Reisslein, "Traffic and Quality Characterization of Single-Layer Video Streams Encoded with the H.264/MPEG-4 Advanced Video Coding Standard and Scalable Video Coding Extension", *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 698–718, 2008.

[133] B. Crabtree, M. Nilsson, P. Mulroy, and S. Appleby, "Equitable Quality Video Streaming", in *Proceedings of the 6th IEEE Consumer Communications and Networking Conference* (CCNC 2009), Las Vegas, Nevada, USA, pp. 1–5, January 2009.

[134] MPEG output document N13514, "Study of ISO/IEC PDTR 23009-3 DASH Implementation Guidelines", *ISO/IEC JTC 1/SC 29/WG 11/N13514*, Incheon, Republic of Korea, April 2013.

[135] ISO/IEC 11172-2:1993, "Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 2: Video", 1993.

[136] ISO/IEC 15444-3:2007, "Information technology – JPEG 2000 image coding system: Motion JPEG 2000", 2007.

[137] ISO/IEC 14496-2:2004, "Information technology – Coding of audio-visual objects – Part 2: Visual", 2004.

[138] SMPTE ST 2042-1:2012, "VC-2 Video Compression", 2012.

[139] RFC 6386, "VP8 Data Format and Decoding Guide", IETF Request for Comments, 2011.

[140] ISO/IEC FDIS 23008-2, "Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 2: High efficiency video coding", 2013.

[141] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 688–703, July 2003.

[142] I. Ahmad, X. Wei, Y. Sun, and Y.-Q. Zhang, "Video transcoding: an overview of various techniques and research issues", *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 793–804, October 2005.

[143] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: an overview", *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 18–29, March 2003.

[144] G. Fernandez-Escribano, H. Kalva, P. Cuenca, L. Orozco-Barbosa, and A. Garrido, "A Fast MB Mode Decision Algorithm for MPEG-2 to H.264 P-Frame Transcoding", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 2, pp. 172–185, February 2008.

[145] J. Xin, A. Vetro, and H. Sun, "Converting DCT coefficients to H.264/AVC transform coefficients", *Advances in Multimedia Information Processing-PCM*, no. 2004, pp. 939–946, 2004.

[146] J. Xin, A. Vetro, H. Sun, and Y. Su, "Efficient MPEG-2 to H.264/AVC Transcoding of Intra-Coded Video", *EURASIP Journal on Applied Signal Processing*, vol. 2007, pp. 1–13, 2007.

[147] G. Fernández-Escribano, P. Cuenca, L. Orozco-Barbosa, A. Garrido, and H. Kalva, "Simple intra prediction algorithms for heterogeneous MPEG-2/H.264 video transcoders", *Multimedia Tools and Applications*, vol. 38, no. 1, pp. 1–25, May 2008.

[148] J. De Cock, S. Notebaert, and R. Van de Walle, "Transcoding from H.264/AVC to SVC with CGS Layers", in *Proceedings of the IEEE International Conference*

on Image Processing (ICIP 2007), San Antonio, TX, USA, vol. 4, pp. IV – 73–IV – 76, September 2007.

[149] J. De Cock, S. Notebaert, P. Lambert, and R. Van de Walle, "Advanced bitstream rewriting from H. 264/AVC to SVC", in *Proceedings of the 15th IEEE International Conference on Image Processing* (ICIP 2008), San Diego, California, USA, pp. 2472–2475, October 2008.

[150] J. De Cock, S. Notebaert, P. Lambert, and R. Van de Walle, "Architectures for Fast Transcoding of H.264/AVC to Quality-Scalable SVC Streams", *IEEE Transactions on Multimedia*, vol. 11, no. 7, pp. 1209–1224, November 2009.

[151] J. D. Cock, S. Notebaert, K. Vermeirsch, P. Lambert, and R. V. de Walle, "Transcoding of H.264/AVC to SVC with motion data refinement", in *Proceedings of the 16th IEEE International Conference on Image Processing* (ICIP), Cairo, Egypt, pp. 3673–3676, November 2009.

[152] J. De Cock, "Compressed-domain transcoding of H.264/AVC and SVC video streams", PhD Thesis, Ghent University, Faculty of Engineering, Ghent, Belgium, 2009.

[153] D. Kim and J. Jeong, "Fast Transcoding Algorithm from MPEG2 to H.264", *Advances in Image and Video Technology*, vol. 4319, pp. 1067–1074, 2006.

[154] B. Petjanski and H. Kalva, "DCT domain intra MB mode decision for MPEG-2 to H.264 transcoding", in *Digest of Technical Papers of the International Conference on Consumer Electronics* (ICCE '06), Las Vegas, Nevada, USA, pp. 419–420, January 2006.

[155] G. Chen, Y. Zhang, S. Lin, and F. Dai, "Efficient block size selection for MPEG-2 to H.264 transcoding", in *Proceedings of the 12th annual ACM International Conference on Multimedia* (MULTIMEDIA '04), New York, NY, USA, p. 300, October 2004.

[156] J. Yang, Q. Dai, W. Xu, and R. Ding, "A rate control algorithm for MPEG-2 to H. 264 real-time transcoding", in *Proc. SPIE 5960, Visual Communications and Image Processing 2005, 59605U*, vol. 5960, pp. 1995–2003, July 2005.

[157] P. Amon, Haoyu Li, A. Hutter, D. Renzi, and S. Battista, "Scalable video coding and transcoding", in *Proceedings of the IEEE International Conference on Automation, Quality and Testing, Robotics* (AQTR 2008), vol. 1, pp. 336–341, May 2008.

[158] A. Segall and Jie Zhao, "Bit stream rewriting for SVC-to-AVC conversion", in Proceedings of the 15th *IEEE International Conference on Image Processing*, San Diego, CA, USA, pp. 2776–2779, October 2008.

[159] M. Sablatschan, M. Ransburg, and H. Hellwagner, "Towards an Improved SVC-to-AVC Rewriter", in *Proceedings of the 2nd International Conferences on Advances in Multimedia*, pp. 18–21, June 2010.

[160] H. Liu, Y.-K. Wang, Y. Chen, and H. Li, "Spatial transcoding from Scalable Video Coding to H.264/AVC", in *Proceedings of the IEEE International*

Conference on Multimedia and Expo (ICME 2009), Cancun, Mexico, pp. 29–32, July 2009.

[161] B. Li, Y. Guo, H. Li, and C. W. Chen, "Hybrid Bit-stream Rewriting from Scalable Video Coding to H.264/AVC", *Proc. SPIE 7744, Visual Communications and Image Processing 2010, 77441A*, August 2010.

[162] M. Sablatschan, J. O. Murillo, M. Ransburg, and H. Hellwagner, "Efficient SVC-to-AVC Conversion at a Media Aware Network Element", in *Mobile Multimedia Communications*, J. Rodriguez, R. Tafazolli, and C. Verikoukis, Eds., Springer Berlin Heidelberg, pp. 582–588, 2012.

[163] P. Kunzelmann and H. Kalva, "Reduced Complexity H.264 to MPEG-2 Transcoder", in *Digest of Technical Papers of the International Conference on Consumer Electronics*, Las Vegas, Nevada, USA, pp. 1–2, January 2007.

[164] S. Moiron, S. Faria, A. Navarro, V. Silva, and P. Assunção, "Video transcoding from H.264/AVC to MPEG-2 with reduced computational complexity", *Signal Processing: Image Communication*, vol. 24, no. 8, pp. 637–650, September 2009.

[165] FFmpeg, "FFmpeg", Home Page, URL: "http://ffmpeg.org". Accessed September 25, 2012.

[166] GPL MPEG-1/2 DirectShow Decoder Filter, "SourceForge.net: GPL MPEG-1/2 DirectShow Decoder Filter - Project Web Hosting - Open Source Software", Home Page, URL: "http://gplmpgdec.sf.net/". Accessed September 25, 2012.

[167] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves", ITU-T Q.6/SG 16 Video Coding Experts Group (VCEG), Doc. VCEG-M33, Austin, Texas, USA, April 2001. Available online "http://wftp3.itu.int/av-arch/video-site/0104_Aus/VCEG-M33.doc", accessed April 14, 2013.

[168] G. Bjøntegaard, "Improvements of the BD-PSNR model", ITU-T Q.6/SG 16 Video Coding Experts Group (VCEG), Doc. VCEG-AI11, Berlin, Germany, July 2008. Available online "http://wftp3.itu.int/av-arch/video-site/0807_Ber/VCEG-AI11.zip", accessed April 14, 2013.

[169] T. Kallioja, "Television Goes Online", *Telecommunications Forum presentation*, Espoo, Finland, November 2006. Available online: "http://www.netlab.tkk.fi/opetus/s383001/2006/kalvot/TelecomForum-2006.11.07-Kallioja-OnlineTV.pdf", accessed June 13, 2013.

[170] P. Assunção and M. Ghanbari, "A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bit streams", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 8, pp. 953–967, December 1998.

[171] M. Lavrentiev and D. Malah, "Transrating of MPEG-2 coded video via requantization with optimal trellis-based DCT coefficients modification", in *Proceedings of the 12th European Signal Processing Conference* (EUSIPCO), Vienna, Austria, 2004.

[172] J. Pan, S. Paul, R. Jain, "A survey of the research on future internet architectures", *IEEE Communications Magazine*, vol.49, no.7, pp. 26–36, July 2011.

[173] D. Trossen, "Invigorating the future internet debate", *SIGCOMM Computer Communication Review*, vol. 39, no. 5, pp. 44–51, October 2009.

[174] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, R. Braynard, "Networking named content", in *Proceedings of ACM CoNEXT 2009*, Rome, Italy, December 2009.

[175] J. Choi, J. Han, E. Cho, T. Kwon, Y. Choi, "A survey on content-oriented networking for efficient content delivery", *IEEE Communications Magazine*, vol.49, no.3, pp. 121–127, March 2011.

[176] H. Koumaras, D. Négru, E. Borcoci, V. Koumaras, C. Troulos, Y. Lapid, E. Pallis, M. Sidibé, A. Pinto, G. Gardikis, G. Xilouris, C. Timmerer, "Media Ecosystems: A Novel Approach for Content-Awareness in Future Networks", *Future Internet: Achievements and Promising Technology*, Springer Verlag, pp. 369–380, May 2011.

[177] IETF RFC 2326, "Real Time Streaming Protocol (RTSP)", IETF Request for Comments, April 1998.

[178] N. Ramzan, E. Quacchio, T. Zgaljic, S. Asioli, L. Celetto, E. Izquierdo, F. Rovati, "Peer-to-peer streaming of scalable video in future Internet applications", *IEEE Communications Magazine*, vol.49, no.3, pp.128–135, March 2011.

[179] M. Eberhard, T. Szkaliczki, H. Hellwagner, L. Szobonya, C. Timmerer, "An Evaluation of Piece-Picking Algorithms for Layered Content in Bittorrent-based Peer-to-Peer Systems", in *Proceedings of the IEEE Multimedia and Expo* (ICME'11), Barcelona, Spain, July 2011.

[180] T. Stockhammer, "Dynamic adaptive streaming over HTTP – standards and design principles", in *Proceedings of the 2nd Annual ACM Conference on Multimedia Systems*, New York, NY, USA, pp. 133–144, February 2011.

[181] Z. C. Zhang and V. O. Li, "Router-assisted layered multicast", in *Proceedings of IEEE International Conference on Communications* (ICC), New York, NY, USA, vol. 4, pp. 2657–2661, May 2002.

[182] IETF RFC 3973, "Protocol Independent Multicast – Dense Mode (PIM-DM): Protocol Specification (Revised)", IETF Request for Comments, January 2005.

[183] A. Kayssi, H. Karaki, and W. Abu-Khraybeh, "RTP-based caching of streaming media", in *Proceedings of the 8th IEEE International Symposium on Computers and Communication* (ISCC), pp. 1067–1071 vol. 2, July 2003.

[184] S. Lederer, C. Müller, and C. Timmerer, "Towards peer-assisted dynamic adaptive streaming over HTTP", in *Proceedings of 19th International Packet Video Workshop* (PV), pp. 161 –166, May 2012.

[185] P. Le Callet, S. Möller, and A. Perkis (eds.), "Qualinet White Paper on Definitions of Quality of Experience", European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003), Lausanne, Switzerland, June 2012.

[186] W. Cherif, A. Ksentini, D. Négru, M. Sidibe, "A_PSQA: PESQ-like non-intrusive tool for QoE prediction in VoIP services", in *Proceedings of the IEEE International Conference on Communications* (ICC), Ottawa, Canada, June 2012.

[187] J. Huusko et al., "Cross-layer architecture for scalable video transmission in wireless network", *Signal Processing: Image Communication*, vol. 22, no. 3, pp. 317–330, March 2007.

[188] P. Helle, H. Lakshman, M. Siekmann, J. Stegemann, T. Hinz, H. Schwarz, D. Marpe, and T. Wiegand, "A Scalable Video Coding Extension of HEVC", in *Proceedings of the Data Compression Conference* (DCC), Snowbird, UT, USA, pp. 201–210, March 2013.

[189] R. Mohan, J. R. Smith, and C.-S. Li, "Adapting multimedia Internet content for universal access", *IEEE Transactions on Multimedia*, vol. 1, no. 1, pp. 104–114, March 1999.

[190] F. Pereira and I. Burnett, "Universal multimedia experiences for tomorrow", *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 63–73, March 2003.

[191] E. Borcoci, A. Asgari, N. Butler, T. Ahmed, A. Mehaoua, G. Kourmentzas, and S. Eccles, "Service management for end-to-end QoS multimedia content delivery in heterogeneous environment", in *Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop* (aict/sapir/elete 2005), Lisbon, Portugal, pp. 46–52, July 2005.

[192] S. Devillers, C. Timmerer, J. Heuer, and H. Hellwagner, "Bitstream syntax description-based adaptation in streaming and constrained environments", *IEEE Transactions on Multimedia*, vol. 7, no. 3, pp. 463 – 470, June 2005.

[193] J. Chakareski, "In-Network Packet Scheduling and Rate Allocation: A Content Delivery Perspective", *IEEE Transactions on Multimedia*, vol. 13, no. 5, pp. 1092–1102, October 2011.

[194] ENVISION, "ENVISION: Co-optimisation of overlay applications and underlying networks", Home Page, URL: "http://envision-project.org/". Accessed September 26, 2012.

[195] DANAE, "European R&D Projects: DANAE", Website, URL: "http://cordis.europa.eu/projects/rcn/71233_en.html". Accessed September 26, 2012.

[196] M. Ransburg, H. Hellwagner, R. Cazoulat, B. Pellan, C. Concolato, S. De Zutter, C. Poppe, R. Van de Walle, and A. Hutter, "Dynamic and distributed adaptation of scalable multimedia content in a context-aware environment", in

Proceedings of the European Symposium on Mobile Media Delivery (EuMob 2006), Alghero, Italy, September 2006.

[197] MEDIEVAL, "Home - MEDIEVAL", Home Page, URL: "http://www.ict-medieval.eu/". Accessed June 13, 2013.

[198] I. Ahmed, L. Badia, D. Munaretto, and M. Zorzi, "Analysis of PHY/Application Cross-layer Optimization for Scalable Video Transmission in Cellular Networks", in *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks* (WoWMoM), Madrid, Spain, June 2013.

[199] ENTHRONE, "Enthrone Web Site", Home Page, URL: "http://enthrone.org/". Accessed September 26, 2012.

[200] S. A. Chellouche, D. Negru, Y. Chen, and M. Sidibe, "Home-Box-assisted content delivery network for Internet Video-on-Demand services", in *Proceedings of the 17th IEEE Symposium on Computers and Communications* (ISCC), Cappadocia, Turkey, pp. 544–550, July 2012.

[201] S. Perez, "Verizon And Motorola Announce FiOS TV Media Server That Can Record Six Shows At Once", TechCrunch, URL: "http://techcrunch.com/2013/01/07/verizon-and-motorola-announce-fios-tv-media-server-that-can-record-six-shows-at-once/", January 7, 2013. Accessed April 21, 2013.

[202] IETF RFC 6190, "RTP Payload Format for Scalable Video Coding", IETF Request for Comments, May 2011.

[203] FFmpeg documentation, "Video Codecs", Website, URL: "http://www.ffmpeg.org/general.html#Video-Codecs". Accessed April 21, 2013.

[204] Participatory Culture Foundation – Develop, "ConversionMatrix – Develop", Website, URL: "https://develop.participatoryculture.org/index.php/ConversionMatrix", October 23, 2012. Accessed April 21, 2013.

[205] S. A. Chellouche, J. Arnaud, and D. Negru, "Flexible User Profile Management for Context-Aware Ubiquitous Environments", in *Proceedings of the 7th IEEE Consumer Communications and Networking Conference* (CCNC), Las Vegs, Nevada, USA, pp. 1–5, January 2010.

[206] C. Müller, "libdash supports now persistent connections and pipelining", blog entry, URL: "http://www-itec.uni-klu.ac.at/dash/?p=553", March 1, 2012. Accessed April 4, 2013.

[207] swift, "swift: the multiparty transport protocol", Website, URL: "http://www.libswift.org/". Accessed April 23, 2013.

[208] V. Grishchenko, F. Osmani, R. Jimenez, J. Pouwelse, and H. Sips, "On the Design of a Practical Information-Centric Transport", Technical Report PDS-2011-006, Parallel and Distributed Systems Report Series, Delft University of Technology, Delft, Netherlands, March 2011. Available online

"http://people.kth.se/~rauljc/tud11/grishchenko2011swift.pdf", accessed April 23, 2013.

[209] V. Grishchenko, A. Bakker, and R. Petrocco, "Peer-to-Peer Streaming Peer Protocol (PPSPP)", IETF Internet Draft draft-ietf-ppsp-peer-protocol-06, February 2013. Available online "http://tools.ietf.org/html/draft-ietf-ppsp-peer-protocol-06", accessed April 23, 2013.

[210] libswift, "triblerteam/libswift · GitHub", Website, URL: "https://github.com/triblerteam/libswift". Accessed April 23, 2013.

[211] B. Shao, D. Renzi, P. Amon, G. Xilouris, N. Zotos, S. Battista, A. Kourtis, and M. Mattavelli, "An adaptive system for real-time scalable video streaming with end-to-end QOS control", in *Proceedings of the 11th International Workshop on Image Analysis for Multimedia Interactive Services* (WIAMIS 2010), Desenzano del Garda, Italy, pp. 1–4, April 2010.

[212] T. C. Thang, J.-G. Kim, J. W. Kang, and J.-J. Yoo, "SVC adaptation: Standard tools and supporting methods", *Signal Processing: Image Communication*, vol. 24, no. 3, pp. 214–228, March 2009.

[213] I. Kofler, "In-Network Adaptation of Scalable Video Content", PhD Thesis, Alpen-Adria-Universität Klagenfurt, Fakultät für Technische Wissenschaften, Klagenfurt, Austria, November 2010.

[214] M. Li, Z. Chen, and Y.-P. Tan, "On Quality of Experience of Scalable Video Adaptation", *Journal of Visual Communication and Image Representation*, in press, March 2013.

[215] D. T. Nguyen, M. Hayashi, and J. Ostermann, "Adaptive error protection for Scalable Video Coding extension of H.264/AVC", in *Proceedings of the IEEE International Conference on Multimedia and Expo* (ICME), Hannover, Germany pp. 417–420, June 2008.

[216] J. M. Monteiro, C. T. Calafate, and M. S. Nunes, "Evaluation of the H.264 Scalable Video Coding in Error Prone IP Networks", *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 652–659, 2008.

[217] E.-D. Jang, J.-G. Kim, T. C. Thang, and J.-W. Kang, "Adaptation of Scalable Video Coding to packet loss and its performance analysis", in *Proceedings of the 12th International Conference on Advanced Communication Technology* (ICACT), Gangwon-Do, Korea, vol. 1, pp. 696–700, February 2010.

[218] Y. Chen, K. Xie, F. Zhang, P. Pandit, and J. Boyce, "Frame loss error concealment for SVC", *Journal of Zhejiang Univ. - Sci. A*, vol. 7, no. 5, pp. 677–683, May 2006.

[219] Y. Guo, Y. Chen, Y.-K. Wang, H. Li, M. M. Hannuksela, and M. Gabbouj, "Error Resilient Coding and Error Concealment in Scalable Video Coding", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 6, pp. 781–795, 2009.

[220] S. Mohamed, G. Rubino, F. Cervantes, and H. Afifi, "Real-Time Video Quality Assessment in Packet Networks: A Neural Network Model", in *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications* (PDPTA'01), Las Vegas, Nevada, USA, June 2001.

[221] G. Rubino and M. Varela, "A new approach for the prediction of end-to-end performance of multimedia streams", in *Proceedings of the 1st International Conference on the Quantitative Evaluation of Systems* (QEST 2004), Enschede, Netherlands, September 2004.

[222] E. Gelenbe, "Learning in the recurrent random neural network", *Neural Computation*, vol. 5, no. 1, pp. 154–164, 1993.

[223] K. D. Singh, A. Ksentini, and B. Marienval, "Quality of Experience Measurement Tool for SVC Video Coding", in *Proceedings of the IEEE International Conference on Communications* (ICC), Kyoto, Japan, pp. 1–5, June 2011.

[224] A. Ksentini and Y. Hadjadj-Aoul, "On Associating SVC and DVB-T2 for Mobile Television Broadcast", in *Proceedings of the IEEE Global Telecommunications Conference* (GLOBECOM 2011), Houston, Texas, USA, pp. 1–5, December 2011.

[225] S. Winkler, "Perceptual distortion metric for digital color video", in *Proc. SPIE 3644, Human Vision and Electronic Imaging IV*, pp. 175–184, May 1999.

[226] C. Yim and A. C. Bovik, "Evaluation of temporal variation of video quality in packet loss networks", *Signal Processing: Image Communication*, vol. 26, no. 1, pp. 24–38, January 2011.

[227] K. Seshadrinathan and A. C. Bovik, "Temporal hysteresis model of time varying subjective video quality", in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP '11), Prague, Czech Republic, pp. 1153–1156, May 2011.

[228] V. Joseph and G. De Veciana, "Jointly optimizing multi-user rate adaptation for video transport over wireless systems: Mean-fairness-variability tradeoffs", in *Proceedings of the IEEE INFOCOM*, Orlando, FL, USA, pp. 567–575, March 2012.

[229] J. P. Hansen, S. Hissam, D. Plakosh, and L. Wrage, "Adaptive Quality of Service in ad hoc wireless networks", in *Proceedings of the IEEE Wireless Communications and Networking Conference* (WCNC), Paris, France, pp. 1749–1754, April 2012.

[230] J. P. Hansen and S. Hissam, "Assessing QoS Trade-Offs for Real-Time Video", in *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks* (WoWMoM), Madrid, Spain, June 2013.

[231] A. Hutter, P. Amon, G. Panis, E. Delfosse, M. Ransburg, and H. Hellwagner, "Automatic adaptation of streaming multimedia content in a dynamic and distributed environment", in *Proceedings of the IEEE International Conference on Image Processing* (ICIP 2005), Genoa, Italy, vol. 3, pp. III–716–9, September 2005.

[232] M. Yarvis, P. Reiher, and G. J. Popek, "Conductor: a framework for distributed adaptation", in *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems*, Rio Rico, Arizona, USA, pp. 44–49, March 1999.

[233] O. Abboud, J. Rückert, D. Hausheer, and R. Steinmetz, "QoE-aware Quality Adaptation in Peer-to-Peer Video-on-Demand", Technical Report PS-TR-2012-01, Peer-to-Peer Systems Engineering, Technische Universität Darmstadt, 2012. Available online "http://www.ps.tu-darmstadt.de/fileadmin/publications/PS-TR-2012-01.pdf", accessed April 14, 2013.

[234] G. Nur, H. K. Arachchi, S. Dogan, and A. M. Kondoz, "Ambient Illumination as a Context for Video Bit Rate Adaptation Decision Taking", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 12, pp. 1887–1891, December 2010.

[235] M. Waltl, C. Timmerer, and H. Hellwagner, "Improving the Quality of Multimedia Experience through Sensory Effects", in *Proceedings of the 2nd International Workshop on Quality of Multimedia Experience* (QoMEX 2010), pp. 124 –129, June 2010.

[236] M. Waltl, B. Rainer, C. Timmerer, H. Hellwagner, "Enhancing the User Experience with the Sensory Effect Media Player and AmbientLib", in *Advances in Multimedia Modeling* (K. Schoeffmann, B. Merialdo, A. Hauptmann, C.-W. Ngo, Y. Andreopoulos, C. Breiteneder, eds.), Springer Verlag, Berlin, Heidelberg, pp. 624-626, January 2012.

[237] B. Rainer, M. Waltl, E. Cheng, M. Shujau, C. Timmerer, S. Davis, I. Burnett, C. Ritz, H. Hellwagner, "Investigating the Impact of Sensory Effects on the Quality of Experience and Emotional Response in Web Videos", in *Proceedings of the 4th International Workshop on Quality of Multimedia Experience* (QoMEX'12), Yarra Valley, Australia, pp. 278-283, July 2012.

[238] C. Müller, S. Lederer, and C. Timmerer, "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments", in *Proceedings of the 4th Workshop on Mobile Video*, Chapel Hill, North Carolina, USA, pp. 37–42, February 2012.

[239] T. Hoßfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen, "Initial delay vs. interruptions: Between the devil and the deep blue sea", in *Proceedings of the 4th International Workshop on Quality of Multimedia Experience* (QoMEX 2012), Yarra Valley, Australia, pp. 1–6, July 2012.

[240] M. Fiedler, T. Hoßfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service", *IEEE Network*, vol. 24, no. 2, pp. 36–41, 2010.

[241] R. K. P. Mok, E. W. W. Chan, and R. K. C. Chang, "Measuring the quality of experience of HTTP video streaming", in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management* (IM 2011), Dublin, Ireland, pp. 485–492, May 2011.

[242] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. Halvorsen, "Flicker effects in adaptive video streaming to handheld devices", in *Proceedings of the 19th ACM International Conference on Multimedia*, New York, NY, USA, pp. 463–472, December 2011.

[243] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. Halvorsen, "Spatial flicker effect in video scaling", in *Proceedings of the 3rd International Workshop on Quality of Multimedia Experience* (QoMEX), Mechelen, Belgium, pp. 55–60, September 2011.

[244] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang, "QDASH: a QoE-aware DASH system", in *Proceedings of the 3rd ACM Multimedia Systems Conference*, New York, NY, USA, pp. 11–22, February 2012.

[245] M. Zink, O. Künzel, J. Schmitt, and R. Steinmetz, "Subjective Impression of Variations in Layer Encoded Videos", in *Quality of Service — IWQoS 2003*, K. Jeffay, I. Stoica, and K. Wehrle, Eds. Springer Berlin Heidelberg, pp. 137–154, 2003.

[246] P. Ni, A. Eichhorn, C. Griwodz, and P. Halvorsen, "Fine-grained scalable streaming from coarse-grained videos", in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, New York, NY, USA, pp. 103–108, June 2009.

[247] C. Sieber, T. Hoßfeld, T. Zinner, P. Tran-Gia, C. Timmerer, "Implementation and User-centric Comparison of a Novel Adaptation Logic for DASH with SVC", in *Proceedings of the IFIP/IEEE International Workshop on Quality of Experience Centric Management* (QCMan), Ghent, Belgium, May 2013.

[248] ISO/IEC 21000-7, "Information technology – Multimedia framework (MPEG-21) – Part 7: Digital Item Adaptation".

[249] Ian S. Burnett, Fernando Pereira, Rik Van de Walle, and Rob Koenen (eds), "The MPEG-21 Book", John Wiley & Sons, Chichester, England, 2006.

[250] ISO/IEC 23009-1:2012, "Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats", 2012.

[251] I. Kofler, "MPEG-21-based Adaptation Decision Taking in the Binary Encoded Metadata Domain", Diploma Thesis, Alpen-Adria-Universität Klagenfurt, Fakultät für Wirtschaftswissenschaften und Informatik, Klagenfurt, Austria, February 2006.

[252] J. Li (ed.), "SP/CP Service Environment – Intermediate", ICT-ALICANTE, Deliverable D5.1.1, September 2011.

[253] C. Müller and C. Timmerer, "A VLC media player plugin enabling dynamic adaptive streaming over HTTP", in *Proceedings of the 19th ACM international conference on Multimedia*, Scottsdale, Arizona, USA, pp. 723–726, December 2011.

[254] Tears of Steel, "Tears of Steel | Mango Open Movie Project", Home Page, URL: "http://mango.blender.org/", accessed June 11, 2013.

[255] R. Lowry, "Concepts and Applications of Inferential Statistics", R. Lowry, 1998-2013. Available online: "http://vassarstats.net/textbook/", accessed June 21, 2013.

[256] LIVE555.COM, "The LIVE555 Media Server", Home Page, URL: "http://www.live555.com/mediaServer/". Accessed May 10, 2013.

[257] VideoLAN, "VideoLAN - VLC: Official site - Free multimedia solutions for all OS!", Home Page, URL: "http://www.videolan.org/". Accessed May 10, 2013.

[258] Markus Waltl, "[ALICANTE] Layered Multicast Adaptation - YouTube", Website, URL: "http://www.youtube.com/watch?v=h4lscvE7lq4", October 2011. Accessed May 10, 2013.

[259] ALICANTE Blog, "ALICANTE Blog", blog, URL: "http://www.ict-alicante.eu/blog/", accessed June 21, 2013.

[260] Y. Zhang (ed.), "Trials and Validation", ICT-ALICANTE, Deliverable D8.3, scheduled for August 2013.

[261] Wireshark, "Wireshark · Go deep", Home Page, URL: "http://www.wireshark.org/". Accessed May 10, 2013.

[262] IETF RFC 4566, "SDP: Session Description Protocol", IETF Request for Comments, July 2006.

[263] G. Xilouris (ed.), "ALICANTE Overall System and Components Definition and Specifications", ICT-ALICANTE, Deliverable D2.1, September 2011.