TECHNISCHE
UNIVERSITÄT
DARMSTADT

QUALITY ADAPTATION IN PEER-TO-PEER VIDEO STREAMING

Supporting Heterogeneity and Enhancing Performance using Scalable Video Coding

Vom Fachbereich Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertationsschrift

von

OSAMA ABBOUD, M.SC.

Geboren am 07. August 1982
in Deir Ames, Libanon

Vorsitz: Prof. Dr.-Ing. Hans Eveking
Referent: Prof. Dr.-Ing. Ralf Steinmetz
Korreferent: Prof. Dr.-Ing. Wolfgang Effelsberg

Tag der Einreichung: 30. Januar 2012
Tag der Disputation: 30. Mai 2012

Hochschulkennziffer D17
Darmstadt 2012

ABSTRACT

Peer-to-Peer (P2P) techniques for Video-on-Demand (VoD) have attracted a lot of attention recently due to their high potential at improving the performance of today's multimedia systems. Evidently, P2P-based streaming systems already serve thousands of videos to millions of users every day. These large-scale systems are possible because client devices act not only as consumers but also as providers when using P2P.

Nonetheless, current P2P VoD systems still suffer from a major limitation: such systems try to provide the same video quality to all users even if they have different devices with a wide spectrum of resources. We believe that it is of essence that future P2P VoD systems are *quality adaptive*, meaning that different devices may retrieve different video qualities based on available resources. In this thesis, we develop quality adaptation concepts and algorithms essential for improving the Quality of Service (QoS) of P2P VoD streaming systems.

We proceed towards our goal with four major steps.

1. We design a novel P2P VoD streaming system based on techniques and architectures that help in reducing server resource utilization. This is achieved using distributed peer and block management algorithms that use more information about the neighboring peers, e.g. their bandwidth and playback state. Based on a capacity model, we additionally develop prefetching and upload strategies that help the system adapt to fluctuations in the number of peers and their resources.

2. We develop concepts and mechanisms that enable the use of Scalable Video Coding (SVC) in P2P VoD systems to achieve quality adaptation. Using SVC, we develop a two-stage quality adaptation algorithm that matches the video quality with available local and system resources. Additionally, it adapts to the heterogeneity of Internet devices by considering static and dynamic resources such as screen resolution, throughput, processing power, and availability of video blocks. Extensive evaluations show the superiority of our quality adaptation algorithm compared to classical approaches. Furthermore, we show that shorter playback delays can be achieved in return for reducing the video quality. In other words, we find that the session quality (start-up delay, video stalls) and delivered SVC quality (layer switches, received layers) exhibit a tradeoff.

3. We address quality adaptation at the system level and inside the networks by investigating the potential of making networks media-aware, i.e., managing resources according to the importance of different parts of the SVC video. Subsequently, we develop a media-aware system based on routing elements that allocate resources depending on the SVC video characteristics and video block playback deadlines. Using extensive evaluations, we show that it is possible to reduce playback delay by up to 52% during congestion while also alleviating the side effects of congestion on user perceived quality.

4. Finally, we design mechanisms that use Quality of Experience (QoE) metrics in the P2P VoD system to enhance its performance. The decision of which SVC quality to choose has so far been driven by QoS metrics, such as throughput. In this thesis, we expand the classical selection algorithms to consider the QoE of the different SVC qualities. The SVC video quality is assessed using objective techniques, which are highly scalable in comparison to subjective methods. We show that by making peers favor certain SVC qualities with high objective QoE, it is possible to enhance the performance of the entire system in terms of session and video quality, while content providers are able to reduce up to 60% of their server costs.

KURZFASSUNG

Peer-to-Peer (P2P)-Architekturen für Video-on-Demand (VoD)-Dienste erregen seit geraumer Zeit großes Interesse auf Grund des von ihnen ausgehenden Potentials zur Leistungssteigerung heutige Multimediasysteme. Schon heute bedienen P2P-basierte Streamingsysteme Millionen von Benutzern mit tausenden von Videos. Diese Dimensionen werden möglich, da die Endgeräte der Nutzer nicht nur als reine Konsumenten, sondern auch als aktive Anbieter der zu verteilenden Inhalte auftreten.

Gleichwohl leiden heutige P2P-basierte VoD-Systeme unter einer bedeutenden Einschränkung: Sie versuchen, allen Benutzern die gleiche Videoqualität bereitzustellen, auch wenn diese sehr heterogene Endgeräte mit einem breiten Spektrum an Ressourcen nutzen. Diese mangelnde Anpassungsfähigkeit existierender P2P-basierter VoD-Systeme hat zur Folge, dass ressoucenschwache Teilnehmer nicht in der Lage sind, am VoD System teilzunehmen oder angeforderte Videoinhalte wiederzugeben. Wir sind daher überzeugt, dass eine wesentliche Eigenschaft zukünftiger P2P-basierter VoD-Systeme in der Möglichkeit zur Qualitätsanpassung liegt, so dass unterschiedliche Endgeräte auch unterschiedliche Videoqualitäten, jeweils passend zu ihren Ressourcen, empfangen können. In dieser Arbeit entwickeln wir Konzepte und Algorithmen zur Qualitätsadaption, welche wesentlich für die Verbesserung der Servicequalität von P2P-basierten VoD-Systemen sind.

Wir nähern uns diesem Ziel in vier Schritten:

1. Zunächst entwerfen wir ein neuartiges P2P-basiertes VoD-Streamingsystem, welches auf Techniken und Architekturen zur Reduktion von Serverressourcen basiert. Dies wird durch die Verwendung von verteilten Peer- und Block-Management-Algorithmen erreicht, welche mehr Informationen über benachbarte Peers, wie bspw. deren Bandbreite und Abspielstatus, einbeziehen. Zusätzlich entwickeln wir, aufbauend auf ein Kapazitätsmodell, Prefetching- und Upload-Strategien, welche dem System helfen, sich an Schwankungen in der Anzahl verbundener Peers und deren Ressourcen anzupassen.

2. Des Weiteren entwickeln wir Konzepte und Mechanismen zur Nutzung von Scalable Video Coding (SVC) zur Realisierung von Qualitätsanpassungen in Rahmen des entworfenen Systems. Mit Hilfe von SVC entwickeln wir einen zweistufigen Algorithmus zur Anpassung der Videoqualität an verfügbare lokale und systemweite Ressourcen. Zusätzlich sorgt dieser Algorithmus für eine Adaption an die Heterogenität von Endgeräten durch die Berücksichtigung von statischen und dynamischen Ressourcen, wie der Bildschirmauflösung, dem Datendurchsatz, der Berechnungskapazität sowie der Verfügbarkeit von Videoblöcken. Ausgiebige Evaluationen zeigen die überlegenheit unseres Algorithmus im Vergleich zu klassischen Ansätzen. Zudem zeigen wir, dass kürzere Abspielverzögerungen erreicht werden können, wenn im Gegenzug die Videoqualität reduziert wird. So stellen wir fest, dass die Sitzungsqualität (initiale Abspielverzögerung, Wiedergabeunterbrechungen) und die ausgeliefer-

te SVC-Qualität (Anpassung der Qualitätsstufe, empfangene Qualitätsstufe) in gegenseitiger Abhängigkeit stehen.

3. Wir adressieren Qualitätsanpassungen sowohl auf Systemebene, als auch innerhalb des Netzwerks. Hierzu analysieren wir, inwieweit das Netzwerk Eigenschaften der übertragenen Videoinhalte berücksichtigen kann, um Ressourcen anhand der Wichtigkeit einzelner Teile des SVC-Videos zu verwalten. Wir entwickeln ein solches System unter Verwendung von Routern, welche Ressourcen in Abhängigkeit der Charakteristiken von SVC-Videos und der Deadline von Videoblöcken alloziert. Durch ausgiebige Evaluationen zeigen wir, dass die Abspielverzögerung während Engpässen im System um bis zu 52% reduziert und gleichzeitig Seiteneffekte auf die vom Benutzer wahrgenommene Qualität gelindert werden können.

4. Abschließend untersuchen wir den Einfluss der Benutzung von "Quality of Experience"(QoE)-Metriken in dem entwickelten System. Hierbei erweitern wir den klassischen Algorithmus zur Auswahl der Qualitätsstufen, so dass er die vom Benutzer wahrgenommene Videoqualität ebenfalls berücksichtigt,. Dabei wird die Videoqualität mit Hilfe von objektiven QoE-Metriken beurteilt, sodass dieser Ansatz im Gegensatz zu subjektiven Methoden sehr gute Skalierungseigenschaften aufweist. Wir zeigen, dass es durch die Bevorzugung von Qualitätsstufen mit hoher objektiver QoE möglich ist, die Performanz des Gesamtsystems in Bezug auf die Sitzungs- und Videoqualität zu verbessern, während Inhalteanbieter gleichzeitig ihre Serverkosten um bis zu 60% reduzieren können.

ACKNOWLEDGMENTS

Although this thesis has only one author, many people have contributed and supported its writing and completion. To those people, I give my appreciation.

First of all, I would like to thank Prof. Ralf Steinmetz for giving me a chance to make my Ph.D. under his supervision and at his department. I would also like to thank Prof. Wolfgang Effelsberg for providing valuable feedback especially in the last phase of my work.

I am grateful for all the colleagues at the Multimedia Communications Lab (KOM) who have, over more than four years, provided the best possible personal and technical environments for making a Ph.D. My special thanks goes to my previous and current team members and friends, Aleksandra Kovacevic, Kalman Graffi, Konstantin Pussep, Sebastian Kaune, Christian Groß, Dominik Stingl, Julius Rückert, and Matthias Wichtlhuber. Without those people it would not have been possible to complete this thesis. I also thank Thomas Zinner for all the joint work and common publications.

A special thanks goes to my parents, my sister, and my three brothers. Although you live far away, you have always provided unlimited support. Last but not least, I am forever grateful to my young family who has provided amazing direct and indirect support, many thanks to you Sabah and Daniel.

Darmstadt 2012

CONTENTS

1 INTRODUCTION

It is the age of Internet-based multimedia. People are increasingly using streaming applications such as YouTube [7] and PPLive [76] to watch live and on-demand content over the Internet. Several phenomena, especially social networking, have enriched the demand for Internet-based multimedia, where users share and consume user-generated videos. This demand has led and is still leading to an enormous growth of multimedia traffic on the Internet [1]. According to a study conducted by Cisco [6], multimedia in general and video streaming in particular is said to constitute 90% of all Internet traffic, with a high expectation of steady increase in the future.

A main development drive for streaming applications is the constant demand for better video quality. Users are and will always expect to receive higher and higher quality videos [40]. This is made possible by the steady increase in available capacity of end user connections. The current state of the art of last-mile connections - whether vDSL or fiber optics - allows users to watch videos with bit rates reaching up to 100 Mbit/s. This rate is almost enough to stream an ultra-high definition video, 16 times larger than the *high definition* we know today.

In order to catch-up with this growth, various service architectures ranging from client/server to distributed cloud approaches have been developed and deployed. YouTube, Hulu, and Netflix are examples of client/server systems that have revolutionized the concept of Internet-based multimedia delivery. YouTube, for example, specializes in user-generated content, where it is possible to watch a huge amount of videos generated by other users. The other mentioned service providers extend the offered content to series and full-length movies. Although client/server-based approaches provide, in general, good performance with high availability rates, they inflict enormous costs, as in the case of YouTube where the cost for providing this service is estimated to be one million US dollars per day [40, 2]. These costs would increase drastically if more videos were offered in higher qualities.

Much research has gone into investigating alternative architectures that can alleviate the downsides of client/server systems. One promising architecture is peer-assisted delivery that relies on a delicate balance between client/server and Peer-to-Peer (P2P) techniques [88]. The application of the P2P paradigm as an assistance for client/server approaches enables the use of end user devices (called peers) to reduce costs and load on content servers. Additionally, P2P offers desirable properties such as self-organization and resource scalability [88, 46].

1.1 MOTIVATION

In this thesis, we focus on P2P[1] Video-on-Demand (VoD) streaming, where users can browse and request video content anywhere at any time. P2P VoD systems are becoming more popular for commercial and non-commercial usage, since they have the reliability of servers and the scalability of P2P. Additionally, it is possible to have only a limited amount of server resources since peers assist the servers by uploading video data they have already downloaded. Thereby, P2P VoD allows for either a higher number of supported peers, or a higher video bit rate than traditional VoD systems.

P2P VoD is becoming especially attractive for content providers due to the increasing bandwidth requirements posed by high-quality video. It is believed that servers alone cannot support such a service to millions of users in a cost-efficient way [40]. Additionally, there have been extensive efforts to standardize the use of P2P techniques for multimedia delivery. The Internet Engineering Task Force (IETF) is working on a P2P Streaming Protocol (PPSP)[2] to pave the path towards standardized mechanisms for P2P *live* and *on-demand* streaming.

Nonetheless, P2P VoD systems still suffer from one major issue: these systems usually offer a single video quality to all clients; they are not adaptive. Users use heterogeneous devices that range from personal computers to Internet-enabled television sets and mobile phones. These devices are heterogeneous not only in their display characteristics, but also in the type of connections they have, where bandwidth, delay and reliability vary drastically. Therefore, a static selection of video bit rates leads to major problems for the users: on one hand, peers with weak resources are directly excluded from the system while, on the other hand, those with strong resources cannot enjoy better quality even though they have enough resources.

Therefore, there is a need to support the wide spectrum of resources within the same video delivery system and enable quality adaptation, i.e., the ability to actively change the received video quality based on the available resources of a peer and of the network.

1.2 QUALITY ADAPTATION

In current systems, it is quite challenging for the content providers to decide on a good target bit rate for video quality. While lower bit rates are usually preferred in order to support a wider spectrum of end user devices, higher bit rates are becoming increasingly popular. To address this issue, YouTube re-encodes every video at different qualities to match the bit rate with the demand of different devices. This, however, generates some issues. The first is that no real-time adaptation is possible, i.e., the user devices cannot switch the video quality on the fly in case of a sudden drop in resources. Second, if YouTube were to use P2P, peers that are streaming different video qualities would not be able to exchange video data with each other. A main question

1 For the sake of brevity, we use the term P2P instead of peer-assisted.
2 https://www.ietf.org/mailman/listinfo/ppsp. Accessed January 2012.

arises: how can we build an architecture that, at the same time, enables quality adaptation and support for different user devices?

Prominent approaches for quality adaptation are based on the idea of using multi-layer video coding techniques so that video quality can be scaled according to the needs of user devices. In this thesis, we make use of the state of the art in multi-layer coding, namely Scalable Video Coding (SVC), to enable quality adaptation in P2P VoD systems.

Our objective is to provide concepts and algorithms that address the problems of adaptation in a VoD system based on SVC. We proceed in three major steps or building blocks, which are presented in Figure 1. This figure depicts a simplified model of a P2P VoD system. The main entities in this model are: users, end user devices (peers), and network elements (routers).

1. QUALITY ADAPTATION USING SVC. This building block constitutes all the algorithms and mechanisms that are required in order to adapt the video quality to available resources as seen from the peers at the *edge*. These algorithms work in a distributed manner and react to resource changes at the edge of the network. Although those algorithms do not have global knowledge but only a local view, they are essential in reacting to the various changes in the network and the VoD system.

2. MEDIA-AWARE NETWORKING. This building block constitutes adaptation algorithms that can be applied to the *core* of the network in order to perform better resource allocation. This is essential in case sudden congestion occurs in the network. We investigate the feasibility and advantages of adding media-awareness to network elements, such as routers, in the context of an SVC-based P2P VoD system.

3. QUALITY-OF-EXPERIENCE-AWARE ADAPTATION. This building block considers the impact of the SVC video quality on *user* Quality of Experience (QoE). Here, we make use of objective QoE estimation of SVC videos in

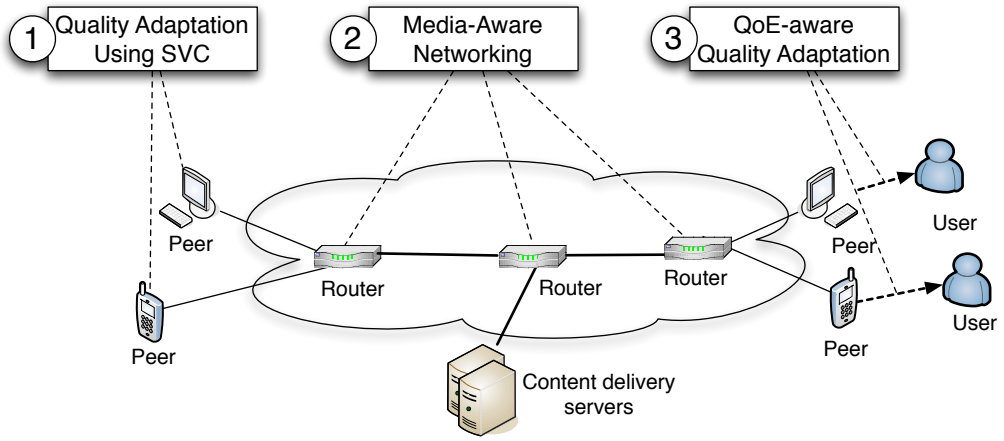


Figure 1: The building blocks to achieve quality adaptation in P2P VoD systems. The system employs quality adaptation using SVC at the peers (edge), media-aware networking in the network (core), and QoE-aware quality adaptation (user).

order to improve the performance. The QoE estimations are used to make better decisions on which video quality to select, thereby enhancing performance from the user point of view while improving the efficiency of the streaming system.

1.3 RESEARCH CHALLENGES

It is challenging to build quality adaptation mechanisms for P2P VoD that consider the *edge* and *core* of the network while also using *user* QoE. Challenges come not only due to the highly dynamic nature of P2P systems, but also because such systems extend to different communication layers. We address the following challenges.

1. *The Need for Quality of Service (QoS)*. A main challenge is the demand for QoS in VoD systems. Unlike classical file-sharing applications, videos in a VoD system are played while being downloaded [45]. Therefore, if video pieces are received after their deadline, playback is affected and user experience is degraded. In streaming systems that use reliable transport protocols, such degradation is visible as a playback stop or stall. We consider a solution that adds more flexibility and degrees of freedom to the VoD system. We achieve this by using SVC, which enhances performance even if resources are dynamic.

2. *Limited and Dynamic Resources*. Peers have limited upload capacities [4, 28]. This can be attributed to the fact that last-mile connections have an asymmetric bandwidth characteristic. Evidently, the Internet was not designed for P2P applications. Due to limited upload capacities, it is not possible for a peer to utilize all of its download capacity when streaming from only one peer. Therefore, a dynamic set of serving peers is required for efficient video delivery, leading to a higher degree of dynamics in the overall system.

3. *Peer Heterogeneity*. Peers are heterogeneous with various resources in terms of bandwidth, screen resolution, and processing power [53]. Therefore, accommodation of streaming bit rates for the entire system is necessary. Differences in bandwidth can be static, caused by different link capacities, as well as dynamic, induced by the network. Peer heterogeneity includes both upload and download capacities, making the prediction of the video quality that can be downloaded or uploaded complex to manage.

4. *Sudden Network Congestion*. Streaming systems may suffer from packet loss or delay due to network congestion. It is challenging to ensure that peers retain connectivity and cope with congestion in the network. It is especially noticeable that much of the Internet routing elements are media-agnostic, i.e., do not explicitly distinguish multimedia traffic from other traffic. Evidently, they were built with classical applications in mind, such as web and email. When congestion occurs, it is challenging for the peers at the edge to overcome their effects even with sophisticated adaptation algorithms since their information about such events is limited.

5. *Quality Adaptation in P2P Systems.* Quality adaptation mechanisms are straightforward in case of centralized systems where video streams traverse a well-known path between the server and the clients. In such systems, the important factors, such as bandwidth, can be accurately measured to some extent. However, this does not hold for P2P systems as video data is retrieved over multiple connections simultaneously. Therefore, allocated resources can change without warning and in a cascaded manner so that multiple connected peers may undergo performance degradations one after the other. This makes the design of quality adaptation mechanisms for P2P VoD systems quite challenging. Such quality adaptation mechanisms have to adapt to various dynamic parameters all at once, without having much information about the uploader peers, their streamed quality, and download resources.

6. *Performing Adequate Adaptation.* Performing the right amount of quality adaptation is challenging. On one hand, it is desired to stream the highest possible video quality. On the other hand, the video quality should be stable without too many oscillations. It is quite challenging to find a compromise between streaming high video qualities and a stable system that does not make excessive use of quality adaptation.

7. *Quality of Experience Considerations.* Quality adaptation is based on changing the video quality depending on the available resources. Usually, only information and parameters from lower-layer QoS are used in the decision process. The most prominent parameter in this context is the download throughput. Although, many systems base the quality decision on such pure QoS parameters, it is of essence to consider how the *users* perceive the different video qualities. Integration of QoE metrics is quite challenging due to the huge overhead such QoE estimation methods inflict. Many methods require costly data transmission or processing at the user devices putting much strain on limited resources. We aim at integrating QoE information into our system without overloading the resources of user devices and the network.

1.4 RESEARCH GOALS

The goal of this thesis is to improve the video quality and playback performance of P2P VoD systems by supporting the heterogeneity of Internet devices. We use quality adaptation based on SVC and QoE metrics so that peers can retrieve and watch videos on devices with heterogeneous resources within a dynamic system. Quality adaptation is a general term that extends into different areas of the streaming system, starting from algorithms running at the peers (edge) to those running the network (core). Quality adaptation mechanisms should also be aware of how users perceive the SVC video and make decisions based on QoE metrics.

Our objective can be decomposed into three main research goals.

- The first goal is to develop *quality adaptation* mechanisms that improve the performance of P2P VoD systems, addressing Challenges 1, 3, 5, and 6. We aim at improving performance by using SVC to match resources to the video quality and reduce video playback delay. Thereby, we want to show the superiority of the designed system in comparison to a non-adaptive system, where a standard video codec is used. Additionally, we want to understand the impact and tradeoffs that the adaptation speed has on the overall system.

- The second goal is to investigate the impact of using *media-aware* network management techniques on the system performance, addressing Challenges 2 and 4. Here, we ask the question: how much can performance be improved if the routing elements become media-aware? New algorithms are especially needed when congestion occurs in the network. It is interesting in this context to compare a media-aware network with a classical QoS-based one and evaluate the different opportunities a routing element has in resource allocation.

- The third goal is to improve the performance from the user point of view by using *QoE estimations* of SVC videos, addressing Challenges 5, 6, and 7. We aim to take a light-weight approach where user devices do not waste any bandwidth or processing power in estimating the QoE. Additionally, we want to evaluate the impact of using QoE-aware adaptation on the system performance and capacity.

1.5 OUTLINE

The rest of this thesis is structured as follows. In Chapter 2, background information required to understand the contributions of this thesis is presented. Chapter 3 reports on other work related to the topic of this thesis. Here we classify earlier work according to four areas: adaptive P2P VoD, P2P video streaming and SVC, media-aware networking, and objective QoE in video streaming. We present other systems related to ours and show the major differences to our approach. In Chapter 4, we introduce our basic system design. This chapter gives an overview of the core functions upon which we build our main system and mechanisms. Chapter 5 details the main contributions of this thesis. This chapter is divided into three major sections. First, we present our quality adaptation algorithms that use SVC. Then, we discuss our media-aware mechanisms that are used within the network to perform resource allocation in case of congestion. Finally, we show how we utilize QoE metrics to improve the performance for the users and content providers. In Chapter 6, we report on the evaluation of our system. We first present the metrics used for evaluating our system, followed by details on our simulation setup, highlighting the tools and configurations we have used. Last, we present the simulation results as well as our analysis and main conclusions. Finally, Chapter 7 summarizes this thesis, detailing our major contributions. We additionally discuss further research points that can be addressed in future work.

2 BACKGROUND

In this chapter, we present the background on basic concepts required to understand the content presented in this thesis. We first provide an overview on basics of multimedia delivery with focus on Peer-to-Peer (P2P) architectures, namely prominent topologies and strategies. Subsequently, we detail aspects related to video coding where we detail the Scalable Video Coding (SVC) standard, a building block of our system. Finally, we present the basics on Quality of Experience (QoE) assessment and how objective QoE metrics can be used to improve the performance of the system.

2.1 BASICS OF MULTIMEDIA DELIVERY

Videos can be provided either in the download mode or in the streaming mode. In the download mode, users can play the videos only after the whole file has been downloaded. On the other hand, video streaming refers to playback of the video file during the download process. Therefore, streaming allows for lower waiting times, as playback can start as soon as the playback buffer is full (typically some 7 seconds of video data). Therefore, streaming introduces some timing requirements for the streaming system. Since video pieces - which are small video transmission units - are required to arrive on time before their playback deadline, the network architecture and mechanisms must be carefully chosen in order to ensure that bandwidth, delay, and loss requirements are fulfilled.

2.1.1 *Live versus On-Demand Streaming*

There are two kinds of multimedia streaming applications: live streaming and Video-on-Demand (VoD) streaming [58]. Live streaming is well known from television, where content is generated and sent to the viewers simultaneously. In a VoD system, on the other hand, the freshness of the content is not important. Videos are pre-recorded and can be played by different viewers asynchronously. Therefore, the video pieces in a VoD video can be distributed in any order and even be pre-loaded and cached in the network. Further, users can seek forward and backward, which again encourages some kind of pre-caching mechanisms. These characteristics of VoD systems enable the possibility to design new architecture and mechanisms for enhancing performance as provided in this thesis.

2.1.2 *P2P Content Distribution*

Many current video streaming systems are based on the client/server architecture with servers providing the video content. However, such systems either

provide low quality content, or introduce costs high enough to prevent deployment when trying to provide high quality. To shift load from servers and to allow for reduced costs, streaming solutions based on P2P architectures have been considered.

The focus of this thesis is on P2P systems as they provide advantageous properties, such as self-organization and resource scalability. Other benefits are distributed storage space, a large amount of resources for computation, and redundancy of network paths. This all leads, with the adequate distributed techniques, to an efficient and robust video delivery. High costs due to bandwidth needs at the server are a major drawback of client-server architectures. With an increased number of clients, the required bandwidth grows proportionally, creating extreme costs for scalability [60]. Therefore, P2P techniques aid the relaxation of bandwidth burdens placed on servers, as peers act both as providers and consumers, making it possible to provide low cost video streaming services [79].

Currently BitTorrent [24] is the de-facto P2P content distribution system. It is mainly used for the distribution of content, such as large video files or software updates. In BitTorrent, files are broken into pieces and as soon as one peer has downloaded its first piece, it can start acting as a providing peer. When a new peer joins a torrent, i.e., a content distribution overlay, it contacts the so-called tracker to obtain a list of peers already in this overlay. As this subset of peers is randomly chosen, it considers neither the heterogeneity of peers nor does it apply any load-balancing mechanism. What makes the BitTorrent system quite stable is the use of the rarest-first piece selection strategy. This means peers would select pieces that are rare in their neighborhood, thereby making those pieces better distributed. This strategy greatly enhances the performance of BitTorrent systems as long as the order of pieces is not important, which is the case for non-streaming applications.

2.2 PEER-TO-PEER STREAMING

Since a large portion of the data exchanged over BitTorrent is video content, much attention started going into performing live and on-demand video streaming based on this system. Here systems such as BiToS [96] and Octoshape [3] work simply by modifying the piece selection algorithms and replacing the rarest-first strategy with an earliest-deadline strategy, so that pieces required for playback are requested first.

Other researchers preferred building the streaming system from scratch without any dependency on BitTorrent mechanisms. In the following subsections, we go over the most prominent topologies used for P2P video streaming. After that, we present scheduling strategies used to ensure that video pieces arrive on time at the receivers.

2.2.1 *Streaming Topologies*

A streaming topology, also called overlay topology, is defined as the set of methods for interconnecting the different nodes for transmitting video data.

In general, there are two types of overlay topologies for video streaming: push-based multicast tree and pull-based mesh. In push-based solutions [60], peers are structured in a tree topology with content providers positioned at the top of the tree pushing video data down the tree towards the leaves. In pull-based systems [96], a peer actively requests parts of the video from peers that have already downloaded it thus forming a mesh topology. Mesh-based topologies are characterized by lower overlay maintenance costs and higher flexibility in piece selection.

Next, we provide an overview on both, highlighting their major advantages and drawbacks. At the end we provide a short discussion on the topology we use in our system and the arguments we base our choice on.

Single Tree Topologies

In a single tree topology, peers are arranged into a hierarchy as presented in Figure 2. The streaming source, usually a server, is located at the top or root of the tree, and video data is actively pushed towards the peers at the bottom of the tree, also called *leaf* peers [60]. When peers join, they are inserted at a specific tree level and their parent peers forward the video content to them. Examples for the use of single tree streaming are Overcast [34] and ESM [23].

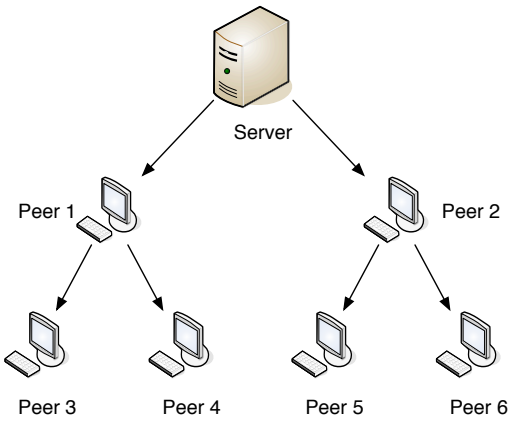


Figure 2: A single tree topology.

Tree depth is defined as the number of levels the tree has. Therefore, the smaller the tree depth, the lower is the delay for the leaf peers (those situated at the bottom of the tree). To achieve a small tree depth, peers need to have a large fan out degree, which specifies how many child peers each peer has. Since a peer has only a limited upload capacity, its maximal fan out degree is constrained.

ISSUES OF SINGLE TREE TOPOLOGIES. While single tree topologies have advantages such as simple design and low management overhead, there exist several shortcomings. Single tree approaches are strongly affected by ungraceful peer departures since after the departure of a peer all of its child peers lose their connection to the video source, therefore, their media quality suffers significantly. The higher in the tree a failing peer is located, the higher is the degree of disruption.

Another issue in tree topologies is that leaf nodes do not utilize their upload bandwidth, as they do not have any child peers. This leads to a reduced efficiency, as the number of leaf peers can be rather high [57]. In [33], it is explained that system throughput is limited by the link with the lowest capacity among all links on the path from the source to the leaf peers, which limits system efficiency.

An additional issue of tree topologies comes from the assumption that peer upload bandwidth is double the video bit rate, since a peer must typically upload twice as much as it downloads (in this case when the fan out degree is two). This is, however, contradicting with average link capacities, where upload capacities are usually much smaller than download ones, rendering classical tree topologies unusable in real life scenarios.

Evidently, there is a need for additional algorithms to overcome the effects of sudden peer departure, also called churn, in single tree systems [33]. Therefore, multi-tree topologies have been proposed.

Multi-tree Topologies

In multi-tree approaches, a stream is divided into several substreams at the streaming server. For each substream, a subtree is created. A peer must join all subtrees and receive all substreams in order to decode the video stream. The power of this approach comes when each peer joins each subtree at a different position. Figure 3 shows an example of how peers can be positioned within the different subtrees. If a peer is in an inner node in a single tree, i.e., not a leaf node and high in the tree, all of its upload bandwidth is thoroughly utilized. In this case, the frequency of being placed as an inner node in a multi-tree topology should depend on the available bandwidth [60]. Although having multiple subtrees and the possibility to send video data in different substreams over different paths, delivery paths usually remain static just like in single tree approaches.

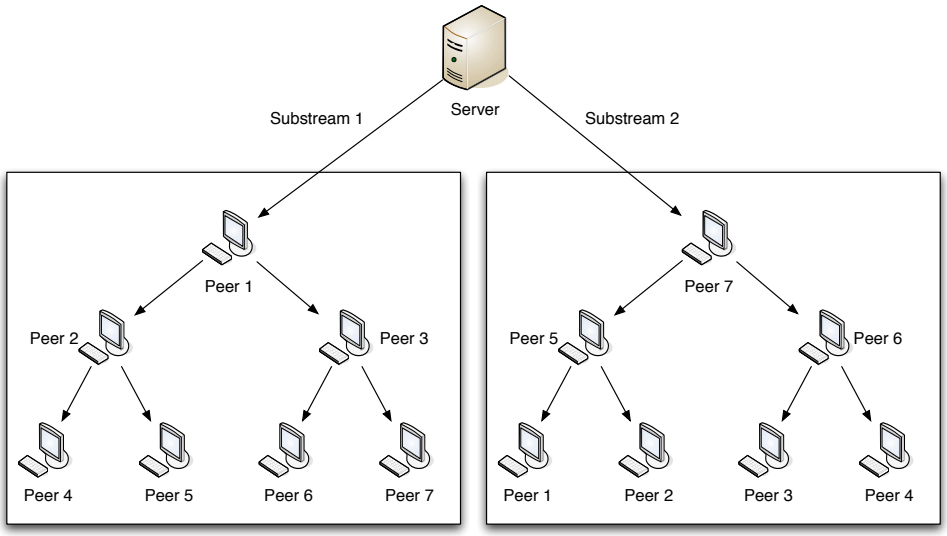


Figure 3: Peers participating in a multi-tree topology. Here, each peer joins two subtrees.

TOPOLOGY CONSTRUCTION. Construction can be done either in a randomized or a deterministic manner. Using a randomized construction, a search for nodes with spare bandwidth is started in each tree. Then one of the hit peers is randomly chosen as parent peer for a child peer. For deterministic construction, each node is assigned to be an interior node in exactly one tree to shorten the tree. Such an construction leads to increased tree diversity and, therefore, more robustness.

As a summary, multi-trees are more resilient compared to single trees, and the upload link capacity of peers is better utilized [59]. In addition, they give rise to more interesting cross-layer techniques, such as media coding and Forward Error Correction (FEC).

Mesh Topologies

Mesh topologies, such as the one shown in Figure 4, enable massive parallel content distribution among peers. They are based on self-organization of nodes in a directed mesh and do not have a static topology [60]. Connections are based on availability of content and bandwidth. While peers in a mesh have more links and thus better connectivity, the mesh content delivery scheme, which is known as swarming, is more complicated [33].

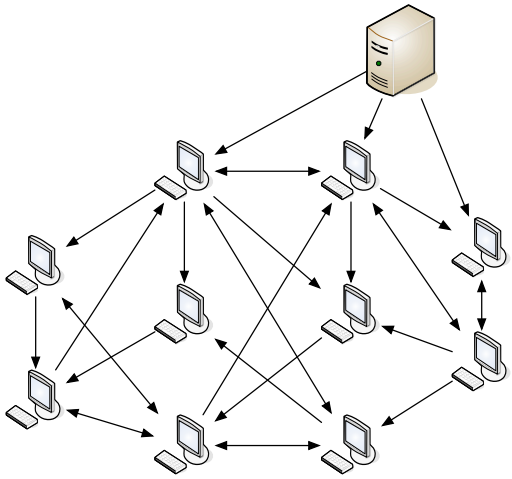


Figure 4: An example mesh topology.

To allow for mesh streaming, a video file is subdivided into many small pieces typically ranging from 32 kB[1] to 512 kB. However, piece sizes of several megabytes have also been used especially for high definition content [104].

Every peer would request the pieces about to be played out from other peers in its neighborhood. To find out which peer has which piece, so-called buffer maps (bitmaps of available pieces) are periodically exchanged between the peers in the same neighborhood.

Mesh topologies can be unstructured or structured. In unstructured meshes, peers are connected to randomly chosen peers to provide more neighbors and different delivery paths. This leads to robustness when failures occur, since every piece can be obtained from other peers in a simple way [59]. Examples

1 kB: kilo Bytes

of unstructured meshes are PRIME [63] and CoolStreaming [106]. In structured meshes, on the other hand, peers are typically arranged into clusters, with the majority of links being established within the same cluster. The peers, which connect the different clusters can be considered as super peers and should have good bandwidth and availability characteristics [33].

The main disadvantage of mesh-based topologies is the overhead that comes from the periodical exchange of buffer-maps and status messages among the peers. These messages can become rather large especially with long or high definition videos. Decreasing the number of pieces relaxes this overhead, however, at the cost of more dependence on peers that a certain piece has been requested from since a piece then can become rather large.

The advantages of mesh-based approaches are low costs and simple maintenance of the network structure. Compared to tree-based systems, these topologies are more resilient regarding the topology in the presence of node failures and departures, as the probability of more paths being available is higher [46]. In [64], Magharei et. al discuss that paths from the source to individual peers are more stable than in tree based schemes and that a peer's degree of stability increases the longer it remains in the overlay.

THE CHOICE OF MESH TOPOLOGIES. When compared to tree-based approaches, mesh topologies have been shown to be superior in terms of performance as well as their applicability to real applications [64]. Therefore, we have decided to use mesh topologies for our system.

Next, we provide an overview over scheduling algorithms in multi-source streaming systems. Scheduling algorithms in general are used in all topologies and are the core of P2P streaming applications.

2.2.2 *Scheduling Algorithms*

Scheduling algorithms are used to determine which video pieces should be transmitted or retransmitted and at what time these transmissions should be carried out. The scheduling decisions also include dropping of pieces if their transmission is not possible due to restricted resources. The aim is to maximize the quality of the video stream without wasting network resources. A *good* scheduling algorithm would help not only in reducing end-to-end delay but also in achieving more robustness and adaptation to system dynamics.

Figure 5 shows an abstract representation of scheduling algorithms running at a peer forwarding video packets. This peer is also called *intermediate peer*. The scheduling algorithm uses rate-distortion information when deciding on which video pieces to drop. Rate-distortion information is a quantification of the amount of distortion inflicted on the video if a piece were to be lost.

Structure and transmission characteristics (e.g., available bandwidth at the different links) of the network should also be considered. During scheduling, not only the packet transmission order has to be determined, but also a prioritization among a peer's descendants should be considered. This prioritization is essential especially when using tree topologies as the receiver peers would be affected in case of piece loss or delay. Furthermore, successive transmissions

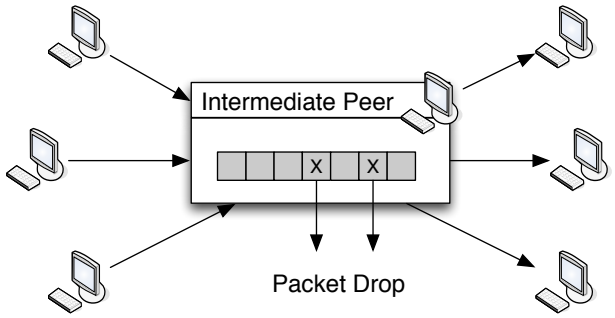


Figure 5: Based on distortion information, (intermediate) peers can make rate-distortion optimized scheduling decisions for incoming pieces.

can be spaced to avoid congestion at bottleneck links as well as to limit delay of control packets. Scheduling algorithms are crucial for achieving a good quality of service due to the strict timing requirements of streaming applications. Moreover, peers are heterogeneous with variable available resources. One peer in a network might have a high amount of available bandwidth and, therefore, is able to receive all video pieces in time, resulting in high media quality. Simultaneously another peer in the same network can have only small amount of available bandwidth and thus the scheduler has to determine which video pieces are the most significant to ensure that the video can be watched at a certain quality level.

In the next section, we provide a review on prominent multi-layer video coding techniques. We first present a classification of such codecs while elaborating on two major approaches: multiple description coding and scalable video coding.

2.3 MULTI-LAYER VIDEO CODING

Video data is currently being stored in a digital form thanks to a wide range of software known as video codecs. These codecs encode/decode video data with the objective of providing good video quality along with a reduced amount of data that has to be stored and transmitted. Therefore, the objective is to provide a compression scheme that achieves a tradeoff between the quality of the video and the quantity of data representing it, i.e., maximum compression ratio. AVC/H.264 [43] and VP8 [86] are some examples of commonly used video codecs.

The Internet is gradually being used for video streaming. However, the Internet was never built with such applications in mind. Problems such as transmission errors, packet loss, and congestion hinder efficient and reliable streaming systems. What makes things worse is that much of currently used video codecs are single-layered, e.g., AVC/H.264. Although those codecs provide high compression rates, they are generally not suitable for dynamic environments. Additionally, users are accessing such streaming systems with heterogeneous devices ranging from powerful machines to mobile devices, thereby stressing the need for more adaptive video codecs.

To address those issues, multi-layer video codecs were developed. They are based on the idea of encoding a video in such a way that different quality can be retrieved from the same global stream. Additionally, even if some part of the video gets lost on the way to the receiver, playback does not have to stop.

2.3.1 *Classification*

In this section, prominent multi-layer video coding techniques are presented. The main objective of multi-layer coding is to add the ability to scale the video stream according to available resources. We have established a classification of the different video coding techniques. This classification is presented in Figure 6.

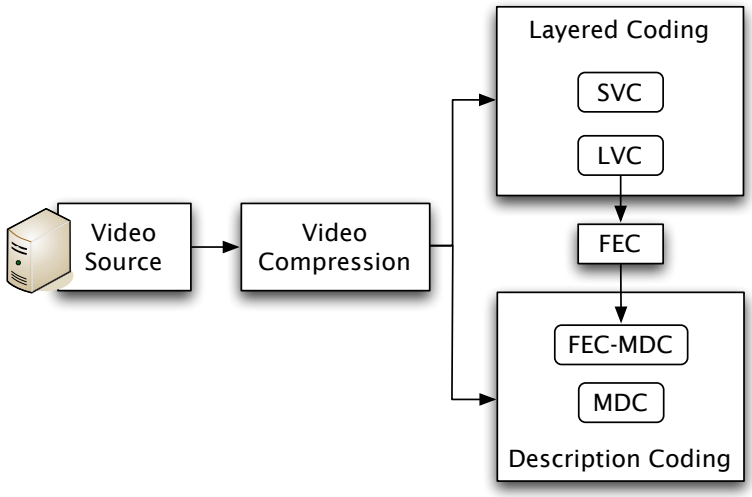


Figure 6: Classification of different possibilities a video encoder can use to generate different types of video structures. These can be generally classified as either layered or description coding.

Raw video data is usually too large to be sent over normal networks, therefore, it is necessary to compress it by exploiting the high redundancy of video frames. For this purpose, video compression can be used to generate either layered or description-based video streams.

Layered Video Coding (LVC), on the one hand, is a term used to describe a large set of coding schemes that organize video data into layers. In LVC, a video stream is structured into hierarchical layers, where the so-called base layer represents the minimal amount of data required in order to decode the video stream. To decode a certain layer, all layers below it have to be available. A prominent example of layered video coding is the Scalable Video Coding (SVC) standard [83]. SVC is presented in Subsection 2.3.3.

Description coding, on the other hand, is based on encoding video data into descriptions that do not depend on each other. The more descriptions are received; the better is the video quality. Most prominent examples of description coding are Multiple Description Coding (MDC) and description coding based on Forward Error Correction (FEC-MDC) [87]. FEC-MDC is generated by FEC-encoding layered video streams. MDC and FEC-MDC are presented next.

2.3.2 *Multiple Description Coding*

Multiple Description Coding (MDC) [87] works by creating several independent so-called descriptions from a video stream. Each description is a video segment that contributes a certain amount to the video quality and can be decoded independently. The more descriptions are received, the higher is the received quality. As a simple example of how MDC works, consider the independent descriptions generated by dividing a video file into even and odd frames. Alternatively, MDC can be realized by dividing video frames into independent sub-pictures by choosing odd and even horizontal and vertical lines of the picture, thus resulting in four descriptions.

MDC descriptions can then be distributed using mesh or tree topologies. MDC works well when combined with multi-tree topologies since then it allows for better resilience against erroneous transmissions and playback distortions [87, 18, 94]. An interesting variant of classical MDC is the so-called *FEC-MDC* as presented in [87]. Similar to MDC, FEC-MDC has the same goal of dividing the video file into multiple independent descriptions but with the distinction of using layered video coding and Forward Error Correction (FEC). The basic structure of an FEC-MDC coded video stream is depicted in Figure 7.

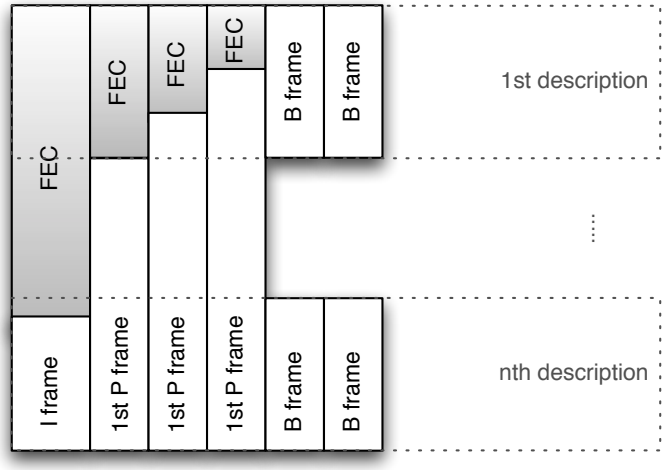


Figure 7: By applying different protection levels to layered video packets, multiple descriptions are created generating an FEC-MDC video stream.

The main advantage of this design in comparison to classical MDC is that it can use a classical video encoding scheme and offer more resilience through the use of FEC. As presented in [87], FEC-MDC starts by encoding the video data using any classical video codec to compress the video stream to generate a layered video streaming. Then, an unequal error protection is applied to the different layers in such a way that correction strength, achieved through more redundancy, depends on the importance of the layer. In comparison to MDC, FEC-MDC requires a certain minimum number of descriptions to be available before any useful video data can be decoded. This is because FEC requires a minimum amount of available data so that it is able to decode the I-frames.

FEC-MDC has a better resilience against packet loss due to its built-in error control techniques and is flexible concerning the used video codec. However, FEC-MDC can inflict high amount of overhead [87].

2.3.3 *Scalable Video Coding*

The H.264/*Scalable Video Coding* (SVC) is the official extension and amendment of the state of the art *H.264/MPEG-4 Advanced Video Coding* (H.264/AVC) standard [43], which is widely used in the Internet for instance by video platforms (e.g., YouTube). AVC is a single-layer codec, which means that different copies of the same video streams have to be encoded to support different end user devices. SVC constitutes the structures and algorithms that make it possible to add scalability features to AVC and the H.264 standard.

In this section, we only scratch on the properties and architecture of SVC required to understand the contribution of this thesis. For further information, the reader is referred to [83, 84].

General Properties of SVC

The Scalable Video Coding (SVC) standard allows for scalability by defining multi-dimensional quality layers. Thus, a video stream can be encoded into a set of substreams or multiple layers each with different quality information. The different layers can be retrieved from a single SVC encoded video file. The lowest layer, called *base layer*, is always needed for decoding the video. With additional layers, also called, enhancement layers, better video quality is generated. SVC is based on three modalities or flavors of scalability: *spatial scalability*, *temporal scalability*, and *quality scalability*. Quality scalability is also called Signal-to-Noise Ratio (SNR) scalability; both terms will be used interchangeably throughout this thesis.

SVC enables not only streaming of different video qualities to heterogeneous devices but also switching the video quality in real time. Therefore, SVC enables the so-called quality adaptation to both local and network resources. Next, we discuss the different scalability dimensions and how they are realized.

The Dimensions of Scalability

The three dimensions of scalability of SVC are now shortly presented. For more thorough information, the reader is referred to the original standard document [43] and publications on the topic [43, 83, 84].

As mentioned above, SVC is an amendment to the state of the art H.264/AVC standard, thereby, SVC incorporates all its core mechanisms with the distinction of supporting scalability. Before we can give information about SVC, some basic understanding of H.264/AVC is required.

In AVC, there are three different frame types, also known from other prominent video codecs, denoted by I-, P-, and B-frames. I-frames include whole pictures and can be decoded independently from other frames of the video. P-frames or prediction frames are frames that can be predicted from other

frames using only incremental information. P-frames can only be used in conjunction with previous frames. B-frames, on the other hand, can use incremental information from both past and future frames. The process of predicting frames from past or future frames is known as *motion compensation*. Information within the picture itself is also compressed based on fixed units known as video blocks. Using the so-called *intra-prediction*, a frame can be reconstructed by decoding blocks from other blocks within the same frame.

We now come to the different scalability dimensions SVC has to offer. These are spatial, temporal, and quality scalability.

SPATIAL SCALABILITY. Spatial scalability denotes the ability to change the video resolution of the video. In Figure 8, we present an example of an SVC video with three standard video resolutions, namely QCIF[2], CIF[3] and 4CIF. When coding and decoding spatial layers, both motion-compensation prediction and intra-prediction are used. Different layers are not independently encoded; their correlation is used in order to enhance the coding efficiency.

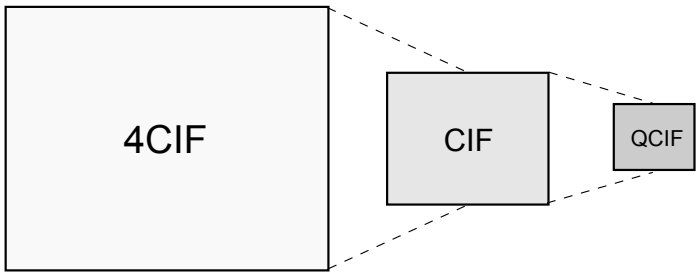


Figure 8: Spatial scalability with example resolutions: 4CIF (704×576 pixels), CIF (352×288 pixels), and QCIF (176×144 pixels).

TEMPORAL SCALABILITY. Temporal scalability denotes the ability to change the frame rate of the video. The frame rate is defined as the number of frames that are displayed in one second.

AVC already had some basic support for temporal scalability. Therefore, SVC simply added some specification on how it can be used in the context of a scalable codec. Since I-frames can be decoded independently from other frames, they constitute the base layer for the temporal dimension.

There are multiple possibilities for realizing the enhancement layer based on different dependency structures of the I-, P-, and B-frames. An example is depicted in Figure 9, in which P-frames depend only on I-frames, while B-frames depend on both I- and P-frames. Within this design, it is possible to first drop B-frames then P-frames to decrease temporal level and bandwidth usage. Other complex dependency structures are also specified in [83].

QUALITY SCALABILITY. Quality scalability, also known as SNR scalability, enables changing the video quality of the individual frames. This is generally realized by using the level of quantization applied to the individual frames.

2 Quarter Common Intermediate Format
3 Common Intermediate Format

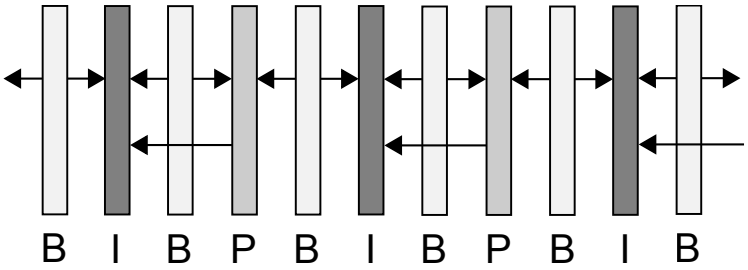


Figure 9: A simple possible frame pattern that would allow for temporal scalability by first dropping all B-, then all P-frames. Arrows show the compulsory dependencies of a frame.

The SVC standard defines three possible variants for this type of scalability. The first is called *Coarse Grain Scalability* (CGS), which is realized similar to spatial scalability where large quantization intervals are used. In this case, the number of quality layers depends on the number of used quantization levels. This kind of scalability has the disadvantage of low coding efficiency. Additionally, the SVC quality layer can only be changed at pre-defined points in the video stream. Further information about this issue is given in [83].

The second variant is called *Medium Grain Scalability* (MGS). It uses some additional techniques to increase flexibility in the quality scalability dimension. Using MGS, it is possible to adjust the selected quality anywhere in the video stream. Nonetheless, this comes at the cost of possible drifts in the video decoding process.

The third and last variant is called *Fine Grain Scalability* (FGS). It gets rid of the possibility of drifts in the decoding process. This is done by removing the dependency of the motion compensation prediction on enhancement layers. In this variant, the motion compensation depends only on the base layer frames.

The SVC Cube Model

The SVC cube model is a conceptual model usually used to visualize and model SVC videos. It helps to understand the interdependency of different quality levels of SVC.

In Figure 10, an example of an SVC video with three quality levels in each scalability dimension is presented. For the spatial dimension, this SVC file has three standard resolutions: namely CIF, SD, and HD. For the temporal dimension, this file has the frame rates: 7 fps, 12 fps, and 24 fps. Finally, the SVC file has three quality levels denoted by Q0, Q1, and Q2.

Each layer is represented by a sub-cube within the SVC cube and denoted by a triplet (d,t,q). The sub-cube in the lower left corner represents the base layer and is denoted by (0,0,0). This layer can be decoded independently from other layers. When decoding a certain layer, all lower layers should be available. For example, to decode the layer (1,1,0), the layers (0,0,0), (0,1,0), and (1,0,0) have to be available. If any of these layers are missing, this layer is not decodable. The sub-cube in the upper right corner is the highest quality layer with HD resolution, 24 fps, and highest image quality. This layer requires all layers in the SVC video so to be decoded.

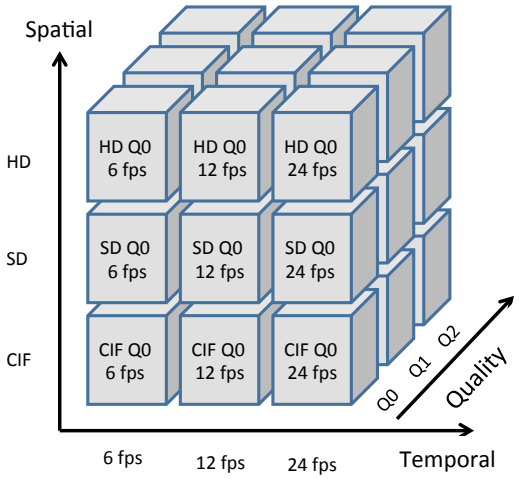


Figure 10: An example of the SVC cube model.

The Choice of Salable Video Coding

In general, the main difference between SVC (or a general LVC) and MDC lies in the inter-dependency of SVC layers and MDC descriptions. Therefore, MDC clearly has the advantage of having flexibility and independence from a base layer, simplifying mechanisms for the network building process. Nonetheless, SVC approaches are the better choice for systems in which media-aware scheduling is used in the network [19]. MDC additionally does not support a fine-granular quality adaptation like SVC, where it is possible to change anything from resolution to frame rate and frame quality. Therefore, we believe that SVC is more suitable to support the heterogeneity of Internet devices.

2.4 QUALITY OF EXPERIENCE

It has long been a quest to find coding techniques that provide high compression ratios but still offer good video quality. Judging compression ratios and measuring them is quite straightforward when compared to measuring the *video quality*. The major challenge in this context is that humans, whose visual system is very complex, will be watching the video eventually. Evidently, what makes things more complicated is that video compression affects various video content differently. The artifacts due to compression on a slow motion sequence are usually more difficult to detect than that those found in high motion sequences. Therefore, there is a need to measure and assess video quality, especially when these videos are being streamed over the Internet.

Quality of Experience (QoE) is a general term used to describe and quantify the quality of the video as experienced by a user. QoE of a video sequence can depend on many factors including - but not limited to - human perception, video content, user display, and user context.

Video quality assessment describes methods and techniques required to measure QoE. QoE assessment, in general, has been always linked to user

studies, where a video sequence is shown to a number of users who are then asked for their opinion. Evidently, much effort has gone into standardization of video quality assessment with focus on how to conduct user studies and common pitfalls in such methods. Such QoE assessment is well-known as *subjective QoE*, where the subjective opinion of the user on the video sequence is of interest. Nonetheless, QoE is not limited to subjective assessment, as we present later, other approaches based on an automated derivation of quality aspects are possible. This class of methods is known as *objective QoE*.

Next, we elaborate on both subjective and objective QoE.

2.4.1 *Subjective Metrics*

The area of *subjective QoE* is quite large including aspects from many fields of study. In this section, we just give a brief overview about this topic. As explained above, subjective QoE is based on showing human subjects some video sequences and ask them for their opinion.

Probably the most challenging of all in the field of user studies is: how can one isolate the effects of many factors when assessing the video quality. For example, how can we make sure that a person dislikes the video due to its low *video* quality rather than him or her not liking the content?

Winkler [100, p. 48-50] has stated that factors affecting user experience can even be related to user interests and expectations, quality of the screen, lighting conditions, screen color properties, and sound characteristics. Any of these factors can have an impact on the outcome of the user study. Therefore, usually user studies involve multiple human subjects to rule out any individual effects. The results of user studies are calculated in the form of *Mean Opinion Score* (MOS).

In order to make subjective studies reproducible and consistent across different experiments, several recommendations for the process of video assessment have been proposed. These recommendations can be found in the ITU recommendations document BT.500-11 [42] and P.910 [41].

The main limitation of subjective QoE studies is that conducting such experiments is very time consuming and expensive. Although some tens of human subjects are enough to get meaningful results in subjective QoE [68], setting up the experiments is very resource consuming.

2.4.2 *Objective Metrics*

Subjective video assessment methods are probably the best methods currently available to measure QoE, but they are not enough. One major issue with this method is that the results hold only for the video content that was shown to the subjects. Nowadays, millions of videos are being streamed every day, with thousands of videos being added regularly. So more flexible methods, which can be automated and executed quickly are needed. This is possible using objective QoE assessment methods.

Objective QoE methods aim at deriving quality ratings in an automated manner without the need for real human subjects. Since the actual development of

such objective methods involves rather intensive subjective tests, these methods already hold much knowledge collected throughout many user studies [68].

Although objective methods are not considered able to exactly match with subjective methods [100, p.70], they do offer new opportunities and application areas especially for Internet-based streaming applications.

Next, we present the most prominent objective QoE assessment methods.

Pixel-based Metrics

Probably the most simple method, pixel-based metrics evaluate the quality of a video by comparing it with a reference at the level of pixels. Well-known methods include the *Mean Squared Error* (MSE) and the *Peak Signal-to-Noise Ratio* (PSNR). Since these methods are quite simple, they have long been used in the research community especially that nothing better was available. Nonetheless, many researchers agree that such methods suffer from many drawbacks. For instance, all pixels within the picture are treated equally, which does not fit the human visual system. Humans tend to focus on certain areas in the video [101]. Recent studies have shown that such methods do not correlate well with the human visual system [82].

Perception-based Metrics

Researchers knew that pixel-based metrics are limited as they cannot capture and compare higher-level information and details within the video. Therefore, better metrics have been proposed. Here we mention the *Structural Similarity* (SSIM) index [98], which builds upon the idea that the human visual system works by extracting structures from an image. The SSIM index works by extracting structures from frames within the video sequence and comparing them to those extracted from the reference video. The more the structures of the sequence under test are similar to those of the reference; the better is the video quality.

Reference versus Non-Reference Metrics

Objective QoE metrics can be classified into two categories depending on their use of a reference video sequence. These are: no-reference, reduced-reference, and full-reference [101, 29].

No-reference metrics only use the received video sequence to estimate the video quality. Reduced-reference methods use additionally some characteristics and features extracted from the original video sequence. Full-reference metrics require the complete original video sequence so that it can be compared with the received sequence. The three different approaches would fit in specific scenarios and applications. Since full-reference metrics require the availability of the original video sequence, their use has been limited to video quality estimations in the lab. Therefore, reduced-reference and even no-reference approaches are used when the video quality has to be estimated in a running application.

Video Quality Metric (VQM)

The *Video Quality Metric* (VQM) [77] is a reduced-reference objective QoE metric that was proposed by the *National Telecommunication and Information Administration* (NTIA). The VQM is considered the state of the art in the area of objective metrics since it has shown excellent properties by independent evaluation [95]. VQM is especially interesting for Internet applications as it can be used for various applications with videos of different resolutions and properties [102]. The superiority in performance as well as its flexibility make the VQM metric a better choice for a growing number of streaming applications [22]. The good performance reported in [95] led the eventual standardization of the VQM metric.

Nonetheless, the VQM metric still entails some costs, namely its high bandwidth requirements for the required reference. According to [29], the reduced-reference of VQM can be as large as 14% of the uncompressed video. Therefore, using such a metric, it is not realistic to send the reduced-reference and process the video quality at the edge.

In this thesis, we focus on using the VQM metric in a P2P VoD system. The reference is not sent to the devices; rather it is merely used at the servers to evaluate the video quality of different SVC layers. The VQM ratings are calculated offline before the video is released. Therefore, no additional overhead is incurred on the devices. The VQM metric helps in generating quality ratings that would guide the peers in making a better choice on the SVC layer to choose. Our approach and how we use the VQM metric are explained in Subsection 5.3.7 of Chapter 5.

2.5 SUMMARY

This chapter has provided the background information required to understand the contents of this thesis. We have first covered the basics of multimedia delivery highlighting the differences between live and on-demand streaming. Additionally, general aspects of P2P content distribution were presented. This brought us to discussions on P2P streaming mechanisms with focus on streaming topologies and scheduling algorithms. Subsequently, we have seen that video coding plays a major role in such streaming systems. Using advanced video codecs, adaptation mechanisms can be built. Prominent approaches were presented, including the SVC and MDC video codecs. The possibility to change the video quality brings new challenges, namely the criteria used to choose the *suitable* video quality. In this thesis, we present approaches that use the estimated QoE to make a better choice on the video quality. This chapter has discussed prominent approaches for estimating the QoE with discussion on both subjective and objective methods.

In the next chapter, we present research work related to ours, classified according to different areas, while highlighting the main differences to our system.

3 RELATED WORK

This thesis aims at improving the video quality and playback performance of Peer-to-Peer (P2P) Video-on-Demand (VoD) streaming systems by supporting the heterogeneity of Internet devices through Scalable Video Coding (SVC). Therefore, research work related to ours stretches out to various fields. For clarity, we classify related work into four major fields: adaptive P2P VoD, P2P video streaming and SVC, media-aware networking, and objective Quality of Experience (QoE) in video streaming. Although each of these individual areas has received extensive research efforts in the past decade, the research community has barely scratched on the benefits while combining these approaches. In this chapter, we present related work classified by the above-mentioned areas while trying to focus on research that comes closest to ours.

3.1 ADAPTIVE P2P VIDEO-ON-DEMAND

In general, the research community has put substantial effort into investigating adaptive P2P VoD [60]. This includes different aspects such as theoretical models [53], replication techniques [20], prefetching policies [40], network awareness [38], and the impact of server allocation [79]. It was early recognized that video coding techniques are crucial for a high streaming experience [11, 21].

Streaming video content poses special and usually strict requirements on data transmission. Video files are usually large and require much more bandwidth than audio files. As long as resources are guaranteed, data transmission would proceed as expected. However, the Internet rarely offers guarantees, thus performance is drastically affected upon resource scarcity. Classical approaches [60, 32] started of by being agnostic to the video codec, and researches would resort to standard video codecs that exhibit the highest compression ratios [60]. These codec-agnostic approaches are also known as single layer video systems. Evidently, different parts for the video stream have different impacts on the video if they were to go missing. For example, key video frames are required to decode other prediction video frames. Losing the key frames means that data depending on it cannot be decoded.

Adaptation was a main drive for much research in the area of P2P streaming. But before having the availability of advanced multi-layer codecs, researchers had to resort to simply extracting information about the importance of different video blocks from the video itself. For example, in [32], Fortuna et al. use the properties of single-layer video codecs to realize a pull-, mesh-based P2P streaming system. This work applies a special piece scheduling algorithm that uses priorities derived from the video blocks and the frames they contain. Therefore, if a block is carrying prediction frames and resources are not sufficient to send it, then the system scales and the additional frames are simply dropped. Although this is considered a basic kind of scalability, it was,

nonetheless, an important step forward to more advanced adaptation techniques.

As outlined above, it is essential in scenarios with resource heterogeneity that multi-layer video coding techniques are used. This allows operating in the presence of devices with varying characteristics, from desktop computers to handhelds [26]. Furthermore, quality can be switched during playback to adapt to changing network conditions and system load.

Early research on quality adaptation started by considering general multi-layer codecs with focus on single dimension scalability [48, 62, 71, 58, 75, 81]. Therefore, these systems did make a clear distinction between temporal, spatial, and SNR scalability. Rejaie et al. present PALS [81], a receiver driven P2P video streaming system with quality-adaptive playback of layered video. Since PALS and the other mentioned systems only considers single dimensional scalability they cannot adapt to heterogeneous characteristics of peers with different degrees of freedom. The other issue with these systems is that no specific coding standard was used and the authors had to assume a certain bit rate for the individual layers. We focus on using the original SVC implementation to extract bit rates of real SVC videos so that more realistic models can be evaluated.

3.2 P2P VIDEO STREAMING AND SVC

Motivated by the above outlined issues, the use of multi-layer codecs, such SVC has received extensive investigations. It was evident for the research community that SVC provided more information about the structure of the video structure for the streaming applications. Video blocks could then be easily mapped to key or prediction frames and the implications of these blocks going missing could be better foreseen. Thereby, it was possible to assign priorities to SVC video blocks based on, e.g. the level of distortion that would be introduced if they go missing.

Live Streaming and SVC

Some systems address various challenges when combining SVC and P2P live streaming that are related to a general streaming system. Here we name [13, 55, 70, 27], that handle issues such as insuring low video startup time and providing a smooth and continuous video playback. Baccichet et al. [13] use prioritization of packets and multicast trees to distribute SVC substreams with a bound on the introduced delay. Lee et al. focus in [55] on challenges for segment seeding and scheduling while deploying a live P2P streaming system that uses SVC. In [71], Nguyen et al. present and analyze a streaming system designed to incorporate network coding and SVC to facilitate deployment of adaptation techniques in streaming systems. In addition to being focused on live video streaming, a distinction between these pieces of work and ours is that they do not consider the impact of using QoE metrics in their scheduling algorithms.

P2P VoD and SVC

Regarding P2P VoD systems that make use of SVC, several architectures have been recently proposed. For instance, [70] proposes and evaluates a system that aims at achieving quality adaptation and a smooth playback. However, the authors do not investigate a real P2P system, but rather focus on a limited local view (simple download of content from several peers). Mokhtarian et al. present in [66] an analysis of P2P VoD systems with scalable video streams. They provide analytical models that estimate the number of peers that can be admitted into the system in case of flash crowds. Their results can be integrated into our analysis to better match server and peer resources.

Proceeding within the area of how to combine SVC and P2P VoD streaming, Cui et al. [26] presented a P2P VoD system that uses layered video coding to address heterogeneity of the resources of peers in such systems. Furthermore, Mokhtarian et al. [66] developed a model that estimates the maximum system capacity of an SVC-based P2P streaming in case the peers are requesting and retrieving a certain minimum video quality. In Section 6.2, we provide a similar model that also considers that peers will request higher qualities and not only the base layer.

Further pieces of work [103, 74] focused rather on the scheduling algorithms trying to answer the questions such as when should a piece be transmitted and how much resources it should get. In general, these works focus on metrics that are relevant for the users and evaluate how to improve the performance of the scheduling algorithms. We re-use some of the presented metrics and extend them with others that reflect the QoE. This is achieved by objective metrics that can be evaluated directly at the peers without any additional interaction with any other entity in the system. Another approach introduced by Oechsner et. al [74] is similar to ours. However, the authors investigate only temporal scalability and evaluate the average played layers and stalling times. This differs from our approach, since we additionally investigate the tradeoff between SVC video quality and session quality.

In general, SVC allows adaptation to be performed at the receiving peers by simply requesting parts of the stream that match their resources. Nonetheless, the existence of more powerful peers enables approaches where those peers can actively re-encode the video file to match resources of the receiving peers as done in [44].

In contrast to the mentioned pieces of work, we present our P2P VoD system with full support for quality adaptation using SVC. Therefore, we use three-dimensional scalability as defined by the H.264/SVC standard [83] to adapt to different peer resources and network conditions. Although the problem of P2P VoD has been well investigated by the research community [60], the impact of quality adaptation while using a full-fledged SVC design has not yet been assessed.

3.3 MEDIA-AWARE NETWORKING

Research on media awareness has been an active topic in the research community for some time [36, 30]. The aim is to find efficient methods to achieve a smooth and high quality playback by making routing elements perform media-aware actions. Approaches related to ours can be broadly summarized as solutions that utilize information on the importance of different media parts to either enhance the quality or limit video distortion. This utilization can be done either at the edge [47, 36] or at the core of the network [30]. Solutions at the edge, on one hand, usually implement media awareness through overlay routing and media-aware scheduling. Further, there exist many solutions that fall into the distortion-aware packet discard category [36, 108]. Upon congestion, such methods would drop media packets according to the distortion that would be inflicted on the video quality. Solutions in the core, on the other hand, try to utilize Quality of Service (QoS) management capabilities available at network routing elements. In [30], Fidler shows how Differentiated Services (DiffServ) [73] can be used to improve system performance when using layered video coding. Our solution differs since we systematically assess the priority of SVC streams and show that this priority consists of temporal and quality aspects. We further take a new approach, namely using router virtualization, to implement media awareness. We show that a simple media-aware network solution for SVC-based streaming systems can greatly enhance the performance of the system. Our system is a first attempt, within this area, at assessing the impact of media awareness in P2P VoD streaming.

3.4 OBJECTIVE QOE IN VIDEO STREAMING

The last area of research related to ours goes in the direction of using QoE in the decision making process of video streaming applications. To the best of our knowledge, our system is the first to use SVC QoE information in a P2P VoD system. Evidently, none of the considered systems listed here actually use P2P. Subsequently, the challenges that those pieces of work address are rather different from ours. Nonetheless, we present some prominent examples that are related. This section is divided into two parts: objective QoE assessment of SVC videos and QoE metrics for quality adaptation.

Objective QoE Assessment of SVC Videos

The first part is related to video quality assessment and quantification of QoE. QoE is typically a subjective measure of a certain system. QoE is quite complex to assess as it is affected by various physical and psychological factors. The way users experience a certain streaming application can be influenced by the quality of the video itself, or how the users perceive it.

Probably the most prominent work that tries to assess the impact of the scalability of SVC video on user perceived quality was presented by Zinner et al. [113]. There, the impact of temporal and spatial adaptation on the perceived video quality was examined. The authors, similar to us, used the flexible ob-

jective QoE metrics to map SVC layers to perceived quality. This work makes use of the state of the art in objective QoE metrics, namely the Video Quality Metric (VQM). In this thesis, we use the derived results and combine them with our quality adaptive P2P VoD system. The work provides quality ratings, which we use when deciding the SVC layer to choose, as the amount of choices can be rather high. Therefore, it is essential to take a decision that has the side effect of improving the video quality.

Another work that evaluates the relation between SVC quality layer and perceived quality was done by Lee et al. [54]. The main result of that work is that content type and video bit rate have the highest impact on the perceived video quality. Lee et al. come to the same conclusion as [113] concerning the high impact of temporal adaptation on the perceived quality. They additionally extend this conclusion to low bit rate video to state that for such videos, the spatial adaptation has the highest impact on the video quality.

In addition to measuring the quality of different SVC layers, the impact of switching layers is also an important factor. Work done by Zink et al. in [111] aims at characterizing the impact of layer change frequency and the amplitude of these changes on the user perceived quality using subjective tests. The main result of that study is that the number of layer changes and amplitude should be kept as small as possible. Nonetheless, the authors do not provide concrete numbers as what would be good values for these factors. Therefore, we present the performance of our system in metrics that reflect these factors.

Assessing video streaming quality is not limited to the actual video quality but may extend to assessing the impact of startup and stalling times as done in [37]. Although we do present metrics that quantify the startup time and stalling time, we do not map this to perceived user quality as this highly depends on the context [72]. We, therefore, stop at these metrics as our system may be used in various commercial and non-commercial applications where the impact of these factors may vary. For example, if a service is paid, then the startup requirements may be much stricter than for a free service.

QoE Metrics for Quality Adaptation

The second part is related to using QoE metrics for quality adaptation. In the work done in [112], the authors propose a framework for QoE management for SVC video streaming systems. Motivated by their previous work on assessing SVC video quality [113], Zinner et al. assess the impact of changing the video resolution using SVC while having different frame up scaling methods. They investigate having different levels of network congestion and video types and measure the VQM ratings of the different algorithms. The authors come to the main conclusion that to achieve better quality, the system is better off reducing the video resolution to avoid more drastic distortions due to lost packets.

Furthermore, Zhai et al. [105] propose a client/server SVC-based video streaming system with user devices having heterogeneous clients. The authors aim at improving the perceived video quality in a wireless setting. Zhai et al. rely on non-reference objective QoE metrics along with quality adaptation methods in the context of live video streaming. The system works by continu-

ously measuring the quality as perceived by the user, which is calculated from various parameters of the video stream. Based on these estimations, the system would decide on switching to another layer to maintain a certain quality level. The main distinction to our work is that we consider a VoD scenario. The subtle difference in this case is that we can more rely on pre-computed quality ratings for the different video qualities and do not have to calculate this in real time at the peers. Additionally, we focus on a reliable transport protocol, which removes the problem of missing packets and possible artifacts in the video stream.

Another client/server-based approach was presented by Kim et al. [51]. The main distinction of this work is that the server has the task of deciding on the quality level suitable for all the users. In our approach, the decision making process is distributed and up to the individual peers.

Last, Menkovski et al. [65] presented an adaptation approach for P2P streaming systems that aims at achieving acceptable video quality for the users. Mekoviski determine acceptable video quality by continuously asking the users for feedback. This makes such a system hard to deploy in real life, as users do not like to be interrupted so often. This feedback is used along with other non-QoE metrics to detect and avoid non-acceptable user experience. The main difference to our work is that we use multi-layer video coding to allow for on-the-fly adaptation even without user feedback. Nonetheless, the mentioned work could be integrated into ours while expanding the granularity of quality assessment from binary (acceptable, non-acceptable) to more fine granular parameters.

3.5 SUMMARY

In this chapter, we have presented the major pieces of work related to ours, spanning multiple areas of research, namely: adaptive P2P VoD, P2P video streaming and SVC, media-aware networking, and objective QoE in video streaming. Although much work has been done in the individual research areas, the combination of all of them has not been thoroughly investigated. To the best of our knowledge, our work is the first that combines these areas and builds the required quality adaptation mechanisms. When evaluating our system, we will select prominent approaches and compare them against our system.

In the next chapter, we provide the basic P2P VoD system design upon which our quality adaptation mechanisms are built.

4

BASIC SYSTEM DESIGN

In this chapter, we present our basic system design upon which we build the quality adaptation algorithms. First, we present the Peer-to-Peer (P2P) Video-on-Demand (VoD) system where we detail its architecture and the different modules and entities. Then, we present the prefetching and upload strategies we have developed in order to improve the efficiency of the VoD system.

In this thesis, we assume that users using the system are collaborating. This means that a user does not intentionally prevent the VoD software from uploading video data to other peers. This can be enforced either using closed source software or by providing incentives to the users. The issue of incentives is a well-known problem in P2P systems and many solutions have been proposed [49, 10, 56, 67, 40]. Our system can be used either as closed source software or along with an incentive mechanism. This, nonetheless, goes beyond the scope of this thesis, since our focus is on quality adaptation in P2P VoD.

4.1 P2P VOD SYSTEM

We now present the P2P Video-on-Demand (VoD) system highlighting its architecture, system modules and entities, as well as peer and block management algorithms.

4.1.1 *System Architecture*

The architecture of our P2P VoD system is depicted in Figure 11. Here we make use of a P2P architecture in which servers and peers work together to deliver the video content. Such architectures have shown to have the highest applicability in realistic scenarios [40].

We design the P2P VoD system from scratch so that we avoid having a performance penalty that would happen when using, for example, BitTorrent and its block selection strategies. Therefore, we are able to focus on the quality adaptation mechanisms

The components involved in the P2P VoD system are the tracker, content delivery servers, and the peers. We assume that any two entities in the system can communicate directly. All entities communicate using the Transmission Control Protocol (TCP), similar to renowned VoD systems [7]. Therefore, we do not have to handle connection establishment and packet loss. The topology is based on a mesh, where we do not explicitly define the application-layer connections between the different entities (See Subsection 2.2.1). These connections will be driven by offer and demand and do not have to be managed, simplifying the design of the streaming topology.

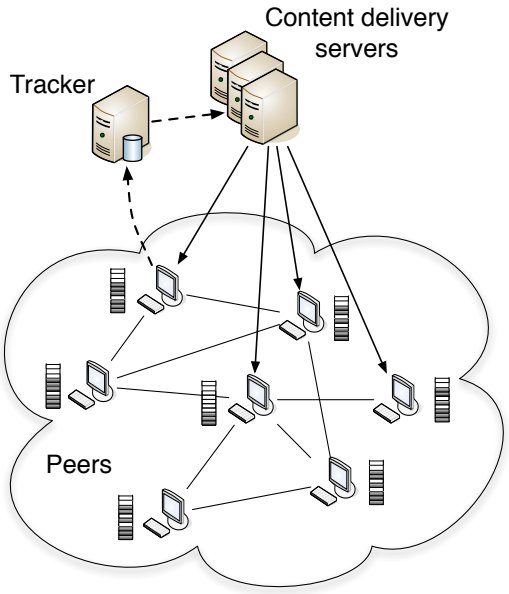


Figure 11: Basic P2P VoD architecture.

The tracker has the basic task of keeping track of all online peers, the video currently being watched, and other information such as the video quality. The streaming servers also inject the initial content and help to maintain QoS in case that there are not enough peers to provide the video data.

A peer can be in two states, either downloader (leecher) or uploader (seeder). The term downloader or leecher denotes the state when the main interest of the peer is to download video data for playback. A downloader peer also uploads data to other peers; however, it usually cannot provide video pieces further beyond its playback position since they are not available. The uploader or seeder state, on the other hand, is reached once the peer has finished download and is still uploading data to other peers. Streaming servers can also be regarded as seeders with the distinction that they are more reliable and can provide more resources.

We start by explaining the tracker design and then we go into the details of the mechanisms running at the peers.

4.1.2 *Tracker Design*

The tracker keeps track, maintains, and manages a list of all peers active in the VoD system. Its main role is to reply to requests asking for lists of peers streaming the same content. Thereby, it enables the peers to connect to each other and to perform the actual video data transfer.

For each peer, the tracker keeps track of the following information: IP address, port number, streamed video quality, and playback state.

The tracker is especially essential when a peer joins the network. The newly connected peer uses the tracker to bootstrap and to discover other peers. The tracker is also occasionally used to get additional peers since provider peers are not always online throughout the streaming session.

The message sent from the peer to the tracker contains a list of connected peers, dropped peers, current selected video quality, and information about the playback state of the peer to keep the information at the tracker fresh. These pieces of information are essential to help the tracker in making a better decision on the returned peer list that considers the various factors.

The information on the currently selected video quality is used in refining the returned peer list. Details concerning the algorithms behind this are explained in Subsection 5.1.2. Since the tracker has information on the playback state of the peers, it makes sure that a server is returned to the peers that are having difficulties in their playback. Thereby, it tries to reduce the time a peer is in a stalling playback state. It is worth noting that the tracker also keeps track of the servers and is aware of their status. This helps in giving the tracker a chance in performing load balancing actions of system resources.

Peer Mechanisms

Figure 12 shows an overview on the set of algorithms and mechanisms running at the peers within the P2P VoD system. These algorithms are broadly categorized as: block management, peer management, and video playback algorithms. These algorithms are explained next.

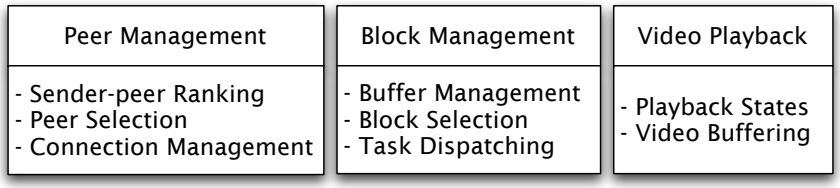


Figure 12: Algorithms and mechanisms running at the peers.

4.1.3 *Peer Management*

The term peer management denotes all the algorithms running at the peer to maintain and manage the connected peers. This includes algorithms for requesting peers from the tracker, connecting to other peers as well disconnecting from them when they are no longer needed. These algorithms are sender-peer ranking, peer selection, and connection management.

The connected peers are divided into two sets: the *receiver-peer* set and the *sender-peer* set. Unless otherwise mentioned, the term *local peer* indicates the peer at which the mentioned algorithms are executed. The receiver-peer set comprises those peers to which the local peer uploads video blocks, whereas the sender-peer set comprises the set of peers from which the local peer downloads video data. The main reason why we separate the connected peers into the two sets is because download needs are not symmetric. For example, if peer A wants to download some video block from peer B, it does not have to be that peer A also wants to have a video block from peer B.

The sender-peer set holds the contact information of the peers, which can potentially provide video blocks to the local peer. This set is filled by periodically querying the tracker for new peers.

The receiver-peer set holds the contact information of the peers to which video blocks are being uploaded. While the sender-peer set does not need to be actively filled, the receiver-peer set is automatically built through the incoming connections from other peers. If the local peer is not uploading to any other peer then this set is empty. Therefore, one main difference between the two sets is, therefore, in the management approach. The sender-peer set has to be actively managed while the receiver-set is built implicitly.

Sender-peer Ranking

The connected peers are not expected to provide video data indefinitely after connecting to them. These peers might fail, get congested, or even leave the system. Therefore, it is important to keep track of how much the peers are contributing to the local peer. This is done using the so-called *sender-peer ranking* algorithms. The main objective of this algorithm is to maintain connections to peers that provide data the fastest while abandoning those peers that do not. Here we define the peer rank as an integer number that reflects how well a sender peer is contributing. We focus on considering peer contribution not due to user decision to contribute or not but rather due video block availability and upload speed.

To decide on whether a sender peer P_{sender} is contributing or not and to further upgrade its rank, the following pieces of information are relevant.

- The number of times the local peer was able to find a required video block at P_{sender}.
- The number of times the local peer was able to download video data from P_{sender}.
- The speed at which P_{sender} was providing the requested video data.

The rank of the sender peer P_{sender} is a positive number initialized with four and is increased by one after a block has been successfully downloaded. The rank will be decreased either when the sender peer rejects a download request, a timeout has occurred, or when a video block necessary for playback is not available at the sender peer.

When using these steps, it often happens that the peer rank reaches zero, that is the minimum possible value. In this case, the peer is abandoned and is not contacted again. If there are not enough peers in the sender peer set, the tracker is eventually contacted for new contacts.

Peer Selection

Keeping track of the dynamic P2P VoD system requires that all peers regularly update their list of possible neighbors in the overlay. This is performed by regularly sending requests to the tracker to get new possible peers to connect

to. To avoid receiving peers that have already been abandoned, the requesting peers ships more information to the tracker to aid it with the selection process. These pieces of information are:

- A filter-list containing the peers that have been abandoned in the more recent peer selection process.
- A white-list containing all the peers that have a positive rank and that are still in the active downloader set.
- The amount of new peers that are needed.
- The playback state of the peer, i.e., whether the video is playing or stalling.

Here it is worth noting that the downloader peer set can be quite dynamic depending on many factors in the system. Therefore, the use of additional peer selection mechanisms at the peer and the tracker are essential in optimizing the topology of the streaming system so that *good* peers are contacted and *bad* peers are dropped.

Connection Management

Matching demand with resources in P2P VoD is not a trivial task. One major challenge is the dynamic nature of the P2P system that could create lots of overhead in the connection management algorithms.

In our system, any two peers in the VoD system are connected over two phases. The first phase, called *pre-connection*, is used to announce to the uploader peers about the potential requesting of video data. The local peer expects an acknowledgment about the readiness of the uploader peers to provide video data.

A pre-connection is important as without it no video data can be requested. Therefore, a peer P_i can download video blocks from peer P_j if and only if the two peers have *pre-connected*. Thus, uploader peers can keep track of the downloader peers that will potentially request video data.

In the next subsection, we take a closer look at the connection management algorithms, namely, those running at the downloader side and at the uploader side.

DOWNLOADER SIDE. When the local peer wants to download video data from a certain peer, first it has to establish a connection with this peer, i.e. pre-connect. Therefore, the local peer sends a connection request to the possible uploader peer. The response to a connection request can be either positive or negative. The positive response means that the other peer has acknowledged the connection and is ready to provide video data. In case the response is negative, this means that the other peer is overloaded and cannot provide any video data. In this case, the peer is kept in the local database as a potential uploader for later connection attempts. Additionally, the rank of the peer is reduced so that too many failed connection attempts will eventually make the local peer abandon this specific peer and search for new ones.

UPLOADER SIDE. The major limitation of the system in terms of bandwidth is eventually due to the limited upload capacity of the peers. Therefore, it is important to limit the number of active upload connections so that the throughput provided per connection is not too small to cause excessive timeouts and playback delays. By limiting the number of active upload connections, it is further possible to provide a guaranteed throughput for each peer and, therefore, make the system more robust. An active upload connection is also called an *upload slot*.

After a connection request has been received, the uploader peer checks whether there is an empty upload slot. If this is the case, then the uploader acknowledges the connection, assigns the upload slot, and serves the request. In case there is no free upload slot, then the uploader peer has to decide on either to refuse the request or to drop an ongoing *less-important* connection.

The importance or priority of a connection is reflected by how fast the requested video block has to arrive before a playback stall would occur at the downloader peer. We, therefore, derive the importance of an upload connection from the priority of the last requested video block.

Every time when a peer requests a video block from some other peer, it includes the requested block number along with the difference between the last buffered block and the current playback position. Since this difference reflects how well this peer is doing in filling its playback buffer, it helps the uploader on making better decisions on whether to reject or acknowledge a download connection. The smaller the difference between the playback position and the requested block, the more important it is for the downloader and, therefore, the higher is the priority of the connection.

This priority comes into play once all the upload slots are occupied and a new connection is being requested. In this case, the current upload connections are sorted based on their priority. The connection with the least priority is gracefully dropped (after completing any ongoing transfers) and the slot is assigned to the more important connection. This design greatly helps in acting quickly to urgent requests and in meeting some of the real-time requirements of the streaming process, therefore, minimizing the chance of experiencing playback stall and delays.

4.1.4 *Block Management*

The second part of algorithms running at the peers is related to block management. These algorithms have the task of keeping track and managing the actual block transfers and their priorities. In general, these algorithms can be divided into three major parts.

1. *Buffer management* algorithms keep track of the video buffer. The video buffer is a sliding window that starts with the playback position and is usually seven seconds long. The buffer maps to the video blocks that are essential for continuous playback.

2. *Block selection* algorithms are responsible for calculating the priority and selecting the blocks that should be downloaded next. Each block should

be assigned a priority so that the uploader can better manage its connections.

3. *Task dispatching* algorithms are needed for requesting the required blocks. The main objective of this part is to keep the uploader peers as busy as possible to maximize the download utilization.

We now elaborate on the three parts of the block management algorithms.

Buffer Management

The buffer is essential, as it makes sure that video blocks coming immediately after the playback position are given extra care during video playback. The video buffer is a sliding window that always starts at the playback position and extends to seven seconds of video data. As the playback position moves forward after the successful playback of video blocks, the buffer is also shifted forward. The video blocks within this buffer are requested with a high priority. Should the buffer become drained, the video player will not find any video data and playback will stop. Playback is only resumed after the whole buffer is full again to avoid too frequent start-stop behavior. The basic structure of the video buffer is illustrated in Figure 13.

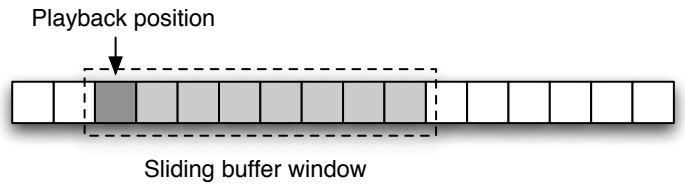


Figure 13: The sliding buffer window.

Block Selection

The block selection algorithms are probably the heart of every P2P streaming system. These algorithms are responsible for deciding on blocks to request and their priorities. We follow a simple approach in which the block with the highest priority is the first to be requested and vice versa.

At any point in time, all video blocks that are to be downloaded are divided into two priority sets: the *high-priority set* and the *low-priority set*. This division is depicted in Figure 14. The point behind this is to reflect the fact that blocks close to the playback position should be treated differently than those at the end of the video file. Therefore, different algorithms are used for the two sets. In general, the algorithms used for the high-priority set act greedy and work on a short-term objective so that to simply keep the buffer full. On the other hand, the algorithms used for the low-priority set try to address the availability of blocks and prefetch certain parts of the video, which are useful for other peers.

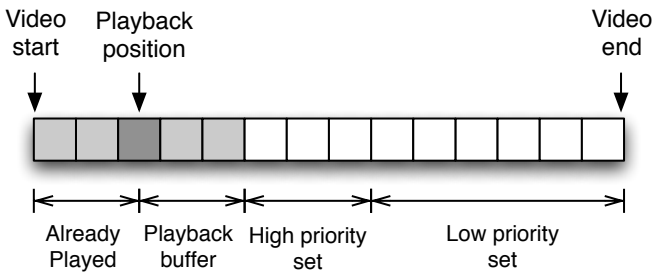


Figure 14: Two priority sets for downloading video blocks.

HIGH-PRIORITY SET. The high-priority set comprises blocks that lie after the playback buffer. The main objective is to keep this set full. Since these blocks are essential for continuous playback, they are downloaded with a high priority. Within the high-priority set, the priority calculation follows a linear approach in which the priority of the block at the beginning of the set gets the highest priority while that at the end gets the lowest priority, similar to the order in which the video player plays the video blocks.

LOW-PRIORITY SET. The low-priority zone starts after the last block of the high-priority set and ends at the end of the video file as shown in Figure 14. The low priority blocks are downloaded only when the high-priority set is full as not to intervene with a continuous video playback. Therefore, if a peer would start downloading low-priority blocks, this would demonstrate that this peer is enjoying good performance. Here, the low-priority selection algorithms come into play to optimize the performance of other peers by prefetching video blocks that they need. Details on this approach are explained in Section 4.2.

Task Dispatching

The sequence diagram that illustrates the dispatch of download requests is depicted in Figure 15. After a peer establishes download connections to other peers, it starts requesting video blocks. However, one important point that has to be considered by the download manager is not to request too many adjacent blocks from the same peer. If this happens and this peer is congested or even fails, the requesting peer will suffer from much stalling events that are difficult to recover from. Therefore, blocks are request in a round robin manner such that all blocks requested from the same peer are as far away from each other as possible[1].

4.1.5 *Video Playback*

The video player, in the classical sense, is the entity that decodes video content and displays it to the users. The video player is usually responsible for

1 Evidently, if all peers in the download-peer set have all required blocks. then the difference between any two blocks requested from the same peer is exactly the total number of peers in the download-peer set.

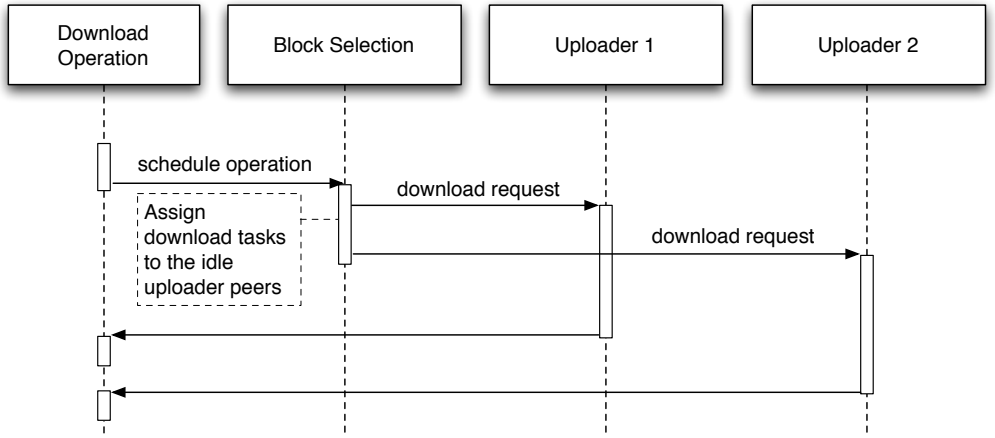


Figure 15: The dispatching of download tasks.

other functionalities that assure continuous playback by communicating various pieces of information with the underlying streaming protocols. Information in the context of VoD includes mainly the playback status. Video playback can be in three different states: idle, buffering, or playing. For simplicity, we assume that the video file is being played sequentially from start till the end.

Playback States

The different states of the video player are depicted in Figure 16. It consists of the following three states:

- *Idle*: indicates that there is no active video streaming taking place or playback has finished.
- *Buffering*: indicates that no playback is taking place due to lack of video data to play. This phase can be reached either after a stalling happens or after streaming is initiated.
- *Playing*: indicates that the player is continuously playing the video stream.

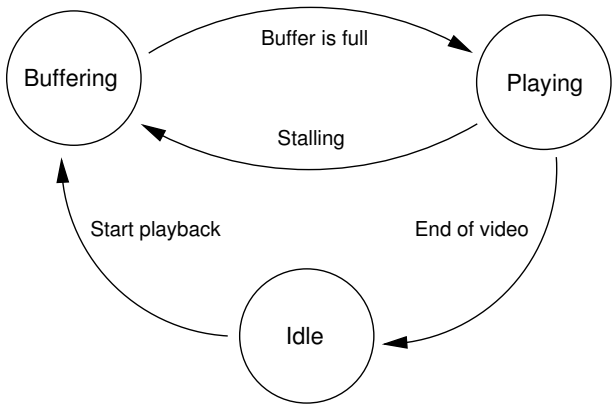


Figure 16: The state diagram of the VoD video player.

After the user initiates playback, typically by pressing play, the player switches from the *idle* to the *buffering* state and the underlying streaming protocols are started so that video data is requested. Once enough data has been downloaded and the buffer is full, the player starts playback and switches state from *buffering* to *playing*. The time elapsed between the two states, i.e., the time difference between when playback was initiated and the actual start is called *initial buffering* or *startup time*. Startup time is unavoidable since it always takes time to fill the buffer with video data while using capacity-limited connections.

After playback starts, the underlying streaming protocols work hard to sustain a continuous stream of data so that the player can always play the next video block at its deadline. If this happens, then the video player remains in the *playing* state. If all data required for playback is available, the playback position is shifted continuously with every round. In case a certain block is missing or not completely downloaded, then the player changes from the *playing* to the *buffering* state. This is known as *stalling*. Finally, when the playback reaches the end of the video file, the player goes back to the *idle* state.

Video Buffering

Video playback and the underlying streaming protocols outline the general basic functionalities of the P2P VoD mechanisms. In general, it is enough if the download throughput is as large as the playback speed or video bit rate. Since it is very difficult to assure such a property, the download buffer is essential to absorb short-term fluctuations in the download throughput. In case stalling does take place, then playback starts only after the buffer is full again so that to avoid too frequent start-stop behavior of the video player.

4.2 PREFETCHING AND UPLOAD STRATEGIES

In this section, we present the prefetching and upload strategies used in our P2P VoD system. Prefetching is a term used to describe the process of downloading video blocks, which are not immediately needed, i.e., within the low-priority set. An upload strategy, on the other hand, denotes the scheduling strategy used by the uploader peers. Both the prefetching and upload strategies play a major role in the performance of the P2P VoD system.

Traditional prefetching strategies in VoD and Bit-torrent-like systems are based on the so-called *rarest-first* strategy [24]. This strategy mainly considers the rarity of a certain block in the local neighborhood. The less a block is available in the local neighborhood, the more it is likely it will be prefetched. This way it is ensured that the video file is distributed as much as possible throughout the system to achieve high download rates and be immune to peer churn. Nevertheless, a fundamental problem of the rarest-first strategy is that it conflicts with the linear streaming requirement [12]. The video player expects the blocks to be available one after the other and not according to the rarity in the neighborhood.

Therefore, we go one step ahead of the rarest-first strategy present the *Soon-Most-Needed* (SMN) strategy that optimizes the block distribution based on playback positions of other peers. Our goal is to improve the performance of the system and use the SMN strategy as an alternative to the rarest-first strategy.

4.2.1 *System Model*

Before we detail our approach, we now present our system model that investigates the tradeoff between server capacity allocation and achievable playback performance. The notations we use in the analysis are as follows:

- S: number of servers
- u_S: server upload capacity
- U: number of uploaders that have the whole file
- D: number of downloaders having an incomplete file
- u, d: upload and download capacity of a peer
- r: video bit rate ($r \leqslant d$)
- f: average peer *prefetching factor*
- g: average peer *upload utilization*
- $d_r = f \cdot r$: required download speed ($r \leqslant d_r \leqslant d$)
- B_i^k: block with index i at peer k
- B_{total}: total number of blocks

U is the number of uploaders that have the whole file, also called seeders. D is the number of downloaders that have none or part of the file. Of course a downloader also uploads blocks. Therefore, the main difference between an uploader and a downloader is that the later might not always be able to fully utilize its upload capacity due to lack of blocks useful for other peers. Thus, we define the upload utilization factor g as the ratio of the average upload speed of a peer to its upload capacity. For uploaders, this value is almost one. Additionally, it is also close to one for downloaders in case of a system that exhibits good block diversity. The prefetching factor f is the ratio of the average download rate to the video bit rate. This parameter represents how fast a peer should download the video file so that playback is sustainable.

The total offered system upload capacity is given by

$$u_{\text{total}} = D \cdot u \cdot g + U \cdot u + S \cdot u_S. \tag{4.1}$$

The total download speed is limited by the system upload capacity, i.e., $d_{total} \leqslant u_{total}$, and the total download speed is given by $d_{total} = D \cdot d_r = D \cdot r \cdot f$. Thus we get:

$$D \leqslant D_{max} = \frac{U \cdot u + S \cdot u_S}{f \cdot r - g \cdot u}, \qquad \text{where} \quad \frac{u}{r} < \frac{f}{g}. \tag{4.2}$$

For a given number of uploaders, Equation 4.2 calculates the upper limit D_{max} of the total number of supported downloaders in relation to the prefetching and upload utilization factors. In addition, it gives rise to one of the trade-offs we address in this part of the system: on one hand, having peers make aggressive prefetching (high f) will lead to high upload utilization since it is more probable that more peers need some prefetched blocks. However, this happens at the cost of using up more system resources. On the other hand, having too small f will lead to low upload utilization. In this thesis, we propose prefetching and upload strategies that, for minimal prefetching factor f, have a sufficiently high value of upload utilization g.

4.2.2 *Soon-Most-Needed Prefetching*

The prefetching strategy proposed in this section aims at striking a balance between achieving in-order block delivery and having better neighborhood block distribution. The prefetching strategy here is intended as a low-priority block selection algorithm as presented in Subsection 4.1.4.

In a classical P2P VoD system, different peers are not aware of the playback positions and the high priority blocks required by neighboring peers and the network. To overcome this problem, we propose the *Soon-Most-Needed (SMN)* prefetching strategy. This strategy optimizes the block distribution in the neighborhood by taking into account the playback positions of neighboring peers. This information is used in such a way that peers actively vote for the blocks they need as explained next.

In Figure 17, we present how voting is performed at a certain peer. Each peer is allowed to vote for m blocks that are still missing from its video file every T seconds, called the refresh interval. We suppose that voting is synchronous, i.e., all votes are received at the same time.

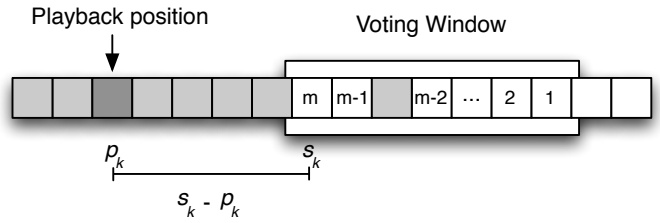


Figure 17: Voting procedure at peer k.

The vote values will be set in such a way to be decreasing with increasing block number. In addition, voting at peer k starts at block s_k, which is the first block after the playback position not received yet. To allow for prioritized allocation, these values are further scaled by the difference between the playback position and the first block being voted. Therefore, peer k will vote for the successive m blocks not received so far according to the equation:

$$Vote(B_i^k) = (m - i + s_k)/(i - p_k) \quad i = s_k ... s_k + m' \tag{4.3}$$

where m' is m with blocks already buffered within the voting window. After each peer has cast its votes into the so-called SMN list, it sends this list to all peers in its neighborhood. The SMN list is sent only to downloaders and not to uploaders, since the uploaders have the whole video file and do not need to prefetch any blocks. The votes are then collected and aggregated by the downloader peer. Aggregation is then simply the adding up all votes for each block, therefore

$$Vote(B_i) = \sum_{k=0}^{|L|} Vote(B_i^k). \tag{4.4}$$

All the downloaders are continuously discovering a list of blocks that are available within their neighborhoods and generating an SMN list. The intersection of both lists is generated. The result is a list of the SMN scores of all blocks that are available at neighboring peers. From this list, all blocks the local peer already has downloaded are removed.

The next step before prefetching can start is taking into consideration local availability of blocks. One block could be well distributed in the local neighborhood while another one with an almost equal SMN score is very rare. In this case, the rare block is selected. To accomplish this, the un-availability factor a is introduced. It is calculated by:

$$a(B_i) = 1 - \frac{Replicas(B_i)}{\text{size of neighborhood}}, \tag{4.5}$$

which is high (close to one) when there are only a few replicas for a certain block and low (close to zero) when there are many replicas. The SMN scores are multiplied with the availability factor and then transferred into download probabilities by normalization as follows:

$$Probability(B_i) = \frac{Vote(B_i) \cdot a(B_i)}{\sum_{j=0}^{B_{total}} Vote(B_j) \cdot a(B_j)}. \tag{4.6}$$

The downloader peer then chooses its prefetching blocks according to these probabilities. If there is no peer in the neighborhood that has any prefetching block, that is when $\sum_{j=0}^{B_{total}} Vote(B_j) \cdot a(B_j) = 0$, rarest-first is then used to build up a strong distribution of blocks.

4.2.3 *Upload Strategies*

After each peer makes a decision on the next blocks to download, it sends a request to its neighboring uploaders. In this case, a certain uploader would continuously receive requests for different blocks and, therefore, has to make a decision on which block(s) to upload. Since the requests have the distinction

of being either high priority or low priority, this distinction should also be considered while making the upload decision. Based on this design, all low priority and high priority requests are put at the uploader in a high and low priority set respectively. Now we present possible upload strategies that the uploader can use for the different sets and how they can be combined.

HIGH PRIORITY SET. To optimize the local connections, an uploader can simply order all high priority requests according to their priority, i.e., their deadline. The closer the deadline of a certain block, the sooner it should be uploaded. In case the deadline of a certain block cannot be met, a quick decline message is sent to allow for fast request retry.

LOW PRIORITY SET. In addition, an uploader is able to optimize the distribution of useful blocks within its neighborhood. Therefore, the uploader can favor low priority blocks according to the presented SMN prefetching strategy. This is important when the system is either in deficit state or on the verge of being overloaded and, therefore, should reorder priorities to have a more robust distribution of soon-most-needed blocks so that the peers can serve high priority blocks in the near future.

UPLOAD BIAS PROBABILITY. Since system resources can be in either surplus or deficit, local and neighborhood-wide optimizations have to be combined and balanced. To do this, we introduce the upload bias probability p used when making an upload decision, which will address the tradeoff between optimizing performance of individual requestors and optimizing the neighborhood. Therefore, a peer will decide on uploading a high priority block with probability p and on uploading a prefetching block with probability $1-p$.

System Load Adaptation

In a P2P VoD system, and for a given server capacity, performance is dependent on the number of downloaders, which degrades drastically near the limit $D_{max} = \frac{U \cdot u + S \cdot u_S}{(f \cdot r - g \cdot u)}$ as presented in Equation 4.2. One interesting question that arises is: how to minimize performance degradation during flash crowds, and how to prepare the system for this situation?

Our approach works as follows: the parameter p is the key to the balance between high priority and soon-most-needed blocks. Therefore, we assign the value p depending on the ratio of demand posed by the downloaders to the supply provided by them. The higher this ratio, the less capable is the server to serve high priority blocks. Thus, the uploaders and downloaders themselves should be more ready to serve such blocks. On the other hand, when the demand is much less than the supply, then the server is able to serve any required high priority request, and thus downloaders should focus on building strong distribution of soon-most-needed blocks in preparation for a

flash crowd. Therefore, and based on the system model presented in Subsection 4.2.1, we assign

$$p = \frac{\text{demand by downloaders}}{\text{supply from downloaders}} = \frac{D \cdot r \cdot f - U \cdot u - S \cdot u_S}{D \cdot u \cdot g}. \tag{4.7}$$

It is worth noting that the above equation should saturate to one at the system limit of supported peers (D_{max}), after which only high priority blocks are served by the whole network, and the system is almost in deficit. However, when $p < 1$, the system elegantly adapts to the number of downloaders while taking resources and system capacity into account. All required information to calculate a new value of p is directly available at the tracker. For simplicity, the tracker can use the average values of g and f. Finally, the tracker can send p piggy-backed within other status messages that are regularly sent to the peers without inflicting any additional overhead.

This concludes our prefetching and upload strategies suggested in this thesis. These strategies are not part of our major contributions. Therefore, they are not the focus of our evaluation. Nonetheless, we provide some evaluation results in Appendix A.3.

4.3 SUMMARY

This chapter has presented our basic system design, detailing the architecture and the different entities in the P2P VoD system. The tracker, peers, and streaming servers employ various algorithms and strategies in order to maintain the VoD system. We have additionally presented the used prefetching and upload strategies that aim at improving the system's playback performance.

This chapter described only the basic system upon which the quality adaptation mechanisms and algorithms can be built. Next, we go into more details and present the core contributions of this thesis.

5 QUALITY ADAPTATION IN P2P VIDEO-ON-DEMAND

The combination of Scalable Video Coding (SVC) and Peer-to-Peer (P2P) Video-on-Demand (VoD) gives great benefits but many challenges have to be addressed. This chapter describes the major contributions of this thesis in the context of *quality adaptation* for P2P VoD system.

Quality adaptation is achieved over three steps, as explained in this chapter. In Section 5.1, we present the SVC quality adaptation mechanisms executed at the peers residing at the edge of the network. In Section 5.2, we present how media awareness of routing elements can be used to perform quality adaptation in the core of the network. Finally, in Section 5.3, we present how using Quality of Experience (QoE) metrics in the quality adaptation algorithm can make the VoD system more efficient.

5.1 QUALITY ADAPTATION USING SVC

We now present the quality adaptation algorithms developed in order to integrate SVC in the basic P2P VoD system presented in Chapter 4. The quality adaptation mechanisms are developed as follows: first, we analyze the SVC video data structure and show how it can be used to enable efficient P2P piece management. Then, we highlight the SVC-based peer selection algorithms running at the tracker and the peers. After that, block selection algorithms required to handle SVC videos are presented. This brings us to the two-stage adaptation loop running at every peer in the VoD system. This adaptation loop makes sure that the peers adapt the video quality to both static and dynamic resources, thereby, addressing the problem of heterogeneity of the Internet.

5.1.1 *Linearization of the SVC Cube*

A video file is divided into many pieces each usually worth couple of seconds of video playback. As elaborated in Section 2.3, video files encoded using the SVC standard are organized in a three-dimensional cube. Therefore, each piece contains all possible layer combinations and is divided into smaller sub-cubes as shown in Figure 18. The more layers are available in any dimension, the higher is the played quality of this video piece. The SVC sub-cubes are further divided into P2P transmission blocks. A block will be the smallest entity for video discovery and transmission.

We have developed a specific data structure to keep the block selection algorithms as simple as possible. The actual representation of the video file is done using a single-dimensional array made up of all the blocks in the whole SVC file. To achieve this, we have to linearize the SVC video model as follows. We traverse the three dimensions of the SVC cube in well defined order and store the sub-cubes one after the other in a single dimensional array as shown

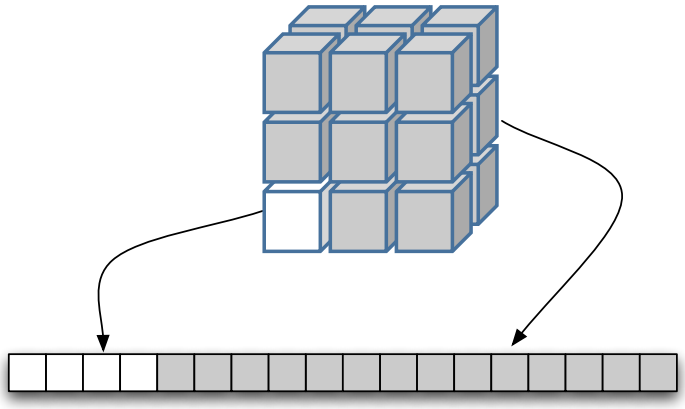


Figure 18: The linearization of the SVC cube-model for one video piece.

in Figure 18. Since the different SVC layers or sub-cubes have different sizes, the number of video blocks required to store each sub-cube is different.

Suppose we have an SVC video file with two spatial, one temporal, and three quality layers, then in total there are $2 \times 1 \times 3 = 6$ video qualities or layers. Each video quality can be described as a triplet:

$$(d, t, q) \qquad d \in \{0, 1\}, t \in \{0\}, q \in \{0, 1, 2\} \tag{5.1}$$

where d, t, q represent the quality level in the spatial, temporal, and quality dimensions respectively.

If we traverse the video cube first in the quality, then in the temporal, and at last in the spatial dimensions, then the linearization process traverses the SVC cube following the block path:

$$(0,0,0) \to (0,0,1) \to (0,0,2) \to (1,0,0) \to (1,0,1) \to (1,0,2). \tag{5.2}$$

The same operation is repeated for all pieces within the SVC video file while concatenating all the blocks together into a single-dimensional array. This design, as shown in Figure 19, allows for more efficient implementation for the access and management of the SVC video blocks.

Since the SVC video structure is layered, decoding a certain layer requires that all layers below it are available. Therefore, we need to directly map from SVC layers to their respective video blocks efficiently. To do this, we define two lists for the whole SVC file. One list stores where a layer starts and another list stores the number of blocks within this layer.

To keep track of which blocks belong to which layer when using the linearized array, we perform mapping as follows. Assuming that each video piece

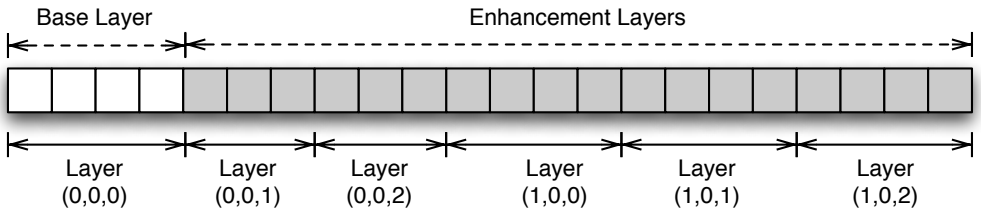


Figure 19: The linearized SVC cube model.

contains n-blocks, each SVC layer is constructed from a certain number of blocks out of the n blocks. We keep a Boolean array per SVC layer of length n that indicates whether each block is part of this SVC layer or not. There is a different Boolean array for each layer in the SVC video file. In order to find out which blocks belong to a number of layers, we perform a simple OR function over their respective Boolean arrays.

5.1.2 *SVC-aware Tracker*

The tracker design outlined in Subsection 4.1.2, does not explicitly consider the actual quality the different peers are streaming. In this section, we describe the additional algorithms integrated into the basic design that takes SVC into consideration.

During peer discovery and selection, the selected video quality of each peer must be taken into consideration. Since each peer in the VoD system has its own SVC layers, not all peers registered at the tracker can support the same video quality. Therefore, more information is needed at the tracker in order to match the layer offer and demand. Thereby, the *initial selected video quality* of the peer is included into the *peer-discovery request*. The tracker can then return only those peers that can support the given video quality. The tracker further stores this video quality in its local database for further peer discovery requests. Later on, each peer would announce its current layer with each keep-alive message sent to the tracker to keep the information there as fresh as possible.

The idea behind the modified peer selection algorithm is to have neighborhood peers whose layer is possibly equal to or higher than that of the requesting peer. Therefore, it is more probable that any of the contacted peers can potentially provide any block needed by the downloading peer. If the number of returned peers is too small, which might happen for peers streaming very high quality, and if the requesting peer does not yet have a server in its peer set, the tracker returns one to it so that its playback is sustained.

5.1.3 *SVC-based Block Selection*

Block selection, at the peers, has the main role of assigning each block a certain priority as explained in Subsection 4.1.4. In addition, it is sometimes required to skip some blocks to allow for continuous playback. We follow the idea of dividing the remaining video data into two sets according to the priority: high-priority video blocks and low-priority video blocks [60]. The partition of these two priority sets is similar to that of a general video streaming system (namely using the buffering window as high-priority zone and the rest of the video as low-priority zone as explained in Subsection 4.1.4).

When using SVC as a video codec, we consider additional parameters when calculating the download priority of a block. Specifically, we consider not only its position in the temporal domain, but also its quality level in the SVC cube.

High-Priority Zone

The download priority of each block in the high-priority zone is determined by considering two factors: the distance to the current playback position and its quality level in the SVC model. These two priorities are denoted by the temporal priority P_T and quality priority P_Q.

The temporal priority generally expresses how soon a certain video block is needed. Video blocks, which are closer to the current playback position, have a higher temporal priority than others. Suppose a certain peer, with the playback position B_{play} is requesting a block with index B_i. The temporal priority of this block depends on its distance to the playback position $(B_i - B_{play})$. This value is, therefore, higher when the requested block is closer to the playback position. We normalize this value by the size of the high priority set HP_{size} to get the temporal priority

$$P_T = 1 - (B_i - B_{play})/HP_{size}. \tag{5.3}$$

Based on Equation 5.3, the block found immediately at the playback position gets the highest priority of one; while those blocks at the end of the high priority set have the lowest priority of zero.

The quality priority, on the other hand, reflects the importance of the SVC enhancement layers. Using it, lower layers for the video stream are given higher priority. For example, without the base layer, the video cannot be decoded. Therefore, the base layer is given the highest priority. The quality priority is calculated as:

$$P_Q = 1 - \frac{(W_d d + W_t t + W_q q)}{W_d d_{total} + W_t t_{total} + W_q q_{total}} \tag{5.4}$$

where d, t, and q denote the spatial, temporal and quality layers respectively, and d_{total}, t_{total}, and q_{total} denote the total number of spatial, temporal and quality layers respectively (see Section 5.1). W_d, W_t, and W_q are used to weight the different scalability dimensions, with $W_d + W_t + W_q = 1$. This equation leads to a decrease in the priority with an increased quality level in any dimension. Therefore, it gives the base layer with $(d, t, q) = (0, 0, 0)$ the highest priority one, while the highest layer gets the lowest priority zero. Using the coefficients W_t, W_d and W_q, the speed of the priority decrease rate in any dimension of the SVC model can be controlled. For example, if we want to prioritize the temporal dimension, we would choose a higher value for the temporal weight W_t. In this thesis, we rather focus on an unbiased weighting in which W_t, W_d and W_q are equal. An analysis of the impact of these parameters is not the focus of this thesis.

Finally, by combining the temporal priority P_T (Equation 5.3) and SVC priority P_Q (Equation 5.4), a block B_i is assigned a final priority as follows:

$$\text{Priority}(B_i) = AP_T + BP_Q. \tag{5.5}$$

In Equation 5.5, the temporal and SVC priorities are added together with the weights A and B so that a balance between temporal urgency and SVC quality is ensured. Therefore, if we are interested in fetching more base layers, then A is chosen higher than B. If we are interested in fetching more enhancement layers, then B is chosen higher than an A. We again choose an unbiased prioritization and choose A and B to be equal.

Low-Priority Zone

For the low-priority zone, we use an algorithm that favors prefetching blocks that will soon be needed by neighboring peers. Prefetching is started once the high priority set is full. Therefore, only peers with excess resources would actually perform the following strategies. Full details about the designed prefetching algorithms were presented in Section 4.2, we now give an overview for completeness.

The peer sends a request to all the peers in its upload neighborhood querying for votes on their most wanted blocks. On receiving such a request, each peer places its votes starting from the first non-received block. Those votes are decreasing with increasing block number. The peer then sums up the votes for each block and then sorts those blocks according to their vote values, i.e., importance. The block with the highest vote, which is not yet available at the peer, will be prefetched and made available to the neighboring peers. The peer filters out those blocks that do not belong to its initially selected video quality. The last step ensures that the selected blocks for prefetching should also be possibly playable by the peer itself, so that unnecessary downloads can be avoided.

5.1.4 *The Quality Adaptation Loop*

In this section, we present our SVC-based quality adaptation algorithms. We first start by detailing the quality adaptation loop running at every peer in the network. Then we elaborate on the main building blocks of this loop.

Figure 20 depicts the basic architecture of the quality adaptation loop.

Quality adaptation is achieved by adjusting quality according to the different peer resources and network dynamics. It is performed by two modules: the *Initial Quality Adaptation* (IQA) and the *Progressive Quality Adaptation* (PQA). Both modules form the algorithms that match the layers with resources available at the peer. On the one hand, the IQA is used for determining the highest possible layer that a peer can retrieve and play, and is performed at session start. On the other hand, the PQA is performed periodically to adjust the layer according to the dynamic changes of the network environment. It is worth mentioning that *progressive* in this context is related to the progressive playback position with which quality adaptation is performed.

After the playback is initiated, static peer resources have to be evaluated. These include: screen resolution, available bandwidth, and processing power. Based on this information, the IQA is executed to make a decision on the feasible video quality. Accordingly, peer selection and block selection are per-

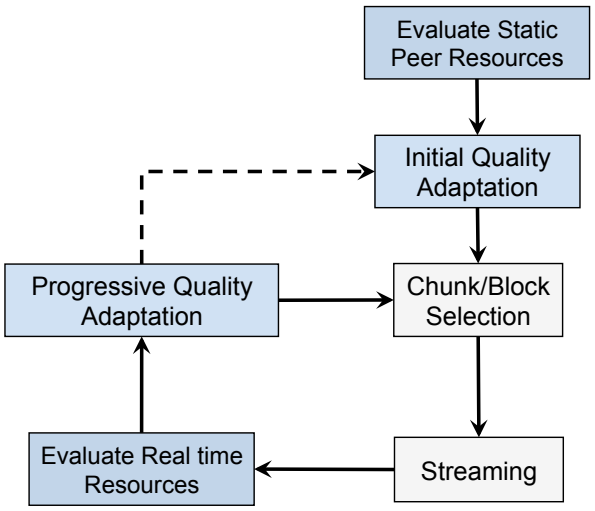


Figure 20: The quality adaptation loop.

formed following algorithms already explained in Subsection 4.1.4 and Subsection 5.1.3. Peers are selected in such a way that they are able to provide the selected video quality. After the neighboring peers have been contacted and upload slots have been assigned, block selection is done.

When the buffer has been filled with blocks matching the IQA-selected video quality, playback is started and the player switches from the *idle* to the *playing* state. Now adaptation to the dynamics of the system is performed. Thereby, real time resources are evaluated and the PQA is executed. To ensure continuous playback, the PQA is performed periodically, and if required, it would increase or decrease the selected video quality accordingly.

The PQA is used throughout the streaming session until playback is finished. However, it is still possible to start over and re-evaluate the IQA in case the peer undergoes a major resource change. For example, this happens when the video player is resized or when the access network is changed. In this case the streamable video resolution or bit rate changes. Subsequently, the IQA is re-evaluated and the steps explained above are repeated as indicated by the dashed line in Figure 20.

Next, we give more details on the quality adaptation modules and their role in the VoD system.

5.1.5 *Initial Quality Adaptation*

Initial Quality Adaptation (IQA) is invoked at the beginning of the streaming session or when the peer undergoes a major resource change. It is designed in such a way that each peer can determine its highest SVC video quality before starting to download the SVC video. The architecture of the IQA is depicted in Figure 21.

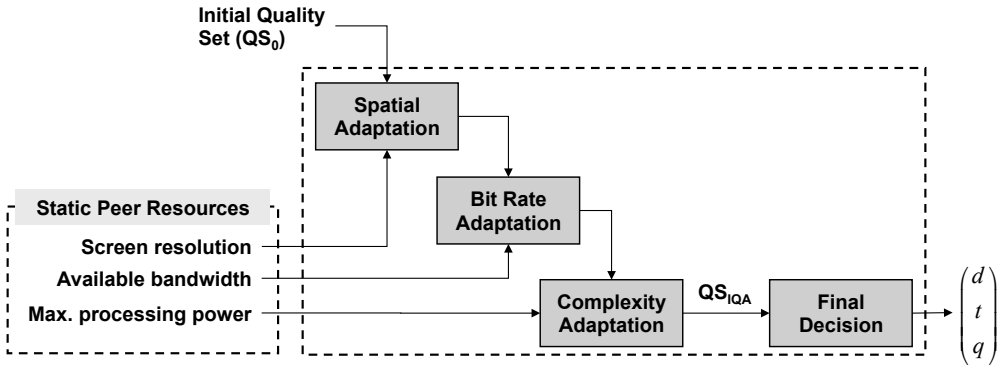


Figure 21: The Initial Quality Adaptation (IQA).

The basic idea behind the IQA is to compare the requirements of each layer of the video stream with the local resources[1] of the peer so that the layers that are not supported are left out of the retrieval process. The subtle property of the IQA, is that it has to make a decision on the quality level without having any information about effective throughput and system dynamics. Evidently, many systems face this challenge, since throughput measuring algorithms always require a certain startup time until they can provide reliable results.

According to the three dimensions of scalability in the SVC model, we have identified the following relevant local resources of a peer: screen resolution, bandwidth, and processing power. Subsequently, we have also designed the IQA using three sub-modules where each one considers one of the local resources as follows.

Spatial Adaptation

The resolution of the user's display or video player determines the picture size of the video to be downloaded. Therefore, the screen resolution will be used to restrict the spatial quality-level of a video. If the resolution of the screen is $W \times H$ pixels, then all layers with a resolution larger than $W \times H$ pixels are filtered out.

Bit Rate Adaptation

The download bandwidth of a peer corresponds to the maximal bit rate of the video stream it can receive. Therefore, bandwidth sets limits on the bit rates of the streamable SVC layers. If the bandwidth of a certain peer is B Kbps, then all layers with bit rates larger than B Kbps are filtered out. It is worth mentioning that in this basic design we assume that each peer is streaming only one video. However, the described algorithms can be expanded to consider streaming of multiple videos at once. Additionally, we assume that the bit rate of each SVC video quality can be accurately measured.

1 Local resources here mean hardware resources that do not change dynamically.

Complexity Adaptation

Decoding SVC video requires more processing power compared to non-scalable videos. Therefore, it is necessary to take processing requirements for decoding SVC videos into consideration and to match it with available processing resources. As a result, the video decoder is prevented from overloading weak mobile devices.

One major challenge while performing complexity adaptation is the lack of accurate models that capture the complexity of decoding SVC videos. Some early work in this area was done by Zhan et. al [61]. In this work, the authors provide insights in the context of estimating decoding complexity for SVC videos. Nonetheless, their models cannot be directly used in our system as the authors take many simplifying decisions. One major result of their study is that spatial quality has the largest impact on the decoding complexity. Therefore, we take a simple approach in which peers may decode only a certain number of spatial layers depending on their processing resources. In the general sense, the more processing power a user device is equipped with, the more spatial layers it may decode. Subsequently, if the required processing power of a certain peer is PP cycles/second, then all layers with complexity larger than PP cycles/second are filtered out. Since no models exist to calculate reliable estimates for decoding complexity of SVC videos, this module would be useful once such models exits.

It is worth mentioning that we assume a fixed processing power. Therefore, the complexity adaptation is only part of the IQA and not the PQA. Nonetheless, one can still consider this parameter to reflect the power level of a mobile device. Therefore, if the power level of a mobile device gets low, processing intensive operations should be avoided. However, algorithms required to take into account varying power level during video playback goes beyond the scope of this thesis.

Using the three types of local resources the SVC layers that are not compatible with local resources are filtered out. The IQA starts from an initial quality set QS_0 that contains all layer combinations. This set is passed through the spatial, bit rate, and complexity adaptation sub-modules to generate the final list QS_{IQA}.

Final Decision

The *final decision* algorithm receives the QS_{IQA} set from the IQA sub-modules. It is then up to it to make a final decision on the layer to be fetched. Here, we stick to a decision based on the layer with the highest bit rate in order to maximize resource utilization. Later in Section 5.3, we make use of QoE metrics in order to make a better final decision on which layer to choose.

5.1.6 *Progressive Quality Adaptation*

The Progressive Quality Adaptation (PQA) module is the dynamic part of the quality adaptation loop. The PQA is invoked periodically during the video

playback with configurable time intervals. Its main task is to adjust the selected SVC layer according to the changing system resources so that potential stalls can be avoided and a smooth playback is ensured. The architecture of the PQA is depicted in Figure 22.

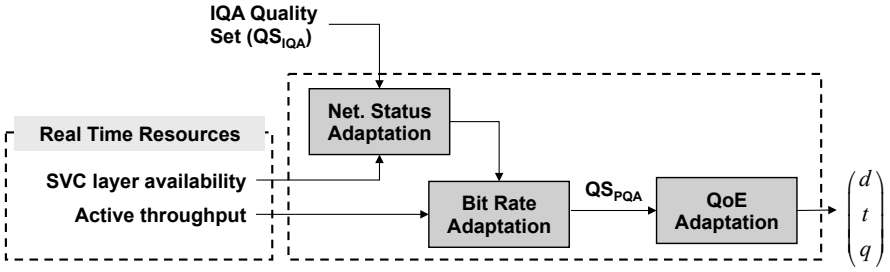


Figure 22: The Progressive Quality Adaptation (PQA).

The PQA uses time varying information regarding the network status measured through the *SVC layer availability* and the *active throughput* to adapt to changes. It starts with the IQA quality set QS_{IQA} that contains all layer combinations supported in terms of the static peer resources as explained above. This set is processed by the different stages of the PQA to produce the QS_{PQA} set that contains layers fitting to available resources. The two adaptation stages of the PQA form together the decision-making process, which are detailed next.

Net-status Adaptation

This part of the PQA keeps track of the block availability of all connected peers. Its objective is to check whether the current layer can be supported by the available blocks from current neighbors. Here *support* means that all the blocks in the high priority or buffering window can be downloaded without changing the currently connected peers. If this is not the case, then the SVC layer of the peer will be decreased to avoid performance degradation until new peers have been contacted. The adaptation process can be briefly described in the following steps:

1. The peer uses the information of all available blocks at its connected peers acquired through the so-called buffer-maps. Buffer-maps are periodically-refreshed indices of available blocks and are exchanged between all connected peers.

2. Then, the peer can calculate a neighborhood availability map for the blocks in its high priority window.

3. The availability map is then compared with the current layer of the peer. If the map covers all blocks to be downloaded for this layer in the high priority window, the current layer will not be changed. Otherwise, the layer will be reduced to the level that is covered by the availability map. In addition, if the availability map contains additional blocks of a higher layer, the PQA then switches up to this layer.

Using the net-status adaptation, the SVC layer of a peer can be adapted according to the real-time resources of its connected peers, so that the playback does not need to stop and wait for unavailable blocks. Consequently, the number of stalls can be reduced during the playback.

Bit Rate Adaptation

This step of the PQA changes the SVC layer by using the active download throughput. The goal of the bit rate adaptation is to predict possible buffer under-runs due to slow block supply. Therefore, it adapts the SVC layer so that the bit rate fits the dynamic throughput, thereby, avoiding potential stalls.

One major challenge for the bit rate adaptation is avoiding oscillations in the selected video quality. Therefore, a main question arises: how can we absorb short term fluctuations in the download throughput to avoid oscillating the selected video quality? We do this as follows. The bit rate adaptation is not executed with every PQA, rather it is only used when needed. This means that the bit rate adaptation is used only when the peer is having excess or deficit throughput.

Excess or deficit of throughput can be detected based on the buffering state, which means how full the buffer is, since it reflects the recent throughput for the current layer. Therefore, the buffering state is used as a trigger for the bit rate adaptation as follows:

1. We measure how full the buffer is with video data for the current SVC layer.
2. If this portion of the buffer is very low (e.g. less than 10%) or very high (e.g. more than 80%) then the bit rate adaptation is used.

Similar to the IQA, bit rate adaptation works by setting limits on the bit rates of the streamable SVC layers. If the throughput of a certain peer is T Kbps, then all layers with bit rates larger than T Kbps are filtered out.

Final Decision

Similar to the IQA, the PQA *final decision* algorithm receives the QS_{PQA} set from the PQA sub-modules and then makes a final decision on the layer to be fetched. We again stick to a decision based on the layer with the highest bit rate in order to maximize resource utilization. Later in Section 5.3, we make use of QoE metrics in order to make a better final decision on which layer to choose.

Summary

In this section, we have developed the quality adaptation mechanisms required for SVC-based VoD systems. Those mechanisms can adapt the video quality to both static and dynamic peer and system resources.

Next, we go over the next major contribution of this thesis. We present the media-aware networking techniques that investigate the benefits of advanced routers in the context of SVC P2P video streaming.

5.2 MEDIA-AWARE NETWORKING

In streaming applications, it is very important to have prioritized traffic management to achieve a good service quality. When classical non-QoS routers are congested, they usually drop packets that might be more important than others, i.e., in a media-agnostic fashion. The opposite, which is having the routing element act in a media-aware manner is denote in this thesis by *media awareness*. Future Internet networks should be able to better react to congestion of multimedia traffic by being media-aware.

However, the questions remain, how can we add more intelligence to routers for better congestion control? In addition, what is the impact of such techniques in the context of SVC-based P2P VoD?

In this section, we investigate the possible benefits of conveying more information about the video priorities to the routing elements to achieve media awareness. Major possible benefit of media awareness is the ability to perform better traffic management. Additionally, more sophisticated application-layer management algorithms are possible.

5.2.1 *Enabling Media-aware Networking*

Before detailing our approach, we first stress the need for including media awareness rather than classical QoS techniques.

Using media awareness, it is possible to have QoS at the application layer, therefore, enabling optimization in the direction of QoE-awareness. Additionally, efficient and custom-made QoS solutions can be developed and used as needed. For example, different media-aware routers and algorithms can be deployed for live and on-demand streaming, since both use different techniques for media awareness.

Building media-aware networks can be realized in various ways. We do not want to delve into lengthy explanations on the possible architectures; rather we just give an overview on possible realization opportunities.

One solution makes use of more flexible routers that can be built using virtualization techniques. Router virtualization is based on the idea of running multiple software routers on a single hardware router. The different software routers can have different roles and be activated and even migrated on demand. Recently, efficient implementations for router virtualization have been proposed where full network speeds can be achieved by combining a fast forwarding plane such OpenFlow[2] along with software routers [17].

Another advantage of router virtualization is that it enables the gradual development of media-aware solutions and protocols, which is not possible with classical approaches. Additionally, a virtualized router does not have to be running all the time as it can be executed only on-demand when congestion occurs, exactly at the point when performance and perceived quality would degrade.

Nonetheless, one can still argue that prominent QoS management techniques such as Differentiated Services [73] can achieve similar goals. But the main

2 http://www.openflow.org

issue there is that using the Type-of-Service bits, only limited priority information can be communicated with the router. Additionally, QoS can only be controlled by the network provider and not the content provider. Since the latter has a better idea about its users' access patterns, content popularity, and the scheduling of its own traffic, a router virtualization solution enables the implementation of more sophisticated management algorithms. Therefore, a solution similar to the one presented in this thesis allows for homogenous QoS handling of multimedia traffic across different networks. Additionally, the content provider, based on some business model with the network operator, can deploy its own policies and algorithms to implement custom-made media-aware solutions.

Router virtualization is one possibility for realizing media-aware networking. Since the discussion on possible network architectures goes beyond the scope of this thesis, we do not limit our solution to any. Therefore, we will refer to the router performing media awareness simply by the *media-aware router*.

5.2.2 *The Network Model*

The network model and scenario used to evaluate the potentials of media awareness in the context of SVC-based VoD are shown in Figure 23. There, routers connect different subnets and forward data from and to several end users. In a P2P streaming system, edge routers' upload utilization will generally become higher, since data would be flowing from end users to the core network [15]. This issue becomes especially evident when streaming video content with high quality.

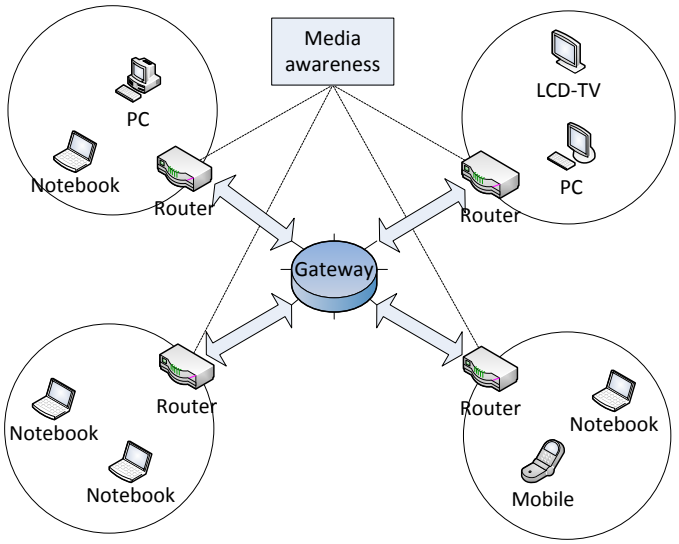


Figure 23: Media-aware network scenario.

5.2.3 *Priority Calculation of SVC Blocks*

Every SVC video block has certain temporal and SVC priorities, denoted by P_T and P_Q respectively. The temporal priority, P_T, generally expresses how soon a certain video block is needed. Therefore, video blocks that are closer to the playback position have a higher temporal priority. The SVC priority P_Q, on the other hand, expresses the importance of the SVC enhancement layers. Therefore, the base layer is assigned a higher priority since without it video decoding is not possible. Details on the two priorities were presented in Subsection 5.1.3.

If a certain receiver peer p_r is requesting blocks from sender peer p_s, then the temporal priority P_T and the SVC priority P_Q are included with every request as calculated in Equation 5.3 and Equation 5.4 respectively. When the sender peer decides to serve a certain requested block, the respective priorities are reported along with the actual transmission. The priorities P_T and P_Q will be used by the media-aware resource allocation algorithms as explained next.

5.2.4 *Media-aware Resource Allocation*

We now present our approach for achieving media awareness in an SVC-based P2P streaming system. The goal here is not to present a complex prioritization algorithm, but rather to assess the impact of simple media awareness in the context of SVC-based streaming.

During congestion, the media-aware router is activated. It then prioritizes and controls block transfers based on priority information that is reflected by the temporal and quality dimensions outlined above.

The main task of the media-aware router is to prioritize block transfers. A video block is a video part as defined by our SVC-based P2P streaming protocol. The size of a block usually ranges from 16 KB up to 2 MB. The media-aware router can retrieve the priority information for each block either using a separate communication channel or using deep packet inspection. To minimize processing overhead, only a single decision has to be made for all packets belonging to the same block. This greatly enhances scalability of the application layer processing algorithms. Figure 24 depicts the proposed architecture of the media-aware router.

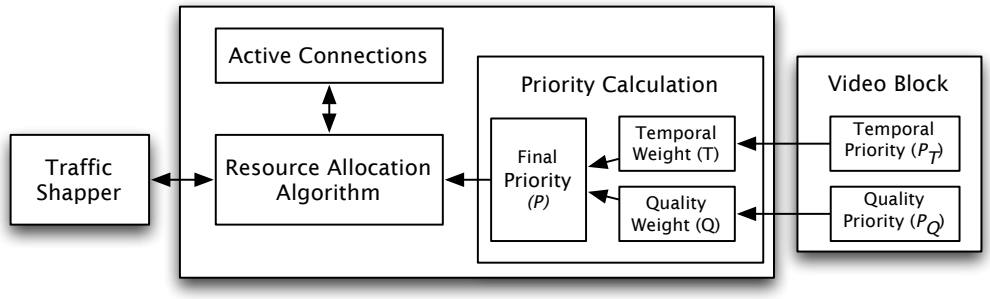


Figure 24: The media-aware router architecture for handling SVC video blocks within a media-aware network.

The media-aware router keeps a list of active transfers or connections that represent the blocks currently being uploaded along with their priorities. When there is a new incoming block, the router first retrieves the temporal (P_T) and quality (P_Q) priorities and adds this block to the list of outgoing transfers. When the media-aware router is becoming overloaded, congestion control is performed by using the priority algorithm to decide on how to allocate resources. For the actual priority calculation, we use exponential compensation to exaggerate the importance of blocks very close to the playback position ($P_T = 1$) or base layers ($P_Q = 1$). Therefore, the media-aware router calculates a single priority for each block transfer by combining the temporal and quality aspects. This priority P is defined as:

$$P = T\, e^{P_T} + Q\, e^{P_Q} \tag{5.6}$$

where T and Q denote the weights for the temporal and quality aspects respectively, with $T + Q = 1$. Therefore, the priority P ranges between one (lowest) and e (highest) according to Equation 5.6.

Once the final priority P is calculated, the resource allocation algorithm is executed, which - based on the list of active connections - decides on how to allocate resources. Accordingly, the traffic shaper for the router is used to apply the decided allocation policy.

Summary

In this section, we have presented our media-aware network approach that gives routing elements more information about the video data. We have derived the required priority-estimating functions used by the routers in our network to perform better resource allocation.

Next, we present the third and last major contribution of this thesis, which is using objective QoE metrics in the quality adaptation algorithms.

5.3 QOE-AWARE QUALITY ADAPTATION

In this section, we develop QoE-aware quality adaptation mechanisms that extend those presented in Section 5.1. Quality of Experience (QoE) estimation of SVC videos can be used by the quality adaptation algorithms to achieve better system performance. Therefore, the main motivation behind this section is to expand the use of classical QoS metrics used for the quality decisions to include other metrics that reflect how users perceive the different SVC video qualities.

5.3.1 *Approach Overview*

In order to use information about the QoE within the quality adaptation algorithms in the SVC-based VoD system, we devise an approach that is depicted in Figure 25. There are two major steps in our system: the *Quality Management* running at the server and the *Layer Adaptation* running at the peers.

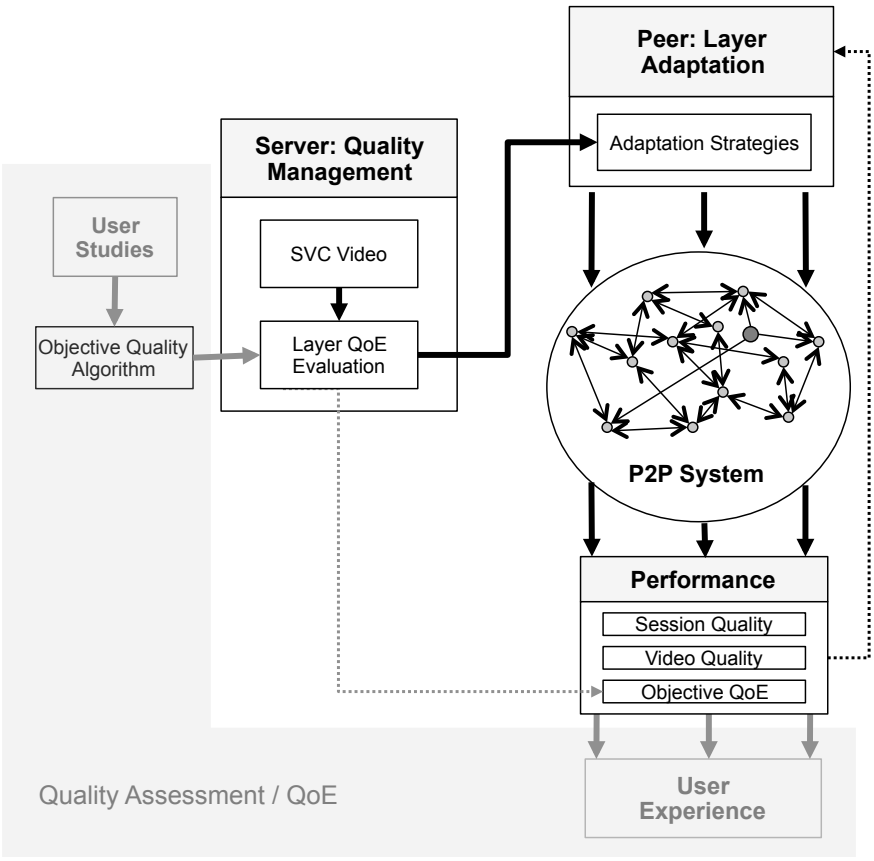


Figure 25: Approach for including objective QoE ratings into the quality adaptation algorithms.

The *Quality Management* step running at a server, additional to the tracker and content delivery servers, has the main task of calculating the QoE ratings of the SVC video sequence. This is done by the *Layer QoE Evaluation* module. This module calculates the QoE ratings for all possible video qualities, which are provided to the peers. As input, it takes the SVC video sequence as well as an *Objective Quality Algorithm*.

The *Objective Quality Algorithm* is used to calculate a single QoE rating of a single video quality. Recently, the state of the art in objective quality estimation is able to give very good objective estimates of the QoE that come close to subjective tests (see Section 2.4). Although the actual objective quality algorithm is derived from user studies, we do not directly make use of user studies in our system. By executing the objective quality algorithm on each possible video quality of the SVC video file, the layer QoE evaluation module generates n ratings for an SVC file with n video qualities (similar to the examples listed in Appendix A.4). For each SVC file that is offered by the system, respective QoE ratings have to be calculated.

The *Layer Adaptation* module comprises the adaptation algorithms as introduced in Section 5.1. These algorithms have the task of deciding on a certain SVC layer given the resources available at the peer. Therefore, these algorithms provide a decision on which blocks to request from other peers given the differ-

ent resources. The standard adaptation algorithms do not use any information concerning the QoE when deciding on which layer to choose.

The QoE ratings are used by the peers to make certain adaptation strategies. The adaptation strategies, as we explain next, address possible opportunities and limitations when deciding on the SVC layer to choose. Since each SVC file has different QoE ratings, the adaptation strategy itself is not static; rather, it is different for each video file. One major research question we address in this thesis is: given a certain adaptation strategy derived from the QoE ratings, what is the impact and reaction of the system to this strategy.

In our system, the QoE ratings are used by the peers in order to select SVC layers that have the maximum ratings and, thereby, the best video quality. Second, the ratings themselves are again used in the performance evaluation of the system. The subtle difference between both is that the first makes a recommendation on which layer to choose while the second checks how these recommendations were being sustained by the peer. The main reason why the chosen layers cannot be sustained can be related to the highly dynamical nature of the P2P network where resources are not guaranteed. Nonetheless, as we later show, choosing SVC layers based on QoE metrics helps in making P2P resources more reliable.

In the next sections, we elaborate on the quality management and the layer adaptation modules.

5.3.2 *Quality Management*

So that the peers can take QoE ratings of the SVC video into account, they have to be estimated, derived, and sent to the peers. This is done by the *Quality Management* part of the system running at the server. The details behind this part are given next.

Deriving the QoE Ratings

The QoE ratings are derived using the *objective quality algorithm* for all possible SVC layer combinations. Although subjective user studies can be performed to estimate the QoE ratings for all SVC layer combinations, this is not feasible as it is not possible to perform user studies for every single offered video content. Therefore, we believe that more subtle methods, i.e., objective methods, which can work completely human free, are more viable for such applications. It is worth mentioning that the actual development of objective metrics involves insights gained through studying the human visual system. Therefore, objective QoE methods make it possible to automate the quality management process. This is essential especially for future streaming applications where a huge amount of video content would be offered to the users.

In the general sense, our system is able to work with any objective quality algorithm for deriving the QoE ratings. We just specify some formal requirements on the values of the QoE rating. They should be positive numbers and range between zero and one. We assume that zero represents the best quality while one represents the worst quality as done in [113]. Considering two SVC

layers having R_1 and R_2 as QoE ratings, if $R_1 < R_2$, then we assume that Layer 1 has a better quality than Layer 2.

QoE Ratings Offline Calculation at the Server

Since all the peers require having the QoE ratings and since calculating these ratings are processing intensive, the server entity has the task to calculate and manage these ratings. Therefore, the content provider would calculate the QoE ratings for each video file it is offering to its customers. Since we focus on a VoD scenario, this is feasible by simply preprocessing the video file before its release. The QoE ratings are subsequently distributed to all the peers, either by the tracker or the streaming server. The ratings can also be enhanced by dividing the video file into smaller scenes. Nonetheless, in this thesis we stick to a single set of QoE ratings for each video file.

Advantages of Objective QoE

We now elaborate on why we choose to use objective QoE estimation methods.

- *Automation of QoE Estimation.* The main advantage of using objective QoE is the possibility to automatically process, generate, and manage the QoE ratings for a huge amount of video content. Additionally, it is possible to evaluate the impact of different adaptation algorithms even using simulative evaluation, which is not possible using subjective QoE metrics.
- *No Need to Compute QoE Ratings at Peers.* Since we do not use subjective methods and additionally pre-computed the objective QoE ratings for all possible SVC layers in advance, the peers do not need to invest their resources in this task. In contrast to the work presented by Zhai et al [105], no calculation is required at the peer to derive the QoE ratings. The main limitation of the mentioned work is that the peers have to invest much processing power in estimating the received video quality. Evidently, deriving objective QoE ratings is very processing intensive [77]. Since we consider a scenario with heterogeneous devices and that respects limited processing capacities, it is essential to minimize the processing footprint of the quality adaptation algorithms. According to [105], the peers can evaluate the quality rating only after they have switched to the target layer. This solution, therefore, is not suitable for assessing the full potential of quality adaptation.
- *No Need to Send a Reference Video.* Our algorithm works simply by using the sent QoE ratings of the SVC video file without any additional information. The positive side effect of this is that no additional video reference (whether full or reduced) has to be transmitted to the peers, thereby, preserving precious bandwidth for the actual video transmission. The amount of data required for the reference data can be rather large. For example, using the Video Quality Metric (VQM) with a reduced reference, the amount of overhead can reach up to 14% [29]. Therefore, only the pre-computed ratings of the different SVC layers are sent to the peers and not the huge reference data.

5.3.3 *QoE-aware Adaptation*

In this section, we consider the quality adaptation mechanisms presented in Section 5.1 while extending the classical QoS-based algorithms with ones that consider the QoE. To extend these mechanisms with QoE-awareness, we use the QoE ratings generated by the objective QoE algorithm. We extend both the IQA and the PQA with the intelligence required to use the QoE ratings.

QoE-aware Initial Quality Adaptation

The *QoE-aware Initial Quality Adaptation (IQA)* is, similar to the IQA presented in Subsection 5.1.5, executed at the beginning of the streaming session with the distinction of using QoE ratings to decide on the final SVC layer.

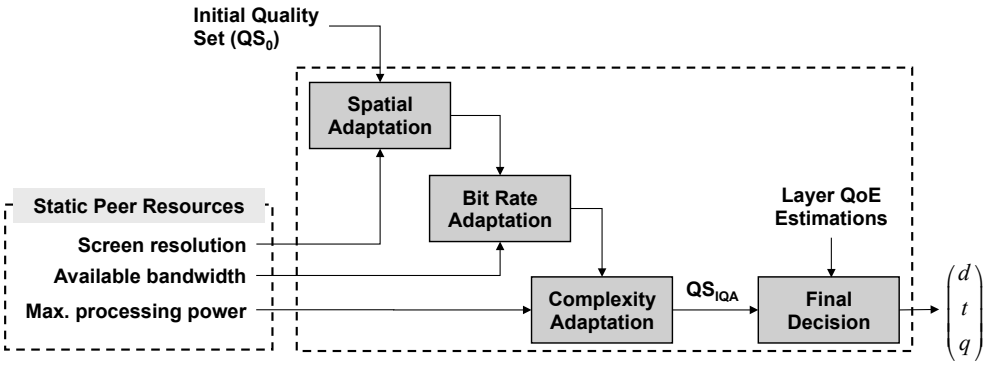


Figure 26: QoE-aware *Initial Quality Adaptation*.

As presented in Figure 26, the final decision on the layer chosen by the IQA is based on the *layer QoE estimations* sent from the server. This means that given the set of layers supported by the local resources, the final chosen layer is that which has the highest QoE rating.

QoE-aware Progressive Quality

The QoE-aware *Progressive Quality Adaptation* (PQA) is again extended from the classical PQA presented in Subsection 5.1.6. The PQA has the main task of keeping track of active resources, such as throughput, and react to the changes by increasing or decreasing the video quality. As depicted in Figure 27, the QoE-aware PQA has an additional module called *QoE Adaptation*. This module uses the QoE ratings calculated by the *Layer QoE Evaluation* module in order to take the QoE into account while making the final decision. In the basic design, this module would select the video quality with the highest QoE rating in order to stream those layers that have the best impact on QoE.

Since the PQA is executed periodically, there is time between two executions for the adaptation process. Motivated by the fact that too frequent layer variation can have an adverse effect on the QoE [111], we present mechanisms to switch the layer smoothly. Therefore, we define two steps for the actual layer adaptation. The first step is the *Layer Decision* while the second is the *Layer Switching*.

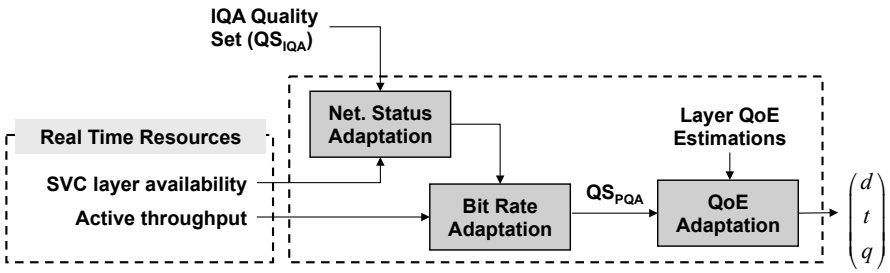


Figure 27: QoE-aware *Progressive Quality Adaptation*.

QOE ADAPTATION. QoE Adaptation is performed over two steps, the layer decision and layer switching. Both take the QoE ratings as input in order to select layers with the highest ratings. Those ratings, as explained above, are delivered by the server from the layer QoE evaluation module. The overall logic and design for our two step adaptation as well as their input parameters and outputs are visualized in Figure 28.

The QoE adaptation is performed as follows. The layer decision is executed on the PQA list to select a new layer or what we call a *target layer*. The layer switching step follows by defining a switching or adaptation path that starts from the current layer and smoothly changes the quality to the target layer.

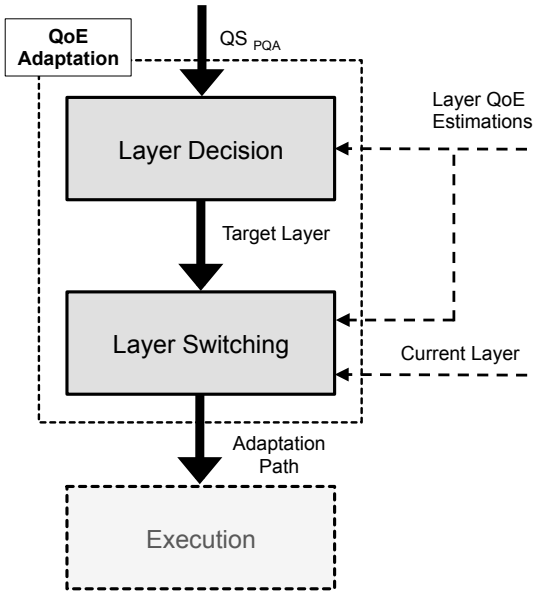


Figure 28: Steps of the QoE-aware adaptation mechanism: *Layer Decision* and *Layer Switching*.

Layer switching, as we present later, offers different possibilities on how to switch from SVC layer A to layer B. One simple possibility is to directly switch making a jump. Other possibilities include smoother switching in terms of keeping the variation of QoE ratings or even the layer bit rates as low as possible.

In the following two sections, we go into the details on the *Layer Decision* and the *Layer Switching* modules.

5.3.4 *Layer Decision*

The *Layer Decision* module has the main task of deciding on a layer that fulfills a certain criterion. The main criterion of the layer decision is to maximize the QoE ratings of the played out video. Therefore, this module, given several SVC layer options, would choose the video quality that has the highest QoE rating. For sake of comparability, we further evaluate three other alternatives. These are the *Simple Layer Decision*, the *Maximum Bandwidth Utilization*, and the *Prioritized Dimensions*. These different strategies are described in more details in the following.

Simple Layer Decision (D_{Sim})

This first strategy is the simplest layer decision strategy, which can be done. This strategy was inspired by the related work presented by Oechsner et al. [74]. In that work, the authors propose that layer selection follows a slow ramp up strategy, in which the base layer is selected for 10 seconds, then adding more layers along the way. Since the original approach does not involve any quality adaptation, i.e., the selected video qualities are static and defined before streaming starts; we slightly modify the strategy by including PQA functionalities to allow for a more fair comparison.

First, all peers still start by requesting the base layer to allow for fast startup time. After that, the peers slowly start increasing the layer.

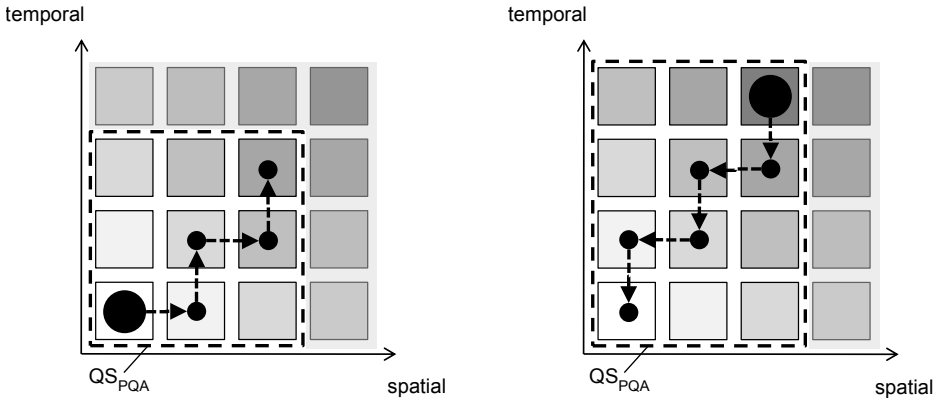
Layers are switched according to the following algorithm. When increasing the SVC layer, the different scalability dimensions are increased in a round-robin manner in the order *spatial*, *temporal*, and *quality* [74]. We always switch up the layer with the lowest value. If one dimension reaches the limit set by the PQA, it is no longer increased; rather, the other scalability dimensions are changed. For switching down the layer, we use the reversed order for the layers, i.e., *quality*, *temporal*, and *spatial*. In all cases, the selected video qualities have to fit throughput-wise. This means that all selected video qualities have to have a bit rate smaller than or equal to the peer's active throughput, otherwise much video stalling would occur.

It is worth noting that, this simple strategy, as described by Oechsner et al. [74], does not take any QoE considerations in account when deciding on the selected video quality.

Figure 29 illustrates an example for this mechanism. For simplicity of presentation, we assume an SVC file of only two dimensions. On the left side, we show the adaptation decision when increasing the layer, while on the right side we show the adaptation decision when reducing the layer.

Prioritized Dimensions (D_{Prio})

This strategy works by defining a certain pattern for the layer switching. This strategy uses the QoE ratings not of individual layers but rather of whole scal-



(a) Layer increase (b) Layer decrease

Figure 29: Examples for adaptation decisions using D_{Sim}.

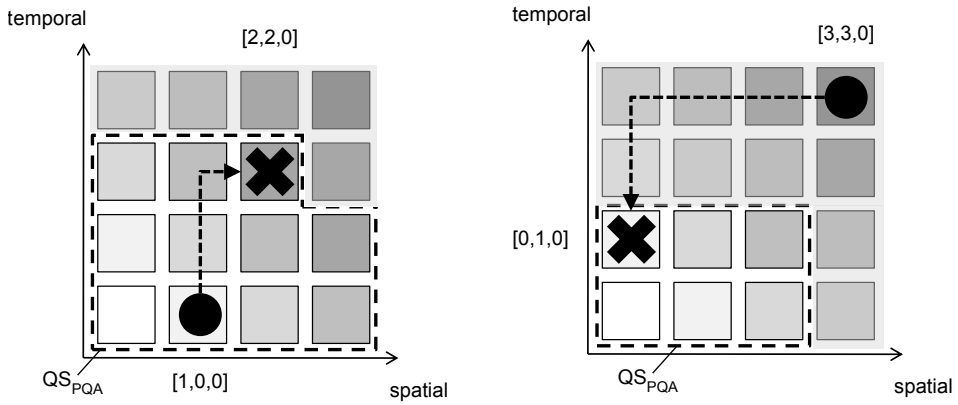
ability dimensions. Therefore, it is required that we define the order or importance of the scalability dimensions. In [113], it has been shown that increasing the layer in one SVC dimension has larger impact on the QoE in comparison to other dimensions. That study concludes that it would be best for the QoE to aim at maximizing first the *temporal* dimension and then the *spatial* dimension. Since this strategy adapts only one dimension at a time, it is regarded as a simple strategy that we use for sake of comparison.

The concrete adaptation works as follows. Given the layer from the previous quality adaptation, and while having enough resources, the quality is increased in the dimension that has the highest impact on QoE. This is done until a certain dimension has saturated, at which the second most important dimension is increased and so on. It is worth noting that, the algorithm still obeys the PQA in terms of only selecting layers that the peer can actually sustain, whether in terms of bandwidth, screen resolution, or processing capacity, as defined in the QS_{PQA} set (see Subsection 5.1.6). Upon scarce resources, the quality is decreased following exactly the opposite order. Therefore, first the quality, then the spatial, and last the temporal dimension.

Figure 30 depicts an example while having an SVC video with four temporal, four spatial, and one quality layers.

On the left hand side, we present the layer decision in case of a layer increase from [1,0,0] to [2,2,0]. The dashed polygon indicates the newly supported layers as selected by the PQA. The actual algorithm is demonstrated using the arrow that is directed from the layer marked with a circle (old layer) to the layer marked with a cross (new layer). Following the general recommendation to first maximize the temporal dimension, this arrow first traverses the temporal dimension until hitting the bounds set by the PQA and then starts traversing the spatial dimension until it reaches its target layer.

On the right hand side, in Figure 30b, we show the process of reducing the layer [3,3,0] to [0,1,0]. In this case, first the spatial, then the temporal levels are decreased while searching for the layer that fits the new resources as indicated by the dashed polygon.



(a) Layer increase (b) Layer reduction

Figure 30: Examples for D_{Prio} with dimension priorities *temporal* > *spatial*.

Although this strategy does not use QoE ratings of individual SVC layers, it still does use the ratings of the dimensions, making this a rather simple strategy. The newly selected video quality will be used for later adaptation as a starting point.

Maximizing the Bandwidth Utilization (D_{Bw})

This *Layer Decision* strategy has the goal to maximize the bandwidth utilization at the peers and would choose the layer, out of those selected by the PQA, that have the highest bit rate. Therefore, this strategy works simply by going though the QS_{PQA} set and choose the layer with the highest bit rate.

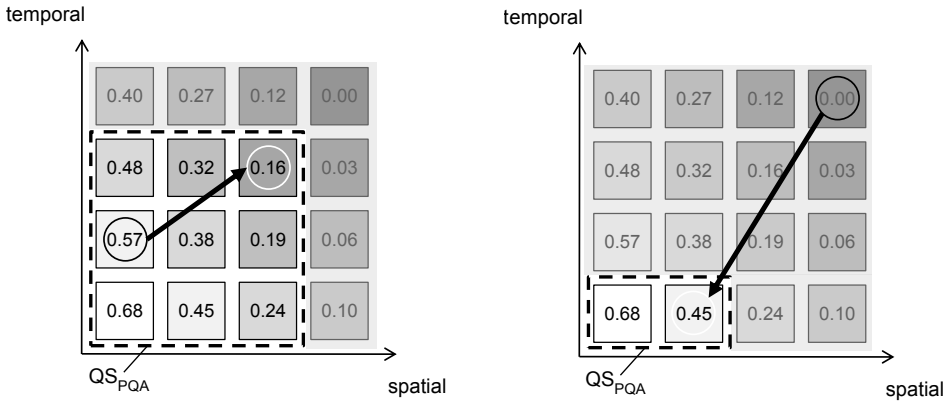
This strategy is QoS-based and has been widely used in most SVC-based P2P streaming systems [13, 55, 70, 27, 66, 26, 103]. Therefore, using this strategy we compare our QoE-aware algorithms with the related work in this area.

Maximizing QoE Ratings (D_{QoE})

The *Layer Decision* strategy constitutes the major contribution of this thesis since it uses full knowledge about the QoE ratings of the different SVC layers during quality adaptation.

Using the information provided by the *Layer QoE Evaluation* module, we get a QoE rating for each SVC layer. Keeping in mind that a QoE rating ranges between zero (best quality) and one (worst quality), we now illustrate how such QoE ratings could look like. We present in Figure 31 an example based on the Video Quality Metric (VQM). The ratings start off with zero for the highest quality at the upper right most layer. The values decrease if we go in any dimensions towards the base layer, which then has the lowest rating. As mentioned above, those ratings are calculated at the server and sent to the peer before streaming starts.

Coming back to the D_{QoE} strategy, it works by simply selecting the layer that maximizes this rating given the different constraints. The algorithm iterates over all layers with the dashed polygon, or PQA chosen layers, and



(a) Layer increase

(b) Layer reduction

Figure 31: Examples for D_{QoE}. The numbers present the QoE ratings based on the Video Quality Metric (VQM). The x and y axes represent the different SVC layers.

chooses the layer that has the highest QoE rating and, therefore, the best video quality.

5.3.5 *Layer Switching*

Since the time between two adaptation processes can be configured to span several seconds or even minutes, switching from one layer to the other can be done more smoothly by stretching the process over a longer time.

The *Layer Switching* module has the task of defining how to switch to a new layer. In other words, given an old Layer A and a new Layer B, as calculated by the layer decision module, the layer switching algorithms define the set of layers that have to be passed when switching from A to B.

The main reason that motivates a smoother switch is the fact that the perceived video quality can be negatively influenced by too frequent quality switches [111]. Therefore, a stepwise adaptation makes it possible to have quality adaptation with smaller steps in between. To avoid having too much layer variation, this mechanism, therefore, samples the adaptation path in order to put a limit on the amount of steps until the target layer is reached. Therefore, each video quality is sustained for a certain minimum time as suggested in [72].

Now we present the different possibilities for the layer switching. These are the *Simple Layer Switching*, the *Prioritized Dimensions*, the *Minimized Variation in Bandwidth*, and the *Minimized Variation in QoE Ratings*. Later in this section, we describe how the adaptation path is sampled in order to limit the amount of adaptation steps.

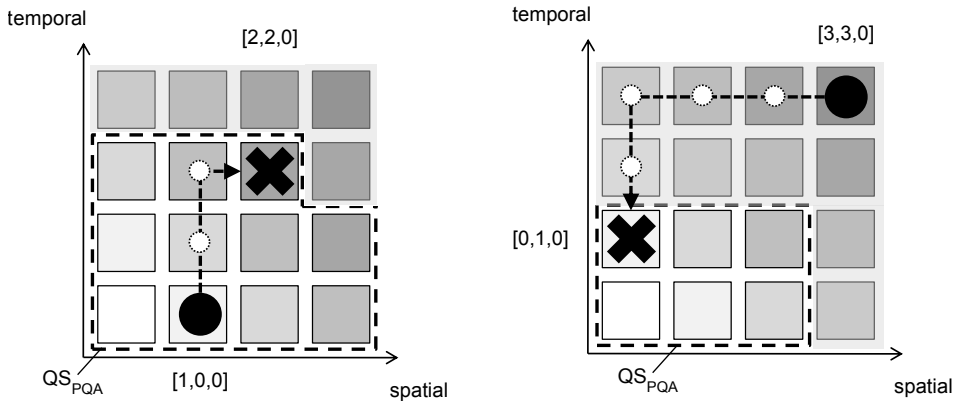
Simple Layer Switching (S_{Sim})

The simplest way to switch between two layers is to perform a direct jump. This strategy is developed for the sake of comparison. Given the layer decision derived from the previous step, this switching algorithm will not choose any intermediate steps and will directly jump in a single step to the layer selected by the preceding *Layer Decision* algorithm.

Prioritized Dimensions (S_{Prio})

The *Prioritized Dimensions* switching strategy is based on a similar idea to that presented in the prioritized dimensions *Layer Decision* (see Figure 5.3.4). Accordingly, the layer is switched in steps following a path in each dimension and starting with the dimension that has the highest impact on the QoE.

Again here we do not consider the concrete QoE ratings of individual layers but those of individual dimensions. This strategy is simplistic in the sense that it does not take into account the QoE ratings or the bit rate of the individual layers traversed during the switching. Besides having multiple steps, the D_{Prio} switching works similar to the S_{Prio} selection strategy. The typical procedures of layer increase and decrease are depicted in Figure 32 at the left and right hand sides respectively. Starting with the left hand side, a layer increase works by incrementally switching following the dimensions that has a larger impact on QoE, in this case starting with the temporal dimension. The steps are incremented until we reach the boundary set the PQA. Then, we start increasing the layer in the second dimension in terms of impact on QoE, in this case the spatial dimension.



(a) Layer increase (b) Layer reduction

Figure 32: Examples for S_{Prio} with dimension priorities *temporal* > *spatial*.

The algorithm for switching down the layer follows the same concept but using the reverse order of SVC dimensions. Therefore, we first start reducing the spatial resolution and last the temporal dimension until the quality set defined by the PQA is reached.

For ease of understanding, here we stick to a 2-dimensional SVC file, however, the explained strategy can still be applied to a 3-dimensional SVC file.

As derived in [113], the three dimensions have the following order in terms of their impact on the QoE: *temporal - spatial - quality*. Therefore, the adaptation path for an increase from the base layer to say layer [3,2,2], would go through the following adaptation steps: [0,0,0], [0,1,0], [0,2,0], [1,2,0], [2,2,0], [3,2,0], [3,2,1], to [3,2,2]. The same path holds in case the layer is to be switched down from [3,2,2] to [0,0,0] but with reversed order.

Minimized Variations (S_{Bw} and S_{QoE})

We now present two strategies for switching between two layers based on the idea to have a smooth transition based on two metrics. These strategies are the *Minimized Variation in Bandwidth* (S_{Bw}) and the *Minimized Variation in QoE Rating* (S_{QoE}). We present them together as they use the same approach to minimize both the bandwidth and the QoE ratings variations.

Starting with S_{Bw}, this strategy uses the bandwidth variation as its minimization metric. Therefore, the aim is to find a path from the old to the new layer, which would inflict the least variation in terms of bandwidth utilization. Traversing layers that have a smooth transition of bandwidth can be quite important especially in P2P systems where resources are scarce and having a smooth layer switch is important.

Coming to the S_{QoE}, this strategy uses the variation of QoE ratings of the traversed layers as its minimization metric. Therefore, the goal is to find a path from the old to the new layer that would go through the layers that have the closest QoE ratings. By doing this, the user perceived quality of the video is changing smoothly between the original and the target layers.

In order to implement both switching strategies we refer to graph theory by generalizing the SVC cube into a graph. Using classical and extensively researched algorithms from graph theory, we can have efficient solutions to our problem of minimizing absolute bandwidth values or minimizing QoE variation.

The above-described problem is related to the so called *Single-Source Shortest Paths* problem. Prominent solutions to such problems are the well known Bellman-Ford and Dijkstra algorithms [25]. We choose to use the Dijkstra algorithm as it exhibits lower complexity and can be easily applied to our scenario where the edges between the SVC layers (the variation of bandwidth or QoE ratings) are positive numbers. As the Dijkstra algorithm has been extensively described in literature (see [25]), we abstain from describing it here; interested readers should refer to the literature.

In the following the old layer is used a source node while the target layer is used as a drain or destination. We now explain the details on how the graph is derived from he SVC cube.

DERIVING THE GRAPH FROM THE SVC CUBE. Before we can apply the Dijkstra algorithm required for the S_{Bw} and S_{QoE} algorithms, we need to first derive a graph from the SVC cube. This is done as follows: each SVC layer in the SVC cube is modeled as an individual node in a graph. Since adaptation is only expected to happen within the PQA-selected video qualities set, i.e., with

the QS_{PQA}, only the nodes that are within this set are connected. Therefore, we make sure that we reach only supported layers.

Edges are generated by connecting only adjacent nodes, since we consider switching one video quality at a time. Figure 33 depicts an example of deriving the graph from a 2-dimensional SVC video along with certain PQA-selected video qualities (shown within the dashed polygon).

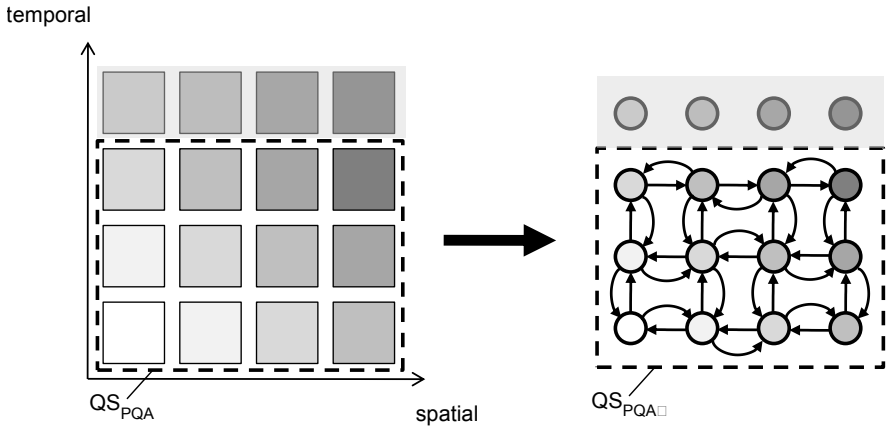


Figure 33: The process of deriving the graph from the SVC cube. SVC layers are modeled as nodes and are connected to their direct neighbors in the set QS_{PQA}.

DERIVING THE EDGE WEIGHTS. We need to derive the actual weights of the edges that connect the different nodes. This is the step that defines the difference between the S_{Bw} and the S_{QoE} switching strategies.

In the case of S_{Bw}, the edge weight is defined as the absolute value of the difference between the bit rates of the SVC layers represented by the adjacent nodes. In the case of S_{QoE}, the weight is defined as the absolute value of the difference between the QoE ratings of the SVC layers presented by the adjacent nodes.

In Figure 34 we present an example on how the edge weights are derived. The presented values can be either in Mbps in case the graph is for S_{Bw} switching, or in QoE ratings in case the graph is for S_{QoE} switching.

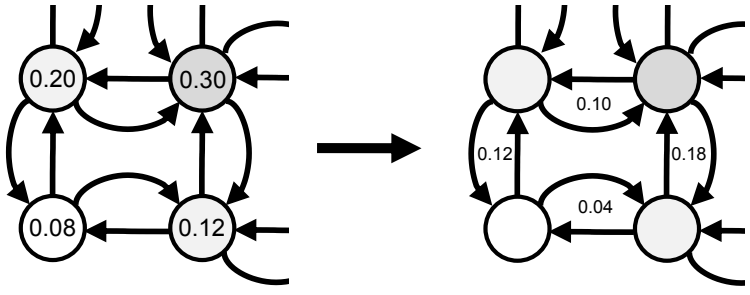


Figure 34: The process of defining the edges' weights. The node labels show example bit rates for the described SVC layers in Mbps. An edge weight is the absolute difference between the two adjacent nodes' bit rates.

Minimizing the Variation

The actual calculation of the switching path is simply performed by applying the Dijkstra algorithm on the calculated graph. Dijkstra will make sure the retrieved switching path minimizes the switching variation of either the bit rate or the QoE ratings. An example derivation of the switching path is depicted in Figure 35.

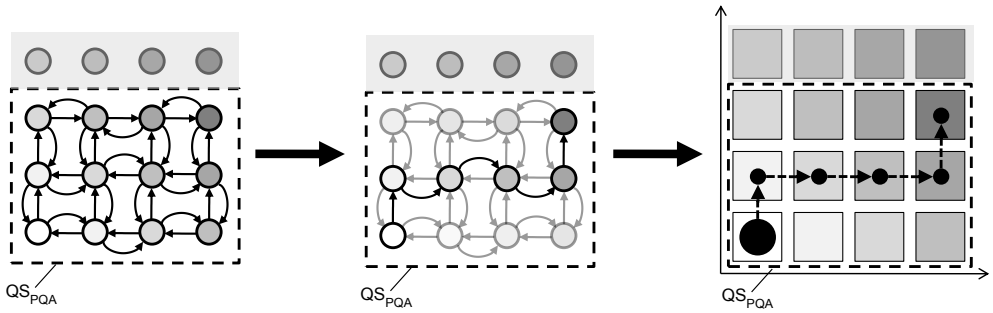


Figure 35: The process of retrieving the adaptation path from the graph. The first step is the application of the Dijkstra algorithm, followed by an interpretation of the result as an adaptation path.

SAMPLING THE RETRIEVED PATHS. Because the main objective of performing *layer switching* is to have a smooth transition between an old and a new selected video quality, it is essential not to overwhelm the user with too many layer variations on the way [111]. Therefore, it might not be that beneficial to switch through all the layers along the switching path. Thus, we need to sample the path to reduce the number of jumps.

Since the PQA mechanism is executed periodically every T seconds (called adaptation interval), the switching from the old to the new layer should be performed before the next PQA operation. Therefore, the length of the adaptation interval poses a limitation on the speed at which switching has to be performed. The larger the adaptation interval, the slower the switching can be performed and vice versa.

Nonetheless, it is important to define a minimum switching time between two layers as to let the users notice a difference when the layer is being switched. Although more specialized subjective tests have to be performed to get a good view on how to choose this value, some studies [72] suggest to use some two to three seconds as a minimal value for switching SVC layers.

Therefore, in our simulations we use a value of three seconds as the time between two switches. Since the default value of the adaptation interval is ten seconds, a switching path involves three adaptation steps by default. Unless stated otherwise, the sampling of the switching path is performed for all strategies.

THE SAMPLING MECHANISM. The next step is to sample the path to define the intermediate layers to go through. To do this, we define the time each layer has to be sustained as T_{sus} in seconds, and the adaptation interval T_{PQA} also

in seconds. Since we do not want to have overlapping adaptation operations, each operation has to end before the next adaptation is executed.

Given an adaptation path P, as the result of a previous *Layer Switching* process, first the maximum number of adaptation steps N_{max} is derived. N_{max} is calculated as $\lfloor T_{PQA}/T_{sus} \rfloor$. In case $T_{PQA} = 10$ and $T_{sus} = 3$ seconds, N_{max} is 3 steps.

If the length of the calculated switching path P is smaller than N_{max}, then no sampling is performed. If the length of the calculated switching path P is larger, then the sampling process chooses N_{max} steps out of the path P. In the later case, we define the step size for the adaptation s_{width} as $size(P)/N_{max}$. The index of the layer chosen out of the original path P of the i-th entry of the sampled path is determined by: $index(i) = \lfloor s_{width} * i \rfloor$. The last element of the sampled path is always chosen to be the last element of the path P in order to meet the target layer set by the PQA.

5.3.6 *Configuration of Strategies*

We have presented how the *Layer Selection* and *Layer Switching* strategies can use different strategies. In this thesis, we perform an extensive study on the different combinations of the two mechanisms.

The combinations which we evaluate are shown in Table 1. Next, we describe those combinations in more details.

Name	**Layer decision**	**Layer switching**	**QoE-aware IQA**
D_{Sim} S_{Sim}	Simple layer decision	Simple layer switching	
D_{Prio} S_{Sim}	Prioritized dimensions	Simple layer switching	
D_{Bw} S_{Sim}	Max. band. utilization	Simple layer switching	
D_{QoE} S_{Sim}	Max. QoE rating	Simple layer switching	X
D_{QoE} S_{Bw}	Max. QoE rating	Min. band. variation Dijkstra	X
D_{QoE} S_{Prio}	Max. QoE rating	Prioritized dimensions	X
D_{QoE} S_{QoE}	Max. QoE rating	Min. QoE variation Dijkstra	X

Table 1: Evaluated adaptation configurations.

D_{Sim} S_{Sim} - SIMPLE SELECTION. This is a reference configuration, which employs the simplest selection and switching strategies. As a reference, it allows to better assess the benefits of our other strategies. The major related work that uses a very similar approach was presented in [74]. Since this strategy always starts by selecting the base layer, gradually increasing the layer, it is expected to offer benefits in terms of fast startup time.

D_{Prio} S_{Sim} - PRIORITIZED SELECTION. The *Prioritized Selection* strategy combines the prioritized dimensions layer selection with the simple layer switching strategy. Using this strategy, we evaluate the effectiveness of switching the

layer according to predefined priorities of the SVC dimensions. As discussed above and presented in [113], we assume a default SVC priority order as follows: temporal, spatial, and quality dimension.

D_{Bw} S_{Sim} - BANDWIDTH SELECTION. The *Bandwidth Selection* strategy combines the bandwidth-based layer selection with the simple layer switching strategy. Using this strategy, we are able to evaluate how well the system performs in case the peers try to simply maximize their download utilization. This in general fits to most systems in the research field that can be classified as QoS-based [13, 55, 70, 27, 66, 26, 103]. Therefore, those pieces of related work are compared to our QoE-based adaptation algorithms.

D_{QoE} S_{Sim} - QOE DIRECT SELECTION. The *QoE Direct Selection* strategy combines the QoE-based layer selection with the simple layer switching strategy. By aiming at selecting the layer that has the highest QoE rating, it is quite natural that the overall video quality perceived by users will be better. Nevertheless, we also show how this affects the system and how it compares to the other strategies.

D_{QoE} S_{Bw} - QOE BANDWIDTH SWITCHING. The *QoE Bandwidth Switching* strategy combines the QoE-based layer selection with a switching strategy that minimizes the difference in bit rates of the intermediate layers while switching quality. The switching, in this case, tries to minimize the variation in bandwidth utilization over the intermediate steps. Since this is a QoE-based switching strategy, we compare it to other QoE-aware strategies.

D_{QoE} S_{Prio} - QOE PRIORITIZED SWITCHING. The *QoE Prioritized Switching* strategy combines the QoE-based layer selection with a switching based on prioritized dimensions. Here, the switching is performed in such a way that more important SVC dimension are giving priority while switching to the target layer.

D_{QoE} S_{QoE} - QOE DIJKSTRA SWITCHING. Finally, the *QoE Dijkstra Switching* strategy combines both the QoE-aware selection and QoE switching mechanisms. This constitutes the main algorithms and techniques that include QoE information into the adaptation algorithms in our streaming system.

5.3.7 *Realization using the Video Quality Metric*

Our approach is based on the idea of using objective quality algorithms and metrics to estimate the QoE of SVC videos. Thereby, QoE ratings can be calculated offline and without human intervention. We, therefore, use the state of the art in objective QoE, namely the Video Quality Metric *VQM* [77] introduced in Subsection 2.4.1.

The ratings generated by the VQM follow our requirements in terms of values, where those range between zero and one, where zero indicates best quality with no visual impairments and one indicates worst video quality. This metric

is considered to be state of the art as it was independently and extensively evaluated and shown to correlate with the human perception of video quality for both TV-like video resolutions [95] as well as for high definition videos [102].

Summary

In this section, we have developed algorithms and mechanisms required to integrate objective QoE metrics with the quality adaptation algorithms. The information about the QoE is used in two steps. The first is to improve the layer decision algorithms in order to choose SVC layers with high QoE ratings. The second is in performing layer switching that would reduce the amount of layer variation. Thereby, the use of QoE metrics should improve the overall performance from the user point of view.

5.4 SUMMARY

In this chapter, we have presented the quality adaptation mechanisms constituting the major contributions of this thesis. We have shown that those mechanisms span to different parts of the system. They are required at the edge where the peers, based on local view, are able to adapt the video quality according to the available resources as presented in Section 5.1. Quality adaptation extends as well to the routing elements in future networks. As presented in Section 5.2, it is possible to reveal the priorities of SVC blocks to the routing elements in order to make better resource allocation decisions. Finally, in Section 5.3, we have presented how objective QoE metrics can be used in order to improve the performance from the user point of view, thereby, completing the *quality adaptation* mechanisms and algorithms required for future P2P VoD systems.

In the next chapter, we perform an extensive simulative analysis of the designed system and mechanisms. In our evaluation, we address various issues ranging from parameter tweaking to demonstrating the superiority of the designed system. We further provide insights on SVC-based quality adaptation in P2P VoD system.

6 SYSTEM EVALUATION

So far, we have presented the quality adaptation mechanisms essential to address the challenge of peer heterogeneity in P2P VoD systems. This chapter presents the extensive evaluation we have performed in order to assess the performance of the designed system and mechanisms.

The quality adaptation mechanisms described in this thesis were mainly evaluated using the discrete, event-based simulator PeerfactSIM.KOM [90]. A test bed evaluation was also performed, which is presented in Section 6.6.

We proceed by first presenting the used metrics which we have defined to capture the performance of an SVC-based VoD system. Then, a model for system capacity is derived that allows us to assess how well P2P helps in improving the performance of VoD systems. Subsequently, we present the evaluation results and analysis for the three major contributions of this thesis, quality adaptation based on SVC, media-aware networking, and QoE-aware quality adaptation.

6.1 METRICS

Before we can evaluate the performance of the VoD system and the impact of quality adaptation, we first need to define relevant metrics that reflect the key features of the P2P VoD system. It is worth noting that we use the reliable Transmission Control Protocol (TCP) for the data transmission, similar to renowned VoD systems [7]. Therefore, we assume that playback is stopped if no video data is available. Unlike many classical live streaming systems that use unreliable transmission protocols, e.g., the User Datagram Protocol (UDP), there will be no artifacts displayed to the user in case of packet loss. Since many of the metrics defined for such systems mainly measure the level of distortion when packets are lost [82, 68], such metrics are not usable in the context of a VoD system.

The metrics we use can be divided into two main categories: *session quality* and *SVC video quality* metrics.

6.1.1 *Session Quality*

In this category, we consider the most important factors that affect the users' watching experience from the playback delay point-of-view. They are:

- *Startup delay*. With this metric we measure how long the user waits for the playback after streaming has been initiated. The startup delay for a peer p is defined as

$$delay_{init}(p) = t_{videoStart}(p) - t_{bufferingStart}(p), \tag{6.1}$$

where $t_{bufferingStart}(p)$ is the time when peer p starts with buffering the video while $t_{videoStart}(p)$ is the time the video playback starts. It is commonly agreed that having a shorter startup time is better for the user experience.

- *Stalling events per peer*. This metric represents the total number of times playback stopped, also known as stalling events. A stalling event is a video freeze due to video pieces important for playback not arriving on time causing an empty buffer. In our design, video frames and pieces are not dropped, rather, playback is stopped until the missing pieces arrive. User experience is degraded with more encountered playback stalls, therefore, we aim at minimizing the number of such events.

- *Average stalling duration*. This metric represents the average duration of a playback stall. We calculate this metric as follows:

$$stalling_{avg}(p) = \frac{\sum_{s \in Stalls_p} duration(s)}{|Stalls_p|}, \tag{6.2}$$

where $Stalls_p$ is the set of all stalling events of peer p, and $duration(s)$ is the duration of a stalling event s. The startup time is not considered in this metric. We assume that a shorter average stalling duration implies a better user experience.

- *Total playback delay per peer*. For simplicity, sometimes we would like to have one metric that summarizes the above-mentioned metrics for the session quality. Therefore, we define the total playback delay as the sum of the startup delay and all the stalling durations as follows:

$$stalling_{total}(p) = delay_{init}(p) + \sum_{s \in Stalls_p} duration(s), \tag{6.3}$$

with $delay_{init}(p)$ being the startup delay, as defined above, $Stalls_p$ the set of all stalling events of peer p, and $duration(s)$ the duration of a stalling event s.

We additionally represent this metric in relation to the total video length. Therefore, we define the relative delay as the total playback delay normalized by the video length as follows

$$relative_{delay} = \frac{stalling_{total}(p)}{Time_{playback}}. \tag{6.4}$$

It is worth noting that we cannot place concrete statements on the *best* values for the above-mentioned metrics for the session quality. In the optimal case, all delays should be zero. This is, however, not possible in reality given the finite capacities of the system. Therefore, we aim at minimizing all the delays to

reasonable ranges. Concerning what could be *good* values; we do not place any limitations as this highly depends on the context [72]. For example, someone using the VoD system free of charge is more willing to wait longer for the video to start. On the other hand, a user watching a football match of his favorite team would be very annoyed by playback stalls. Concluding this point, we can say that context-dependent user studies have to be performed in order to map session quality to user QoE. Nonetheless, for a general context, we assume that bringing those delays to reasonable-low ranges is an essential step towards better video experience.

6.1.2 *SVC Video Quality*

Session quality metrics are not enough to capture the whole performance of an SVC-based VoD system. What is needed to complete the set of metrics, are what we call the *SVC Video Quality* metrics. This set of metrics captures the performance from the SVC point of view. These metrics are:

- *Number of layer changes during video playback.* This metric represents the number of layer changes encountered by a peer throughout the whole streaming session. The point of this metric is to capture the effect the Progressive Quality Adaptation (PQA) by counting the number of layer changes.

  A study done by Zink et al. in [110] has reported that having too frequent layer changes might be more annoying for users than watching the video at the lowest quality. Therefore, we measure the average number of SVC layer changes as an indictor of SVC video quality. A smaller number of layer changes along with a high SVC quality indicate a better VoD system. In this case, the system is able to sustain high video quality to the peers without having to continuously change it.

- *Layer Change Amplitude.* In addition to the number of SVC layer changes, the amplitude of the layer change is important for the perceived quality [111].

  We define the layer change amplitude as the absolute difference between the layers (over three dimensions) as follows:

$$\mathrm{amplitude}(l_1, l_2) = |d_1 - d_2| + |t_1 - t_2| + |q_1 - q_2|, \tag{6.5}$$

  where $l_1 = (d_1, t_1, q_1)$ and $l_2 = (d_2, t_2, q_2)$ describe the old and new SVC layers respectively. Each layer is represented through its spatial, temporal, and SNR quality dimensions.

  According to [111], it is reported that having a lower amplitude per layer change makes the quality-adaptive system better accepted by users.

- *Relative Received Layer.* In addition, we consider how high the received SVC layer is for each peer during the playback. Since each peer has different local resources and thus can retrieve only a certain range of SVC

layers, we cannot directly use the *absolute layers* received by the peers to compare their performance. Instead, we define the *relative layer* to assess whether the peers are receiving the highest quality they can actually get given their resources. The relative layer of each received video piece is equal to the received layer divided by the initial SVC layer calculated by the Initial Quality Adaptation (IQA), as follows:

$$quality_{rel}(d,t,q) = \frac{d+t+q}{D_{init}+T_{init}+Q_{init}}, \tag{6.6}$$

where d, t, q are the received layers in spatial, temporal and SNR dimension respectively as chosen by the quality adaptation algorithms, while $D_{init}, T_{init}, Q_{init}$ are the initial SVC layers as chosen by the IQA.

Since the layer selected by the quality adaptation algorithms can never be higher than that selected by the IQA (which is determined by physical resources), the relative received layer calculated by Equation 6.6 falls into the interval [0,1] for all peers. Using this metric, we can better compare the received SVC layers for peers with different local resources. A higher value of the relative received layer indicates that the peer is better able to maintain the quality supported by local resources. Having a lower value, on the other hand, means that although there are enough local resources, the P2P network itself is not able to provide the highest layer to the peer. This can be due to network congestion or weak server resources.

- *VQM quality*

  The last metric in the SVC Video Quality category is the VQM quality. It represents the quality of the SVC video as perceived by a user. The Video Quality Metric (VQM) [77] is an objective metric that highly correlates with subjective evaluation of the QoE (See Subsection 2.4.1).

  Using the VQM metric and as explained in Subsection 2.4.1, each SVC layer is mapped into a QoE rating or VQM value. For each peer we eventually calculate the average over all the layers played out throughout the streaming session. The actual VQM values used in our evaluation were derived from the work by Zinner et al. in [113]. This data is listed in Appendix A.4.

  The important aspect of using the VQM quality is the fact that one can directly map it into a QoE value based on the Mean Opinion Score (MOS) method [77, 113]. In Table 2, we present the mapping between the VQM ratings and the respective subjective QoE. Using this metric, we aim at evaluating how good the quality adaptation algorithms are in selecting and maintaining the high VQM quality layers.

6.2 MODELING SYSTEM CAPACITY

In order to simulate our system with reasonable values for the server capacity for certain peer characteristics, we derive a model for system capacity for SVC-based P2P VoD.

VQM	MOS
< 0.2	5 (excellent)
$\geqslant 0.2$ & < 0.4	4 (good)
$\geqslant 0.4$ & < 0.6	3 (fair)
$\geqslant 0.6$ & < 0.8	2 (poor)
> 0.8	1 (bad)

Table 2: Mapping from the VQM rating to subjective QoE [77].

The system capacity in the case where all peers retrieve the same video quality is easier to calculate. When SVC is used, on the other hand, things get more complicated as the quality retrieved by the peers depends on many parameters. Therefore, we aim at capturing the major influencing factors so that to derive a model for the system capacity of an SVC VoD system. To the best of our knowledge, this is the first model for system capacity in such systems.

We now present the basic notations used by the model. This is an extension of the model already presented in Subsection 4.2.1.

- S: number of servers
- u_S: server upload capacity
- $G = \{g_1, g_2, g_3\}$: the three sets of uploaders, where peers in the same set have the same capacity. Three sets are assumed enough to capture most common characteristics of today's devices.
- U_g, $g \in G$: number of uploaders for each set that have the whole file
- D_g, $g \in G$: number of downloaders for each set having an incomplete file
- u_g, $g \in G$: upload capacity of a single peer in a certain set
- d_g, $g \in G$: download capacity of a single peer in a certain set
- r_g, $g \in G$: bit rate for the initial SVC layer. This is the bit rate of the layer selected by the IQA. It is assumed the same for all peers within the same set since this quality is chosen according to the available resources. The bit rate of the IQA-selected layer is smaller or equal to the download capacity of the peers in each set ($\forall g \in G : r_g \leqslant d_g$).
- f: average peer prefetching factor (See Subsection 4.2.1)
- g: average peer upload utilization (See Subsection 4.2.1)

Therefore, we can calculate the following:

- The total number of uploaders and downloaders:

$$U = \sum_{g \in G} U_g$$

$$D = \sum_{g \in G} D_g$$

- The average video bit rate retrieved by all peers:

$$\bar{r} = \frac{\sum_{g \in G} r_g \cdot D_g}{\sum_{g \in G} D_g}$$

- The average upload capacity of all peers in the system:

$$\bar{u} = \frac{\sum_{g \in G} u_g \cdot (U_g + D_g)}{\sum_{g \in G} (U_g + D_g)}$$

The total offered system upload throughput is

$$u_{total} = D \cdot g \cdot \bar{u} + \sum_{g \in G} U_g \cdot u_g + S \cdot u_S. \tag{6.7}$$

Since the total required download throughput for all peers is $D \cdot f \cdot \bar{r}$, then

$$D \cdot f \cdot \bar{r} = D \cdot g \cdot \bar{u} + \sum_{g \in G} U_g \cdot u_g + S \cdot u_S. \tag{6.8}$$

Therefore, we can derive the required number of servers as follows:

$$S = \frac{D \cdot (f \cdot \bar{r} - g \cdot \bar{u}) - \sum_{g \in G} U_g \cdot u_g}{u_S}. \tag{6.9}$$

Restructuring this equation, and for a given number of servers, we can calculate the minimum server capacity as:

$$u_S = \frac{D \cdot (f \cdot \bar{r} - g \cdot \bar{u}) - \sum_{g \in G} U_g \cdot u_g}{S}. \tag{6.10}$$

6.3 EVALUATING QUALITY ADAPTATION USING SVC

We now present the evaluation results for the first step of our quality adaptation algorithms. This constitutes the *quality adaptation using SVC* presented in Section 5.1.

Our evaluation goals in this section are: first, to evaluate the impact of quality adaptation on the performance in comparison to a non-adaptive VoD system, i.e., a media-agnostic one. The second goal is to assess the impact of the different quality adaptation algorithms, and how the IQA and PQA mechanisms influence the performance. Finally, we investigate how having different intervals of the progressive quality adaptation mechanism affects the performance while aiming at identifying the major tradeoffs of the system.

6.3.1 *Simulation Setup*

We now present the simulation setup and parameters used to evaluate the quality adaptation mechanisms. These are the peer resources, the SVC file properties, and the workload.

Peer Resources

Since our main goal of quality adaptation is to investigate how SVC can help in addressing the challenge of peer heterogeneity, we consider three peer sets, each having different resources. Peer resources are configured as shown in Table 3. The three peer sets are referred to as *slow*, *medium*, and *fast* peers. Slow, medium, and fast in this context refer to all resources: screen resolution, upload bandwidth, and download bandwidth. The given peer resources help us to assess the impact of heterogeneous resources in terms of bandwidth and screen sizes.

	Servers	Clients_{slow}	Clients_{medium}	Clients_{fast}
Number	4	30	30	30
Upload BW (Kbps)	6000	128	320	800
Download BW (Kbps)	-	256	560	1200
Player resolution (Pixels)	-	176×144	352×288	704×576

Table 3: The used peer resources.

SVC File Properties

What makes our work distinct from the related work is the fact that we account for the layer characteristics derived from the original SVC standard. Therefore, it was important to have a good model for the used video file. For this end, we have relied on deriving the SVC video characteristics from an SVC video file encoded using the JSVM SVC Reference Software [80].

Table 4 gives an overview on the SVC video characteristics used for this section. The SVC video has three spatial layers (d), four temporal layers (t), and one SNR layer (q)[1], with a total of 12 SVC layer combinations. From now on, a layer combination will be simply called a *layer*. The two rightmost columns represent both the partial bit rates of the individual layers as well as the total bit rate for the whole quality level. We have encoded the SVC file with relatively low SNR quality in order to have low overall bit rates and achieve scalable simulations. Later on for the test bed evaluation in Section 6.6, we use higher bit rates for the SVC file.

1 We had to use just a single SNR layer and avoid using SNR scalability altogether since, at the time of writing of this thesis, the JSVM encoding software [80] was unreliable when used with SNR scalability.

Layer Index	SVC Level (d,t,q)	Picture size (Pixels)	Frame rate (fps)	Partial Bit Rate (Kbps)	Total Bit Rate (Kbps)
0	0,0,0	176×144	3.75	60	60
1	0,1,0	176×144	7.5	30	90
2	0,2,0	176×144	15	30	120
3	0,3,0	176×144	30	30	150
4	1,0,0	352×288	3.75	180	240
5	1,1,0	352×288	7.5	90	330
6	1,2,0	352×288	15	60	390
7	1,3,0	352×288	30	60	450
8	2,0,0	352×288	3.75	270	510
9	2,1,0	704×576	7.5	150	660
10	2,2,0	704×576	15	180	840
11	2,3,0	704×576	30	160	1000

Table 4: SVC video structure with respective quality levels, partial bit rates, and total bit rates.

Workload

The workload parameters used in our simulations are presented in Table 5. The simulations are performed as follows.

We consider a typical P2P VoD scenario with 90 peers accessing the same SVC video content. Such a number of peers for one video file has been shown to be quite typical in VoD and BitTorrent systems [50]. The peers arrive based on an exponential distribution, which reflects the behavior of P2P systems [97]. We make use of four content delivery servers, each with six Mbps upload capacity. The playback buffer is chosen to be seven seconds so that to ensure a small startup time and acceptable playback delay. Each peer is allowed to setup ten parallel download and 15 upload connections. The video is five min-

Parameter	Value
Number of peers	90
Peer arrival pattern	Exponential
Inter-arrival time	90 second
Playback buffer size	7 seconds
Maximum download connections	10
Maximum upload connections	15
Video length	5 minutes

Table 5: Used workload configuration.

utes long, which is the average length of videos on the YouTube platform, the mostly used VoD service worldwide.

To get an idea how this workload affects the overall peer arrival, we run a simple simulation to measure the number of online peers and their states. In particular we make a distinction between seeders, peers with running playback, and peers whose playback has finished.

In Figure 36, we show a graph depicting the evolution of the peers in the system. There, we show the number of peers in three categories plotted against the simulation time in seconds. Please note that this is only one simulation run, which is just intended to better understand how a simulation proceeds. All further simulations are repeated ten times to rule out any random effects where we calculate 95th percentile confidence intervals.

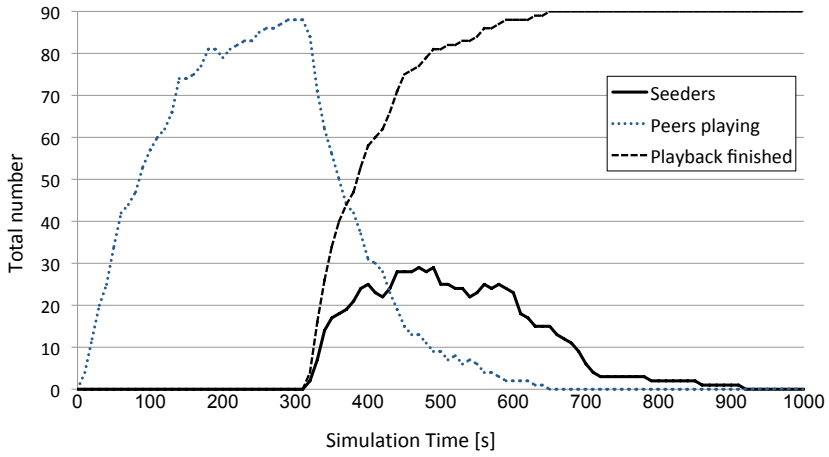
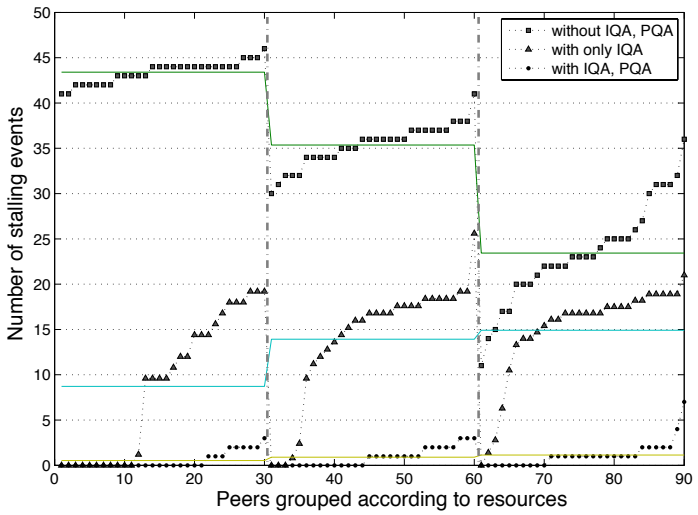


Figure 36: An example on the evolution of the peers in the system following the flash crowd pattern.

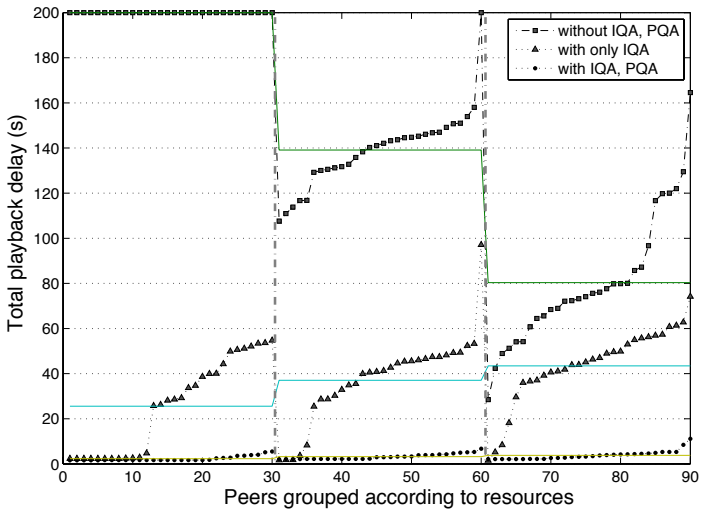
Figure 36 depicts the so-called *flash crowd* behavior typically found in P2P VoD systems. Subsequently, many peers tend to join the system within a short time span causing a surge in the number of requests. Since P2P techniques are especially beneficial in such flash crowd scenarios due to their resource scalability features, we use this workload for the rest of our evaluations. Please note that the number of seeders is never as high as the peers that finish playback. The reason behind this is that peers that finish playback stay as seeders for a random time ranging between zero and ten minutes.

6.3.2 *Quality-Adaptative Versus Media-Agnostic VoD*

In this first scenario, we evaluate how our proposed adaptation algorithms improve the performance of the P2P VoD system. We simulate our streaming system in three different cases: with no adaptation at all, i.e., all peers try to continuously retrieve the highest quality possible as in any media-agnostic system, with adaptation algorithms utilizing first only IQA and then with both IQA and PQA.



(a) Number of stalling events per peer



(b) Average playback delay per peer

Figure 37: Quality-adaptive streaming using SVC versus streaming using a media-agnostic system (no IQA, no PQA). The peers are divided into three groups according to their resources (slow, medium, fast). For each group, the peers are sorted according to the performance metric in the Y-axis.

The results for the session quality are presented in Figure 37. The left subfigure illustrates the average number of stalls while the right subfigure illustrates the average total playback delay for each peer in the network. The x-axis refers to the peer IDs in the VoD network. For better comparability, we present the per-peer results in an increasing order and further divide the results into three groups according to the bandwidth capacities of the peers, starting from slowest (on the left) to the fastest (on the right). The horizontal lines present the average values for each group.

Looking at the session quality performance of the three groups when no adaptation is used, we see strong variations in performance. Starting from left

to right: the weak, medium, and strong peers, had an average of 43, 35, and 23 stalling events respectively. This performance gap is even more visible for the total delay, where the maximum delay of 200 seconds reached. This is a threshold we have set after which the peers would leave the system without watching the whole video. What we see here is the natural effect of correlated playback delay and resources usually evident in media-agnostic VoD systems. This leads to excluding peers with weak resources, forcing them to leave the system.

However, already with the addition of the IQA, the slow peers can take part in the system and even have good performance. The slow, medium, and fast peers had only 9, 14 and 15 stalling events respectively. The total delay mounted to 25, 37, and 43 seconds for the three groups respectively.

Another interesting improvement gained when using the IQA, is the homogeneous performance for the three groups. We see that having fewer resources does not affect the session quality, but rather only reduces the video quality. Although the group with lower bandwidth can only receive low quality video, it can, nevertheless, enjoy continuous playback.

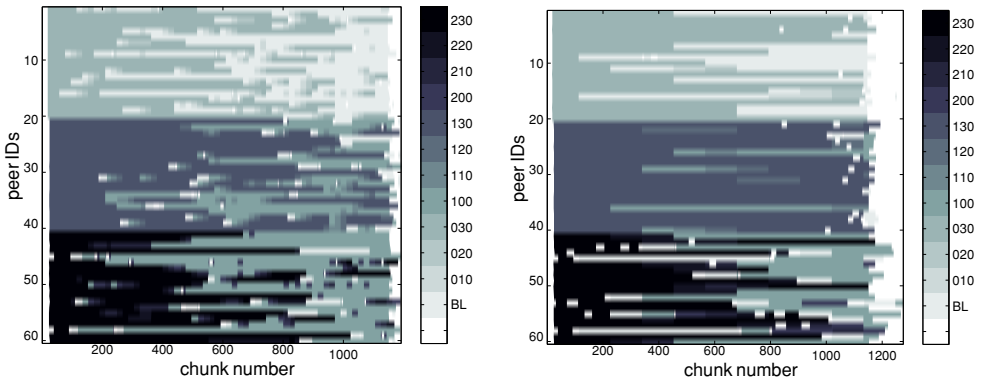
IQA is essential to adapt the system to static resources, however it is not enough, as it cannot predict system dynamics. As can be seen from the lowest curves in Figure 37, the performance when using both IQA and PQA was the best. Each peer, irrespective whether slow, medium, or fast, witnessed on average two stalling events and had three seconds of total delay. The PQA, therefore, helps in achieving better session quality and a performance that is more homogeneous across heterogeneous peers.

6.3.3 *Impact of the Adaptation Interval*

From the previous evaluation, we see the need to have both initial and progressive quality adaptation. For the PQA, the question arises: how often should it be invoked? I.e., how often should each peer adapt to system variations? To better understand the effects of this parameter and to understand the tradeoffs regarding adaptation dynamics, we evaluate the system with different PQA or adaptation intervals.

To assess the effect of having different adaptation intervals, we present in Figure 38 a visualization of the received layers for the peers when having an adaptation interval of five (Figure 38a) and 30 (Figure 38b) seconds. For clarity, we present only 20 peers per resource group. The peers are grouped according to their resources, with strong peers in the bottom (ID range: 40-59), medium peers in the middle (ID range: 20-39), and weak peers in the top (ID range: 1-20). Each line represents the retrieved video quality for one peer over the whole video file (for different video pieces). A darker color indicates a higher received layer. A pure white color indicates a playback stall due to missing video data.

We can observe the effect of having different adaptation intervals on the overall layer decision at the peers. Having a more frequent adaptation (left figure) makes the peers perform more changes in their quality since we observe more color variations along single lines. This seems to indicate that we



(a) PQA interval T=5 seconds

(b) PQA interval T=30 seconds

Figure 38: Visualization of the layer selection decision for different PQA intervals.

should have longer adaptation intervals. To better understand the effects, we now perform a more thorough analysis of changing the adaptation interval.

6.3.4 *Session Quality versus SVC Video Quality*

After having an idea about the general impact of different adaptation intervals, we now extend our analysis and simulate the system with the following adaptation intervals: 5, 10, 20, 30, 45, 60, and 90 seconds. To better quantify the effects of having those intervals, we present results using the metrics defined in Section 6.1. Session quality performance (total playback delay) is presented in Figure 39, while SVC video quality (layer changes, relative received layer) is presented in Figure 40. To further assess the importance of having IQA, we run two sets of simulations, one with only PQA (all peers start with highest video quality) and one with full quality adaptation using both IQA and PQA. Both sets of simulations are compared with media-agnostic streaming, i.e., when no quality adaptation is performed.

Through the comparison of the session quality for the different adaptation intervals, we can see that the more frequently PQA is invoked, the shorter is the total playback delay (Figure 39). The reason for this is that with a larger adaptation interval, the peers will be slower to react to system dynamics. Based on this observation, we can conclude that the performance in terms of the session quality decreases with a larger adaptation interval. Furthermore, when the IQA is not used, the adaptation behavior is no longer predictable. In the range of 5-45 seconds, the total relative delay is almost the same, and then it strongly increases for higher values. This can be explained by the fact that in P2P systems performance degradation at one peer is quickly propagated to the whole network. Therefore, the IQA is necessary, since it adapts the quality at all peers already from the beginning.

We have also investigated the received SVC video quality. What we desire is a high relative layer level that changes as less as possible, since too frequent layer changes tend to annoy users [110]. From Figure 40a we can see that as the adaptation interval grows, the number of layer changes becomes smaller.

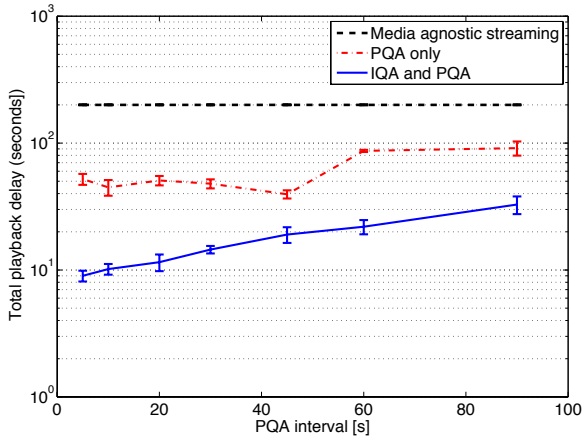
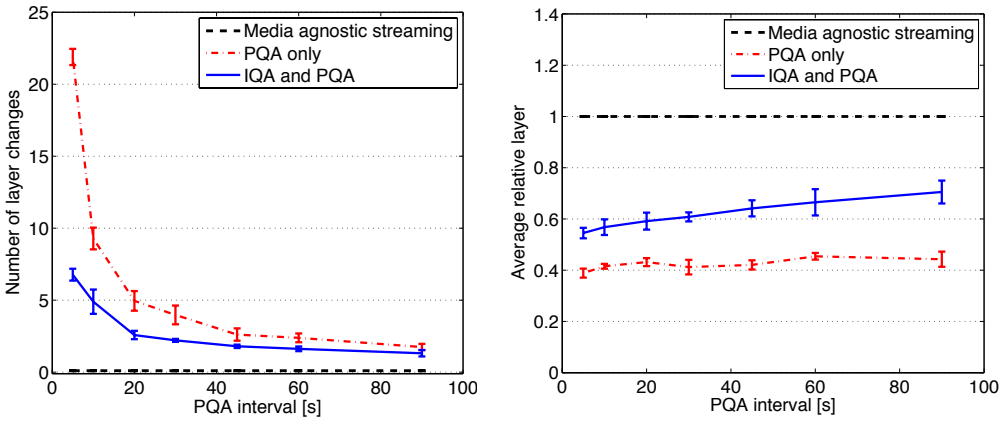


Figure 39: Total playback delay with different adaptation intervals.

When the interval is infinitely large, the number of layer changes becomes zero, since the layer selected by the IQA will be used throughout the streaming session. On the other hand, Figure 40b shows that the average relative received layer increases with a larger adaptation interval. The reason for this is that the PQA decreases the layer level to avoid potential stalls during the playback (which explains the better session quality). The more frequently the PQA is invoked, the more layer drops it may cause. Consequently, the average layer level throughout the playback will also decrease. From the results of Figure 40, we can conclude that the SVC video quality for the peers is better with a larger adaptation interval.



(a) Number of layer changes

(b) Relative received layer

Figure 40: SVC video quality with different adaptation intervals.

To summarize, the relation between session quality and SVC video quality exhibits a tradeoff for the different adaptation intervals. Therefore, one has to carefully address this tradeoff to achieve a compromise for the performance of the two metrics when choosing the adaptation interval. Depending on which aspect is more important for the users, the adaptation interval can be adjusted accordingly to meet the given requirements. For this specific workload con-

sidered here, we may say that an adaptation interval ranging between 10 and 20 seconds provides the best performance. This also gives us insights into the level of dynamics in P2P systems, since a peer viewing the system from the edge experiences the changes within some tens of seconds.

6.4 EVALUATION OF MEDIA-AWARE NETWORKING

We now come to the evaluations of the second major contribution of this thesis, the media-aware network system presented in Section 5.2.

We have two goals for this evaluation. The first is to assess the impact of media-awareness in SVC-based VoD systems. Second, we evaluate the interdependencies between adapting at the edge and adapting in the network using the media-aware system.

6.4.1 *Simulation Setup*

We still use the simulation setup, SVC file properties, and workload presented in Subsection 6.3.1. Here, we just need to provide the additional parameters that are relevant for the media-aware network and system.

The peers are now distributed over four subnets, as depicted in Figure 41. All traffic leaving a subnet goes through the router that performs media awareness upon congestion.

The reference approaches, which we compare our approaches against, are a media-agnostic network and a network that uses DiffServ. The media-agnostic network applies classical congestion control and random packet dropping. As for the DiffServ network, all packets of urgent blocks (within 7 seconds after the current playback position) are given a high priority service class. The DiffServ router prioritizes those blocks based on a first-come first-serve policy.

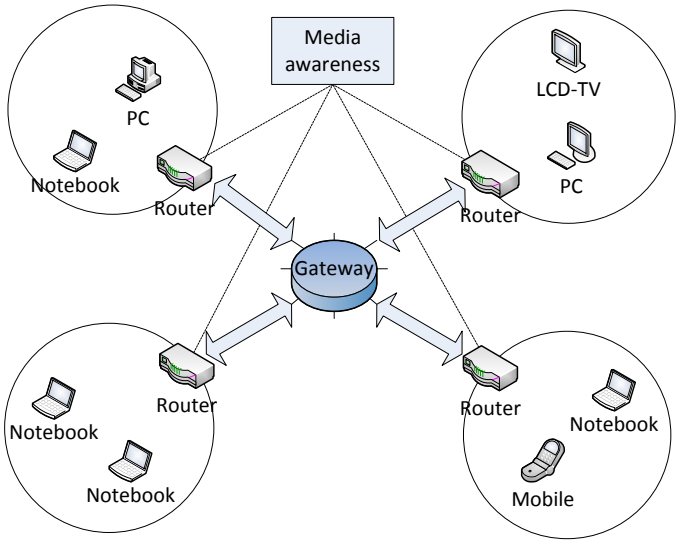


Figure 41: Media-aware network scenario (replotted for clarity).

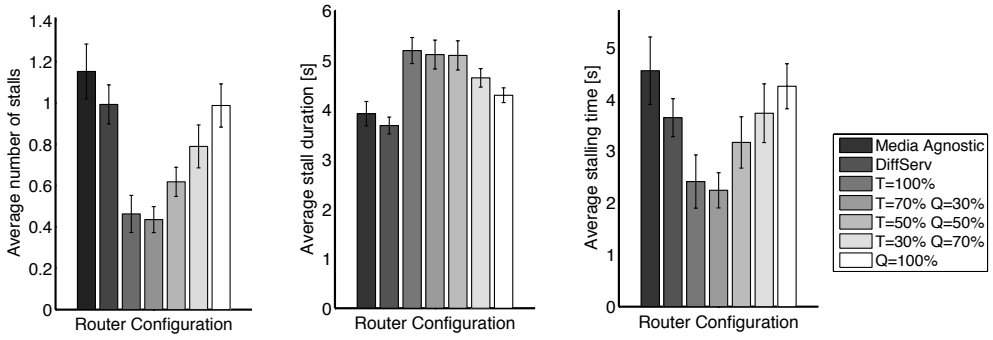
For calculating the priority according to Equation 5.6 we choose an unbiased weighting: $W_d = W_t = W_q = 0.33\bar{3}$. To assess the impact of media awareness as well as find the best router configuration we consider the following scenario: after streaming starts and peers start joining, we leave the system for 10 minutes to warm up. Then, we invoke an upload bottleneck of three Mbps for ten minutes. This means that, during the ten minutes, the router can only upload at three Mbps. Such a bottleneck can be due to limited resources allocated for this specific video stream or to cross traffic. For comparison, we run the system with media-agnostic and DiffServ routers, as explained above.

6.4.2 *Impact of Media-awareness*

In this experiment, we assess the impact of the media-aware network as well as find the best configurations. We test the media-aware router for a bottleneck of three Mbps and with different T and Q values, namely: T100, T70/Q30, T50/Q50, T30/Q70, Q100. For this experiment, the PQA adaptation interval was fixed to ten seconds, which represents a moderate value. We focus on the performance during the bottleneck, as then performance degradation takes place.

Session Quality.

We first present in Figure 42 the session quality during the ten minute bottleneck period. There we see the average number of stalls, average stall duration, and the total stalling duration, all calculated per peer.



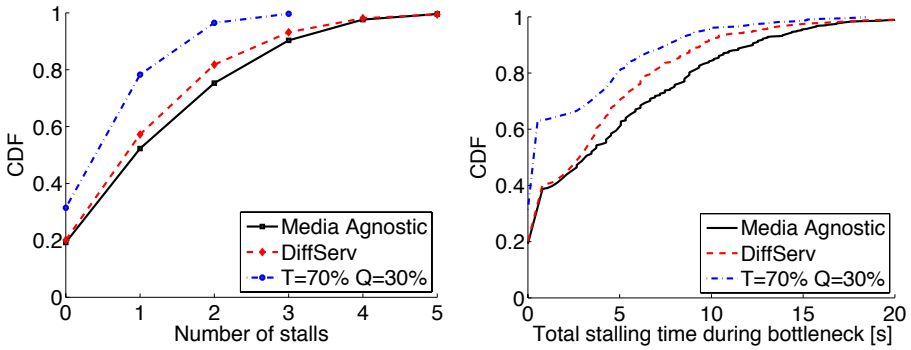
(a) Average number of stalls per peer (b) Average stall duration (c) Average total stalling time per peer

Figure 42: Session quality during the bottleneck.

Starting with Figure 42a and comparing the media-agnostic with the media-aware system, we can see that the average number of stalling events per peer is reduced from 1.15 down to 0.44. Although media awareness slightly increased the duration of stalling events (Figure 42b), it reduced the total stalling duration during the bottleneck as depicted in Figure 42c. That is, the total stalling duration is reduced from 4.6 down to 2.2 seconds per peer, resulting in 52% less stalling. Although DiffServ performed better than a media-agnostic network, it was still not able to compete with the media-aware solution.

Examining the different configurations of the media-aware router, we can see that the best results were achieved with T70/Q30, where there is just 2.2 seconds of total stalling during the bottleneck. This can be explained by the fact that a larger T means that sooner needed video blocks, especially in the buffer zone, are sent faster. It was still, nevertheless, important to include the SVC quality dimension with Q = 0.3 to make sure that peers do not have to wait long for SVC layers they have already requested.

Figure 43a and Figure 43b present the results as a Commutative Density Function (CDF). These graphs show the ratio of peers that had a specific number of stalls or stalling time. For T70/Q30 the highest number of stalls for any peer is 3. Furthermore, about 80% of the peers had either one or no stalls with T70/Q30, whereas for the media-agnostic and DiffServ routers, those peers had around 2 stalling events.



(a) CDF of number of stalls

(b) CDF of total stalling time

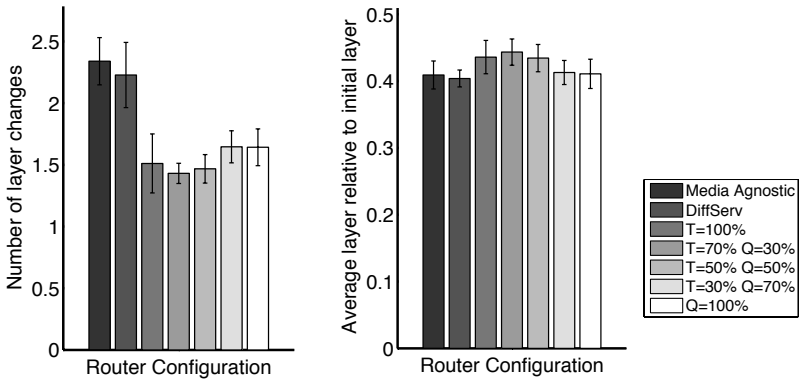
Figure 43: CDF of session quality during the bottleneck.

Video Quality

Now we look at the video quality during the bottleneck. The results are presented in Figure 44. We see that the number of layer changes during the bottleneck is affected by media awareness (Figure 44a), while the average relative quality is minimally affected (Figure 44b).

We can see that peers had the most layer changes with the media-agnostic and DiffServ routers. Evidently, those approaches have inflicted a larger number of stalls, which in turn caused the peers to adapt and reduce the requested layer, therefore, performing more layer changes. We can conclude that having media awareness leads to less quality changes because the peers are able to receive the requested quality. Again, the media-aware router with T70/Q30 yielded the best performance.

Concluding this experiment, we can say that media awareness based on a more weighted temporal priority (T70/Q30) has shown that video stalls and quality switches occur less often. Based on relevant user studies [113, 110], our approach, therefore, helps in enhancing the perceived video quality since the video playback is smoother and has fewer quality switches.



(a) Layer changes per peer (b) Average layer per peer

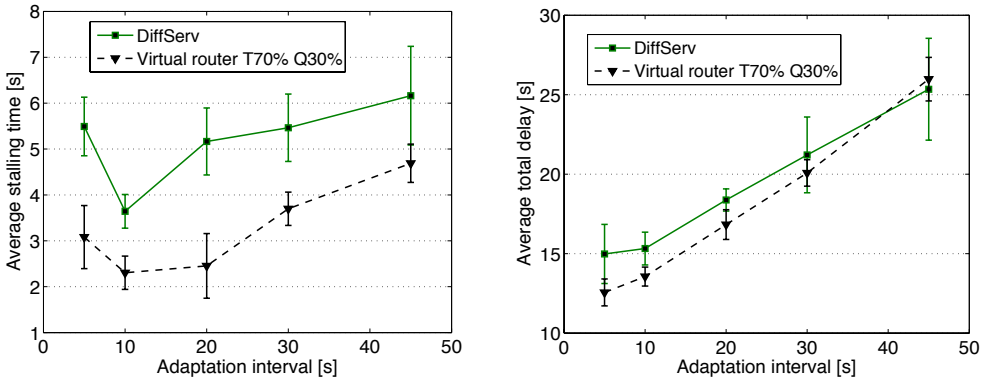
Figure 44: Video quality during the bottleneck.

6.4.3 *Adaptation at the Edge versus in the Network*

The second experiment deals with assessing the interdependencies between the PQA or adaptation interval and media awareness during a bottleneck. We check whether the adaptation interval depends on or affects the media-aware solution. Thus, we vary the adaptation interval choosing the values: 5, 10, 20, 30, and 45 seconds. We restrict our evaluation to the DiffServ and media-aware routers with T70/Q30.

Session Quality

Session quality with different adaptation intervals during the bottleneck and for the whole simulation are presented in Figure 45.



(a) Average stalling time per peer during the bottleneck (b) Average total delay per peer during simulation

Figure 45: Session quality with different adaptation intervals.

We can see that the media-aware approach outperforms DiffServ. During the bottleneck, the average stalling time could be drastically reduced for the different adaptation intervals. Additionally, we see that for the DiffServ approach during the bottleneck (Figure 45a), the relation between average stalling time

and adaptation interval is not predictable, which is the case once the media-aware solution is in place. Therefore, media awareness is quite crucial especially in applications where the system provider would change the adaptation interval during runtime, since this requires a more predictable relation.

Although the results for the total delay during the whole simulation (Figure 45b) do not show a huge performance gain, it is the performance during the bottleneck that has the highest impact on the overall performance of the system, and, therefore, is more relevant.

6.5 EVALUATING QOE-BASED QUALITY ADAPTATION

Now we present the evaluation of the third major contribution of the thesis. These are the QoE-aware quality adaptation mechanisms presented in Section 5.3.

The goals of our evaluation in this context are to investigate the impact of using objective QoE metrics on the dynamics of the P2P network. We are interested in assessing the impact of changing the server capacity on SVC video quality and system performance. Subsequently, we investigate how the different QoE-aware quality adaptation mechanisms compare in terms of performance. Finally, we want to understand the impact of changing the adaptation interval on the entire system.

6.5.1 *Simulation Setup*

The simulation setup used to evaluate the QoE-aware quality adaptation mechanisms is similar to that presented in Subsection 6.3.1. The main distinction is that here we make use of SVC file data along with the VQM measurements performed in [113]. Since the videos presented there, as we see later in this section, have rather high bit rates, we had to adjust the peer resources. These are presented next, followed by the SVC file properties. We use the same workload as for the previous simulations as presented in Subsection 6.3.1.

Peer Resources

In Table 6, the used peer resources are shown. The system comprises one tracker, 9 streaming servers, and 100 peers. The peers are divided according to their resources into three groups: slow, medium, and fast peers.

	Servers	Clients_{slow}	Clients_{medium}	Clients_{fast}
Number	9	20	50	30
Upload BW (Mbps)	25	0.5	3	8
Download BW (Mbps)	-	2	8	16
Player resolution (Pixels)	-	480x270	960x540	1216x684

Table 6: The used peer resources.

SVC File Properties

In this evaluation, we use the three SVC video files presented in [113]. The properties of these three files, namely: number of layers, resolution, frame rate, and total bit rates are presented in Table 7. These values are again derived from real SVC video sequences encoded using the reference implementation JSVM [80]. The three video sequences are freely available[2] and are denoted by *blue sky*, *crowd run*, and *park joy*. The QoE ratings used in the simulations were derived using the VQM metric, as detailed in [113]. The VQM ratings for the three sequences are presented in Appendix A.4.

Layer Index	SVC Level (d,t,q)	Dimension (Pixels)	Frame Rate (fps)	Total Bit Rate Blue Sky (KBps)	Total Bit Rate Crowd Run (KBps)	Total Bit Rate Park Joy (KBps)
0	0,0,0	480x270	1.875	99.502	83.343	89.190
1	0,1,0	480x270	3.75	116.962	124.848	134.219
2	0,2,0	480x270	7.5	133.664	174.872	180.994
3	0,3,0	480x270	15	148.941	216.091	197.295
4	0,4,0	480x270	30	164.336	235.070	202.510
5	1,0,0	640x360	1.875	237.859	196.833	195.647
6	1,1,0	640x360	3.75	278.904	295.807	298.488
7	1,2,0	640x360	7.5	319.317	423.468	418.088
8	1,3,0	640x360	15	356.181	534.859	473.345
9	1,4,0	640x360	30	394.226	591.817	487.983
10	2,0,0	960x540	1.875	344.678	326.427	338.438
11	2,1,0	960x540	3.75	405.544	486.092	528.358
12	2,2,0	960x540	7.5	467.042	688.186	778.516
13	2,3,0	960x540	15	521.065	881.488	956.511
14	2,4,0	960x540	30	574.822	1,003.290	1,004.883
15	3,0,0	1216x684	1.875	483.708	508.360	568.799
16	3,1,0	1216x684	3.75	571.487	749.941	890.658
17	3,2,0	1216x684	7.5	662.858	1,046.318	1,328.312
18	3,3,0	1216x684	15	742.194	1,333.625	1,703.705
19	3,4,0	1216x684	30	820.000	1,540.000	1,850.000

Table 7: The properties of the three SVC videos: *blue sky*, *crowd run*, and *park joy*. Note that 1 Kilo Byte per second (KBps) = 8 Kilo bits per second (Kbps).

Calculating Required Server Capacity

Server upload capacity is an important parameter for the provisioning of the system. Before we can evaluate our system, we have to derive reasonable values for the server upload capacity u_S based on the SVC video parameters and peer upload capacities. To do this, we refer to the model derived in Section 6.2.

We want to calculate the required upload capacity in the worst case when streaming the *Crowd Run* video. This implies that the 100 peers are all leechers. Therefore, the number of uploaders U is zero while the total number of

2 http://media.xiph.org/video/derf/ [Accessed Jan. 2012]

downloaders D is 100. The number of servers S is the nine. The average upload capacity $\bar{u}$ over the three peer sets is 4,000 Kbps. To calculate the average streamed bit rate $\bar{r}$, we have to consider each peer set separately. Each peer, in general, wants to saturate its download link but still stream SVC layers that fit its resources. Therefore, in the worst case for the server, $Clients_{slow}$ would stream layer index four (total bit rate: 235.07 KBps), $Clients_{medium}$ would stream layer index 13 (total bit rate: 881.488 KBps), and $Clients_{fast}$ would stream layer index 19 (total bit rate: 1,540 KBps).

To calculate the average streamed bit rate, we resort to the average over all peers calculated as

$$\bar{r} = \frac{\sum_{g \in G} r_g \cdot D_g}{\sum_{g \in G} D_g} = 7,626.864 \text{ Kbps}.$$

Assuming a default peer prefetching factor f of 1.0 and an peer upload utilization g of 0.8 and using Equation 6.10, the worst-case required capacity u_S of a single server is:

$$u_S = \frac{D \cdot (f \cdot \bar{r} - g \cdot \bar{u}) - \sum_{g \in G} U_g \cdot u_g}{S} = 49,187.38 \text{ Kbps}. \tag{6.11}$$

This server capacity is based on the assumption that there are no seeders in the system and that all peers are streaming the highest possible quality their resources can support. Therefore, we expect that the effective needed server capacity to be lower than that. To calculate this needed capacity, we evaluate the impact of different server capacities in the next section. Nonetheless, the above calculation helps in stressing the benefit of using QoE-aware quality adaptation mechanisms in a P2P VoD system since we later see that a server upload capacity of only 25 Mbps is enough for sustaining good performance for all the peers.

6.5.2 *Impact of Server Resources*

This first set of experiments deals with assessing the impact of changing server resources on the system performance. The right provisioning of server capacity is essential for content providers that want to provide good performance but still keep costs low.

In Section 6.2, we have derived how server capacities should be provisioned for the default system configuration. Those calculations are based on a worst-case scenario, meaning that each peer is assumed to be getting its highest possible quality. Thereby, no peer needs to perform quality adaptation. Nonetheless, the described model assumes a perfect interaction between the peers and abstracts much of the system's overhead. Because a content provider is interested in using P2P technology in order to reduce hosting costs, we focus on investigating a deficit scenario. In other words, server resources are intentionally set lower to assess performance when quality adaptation is needed.

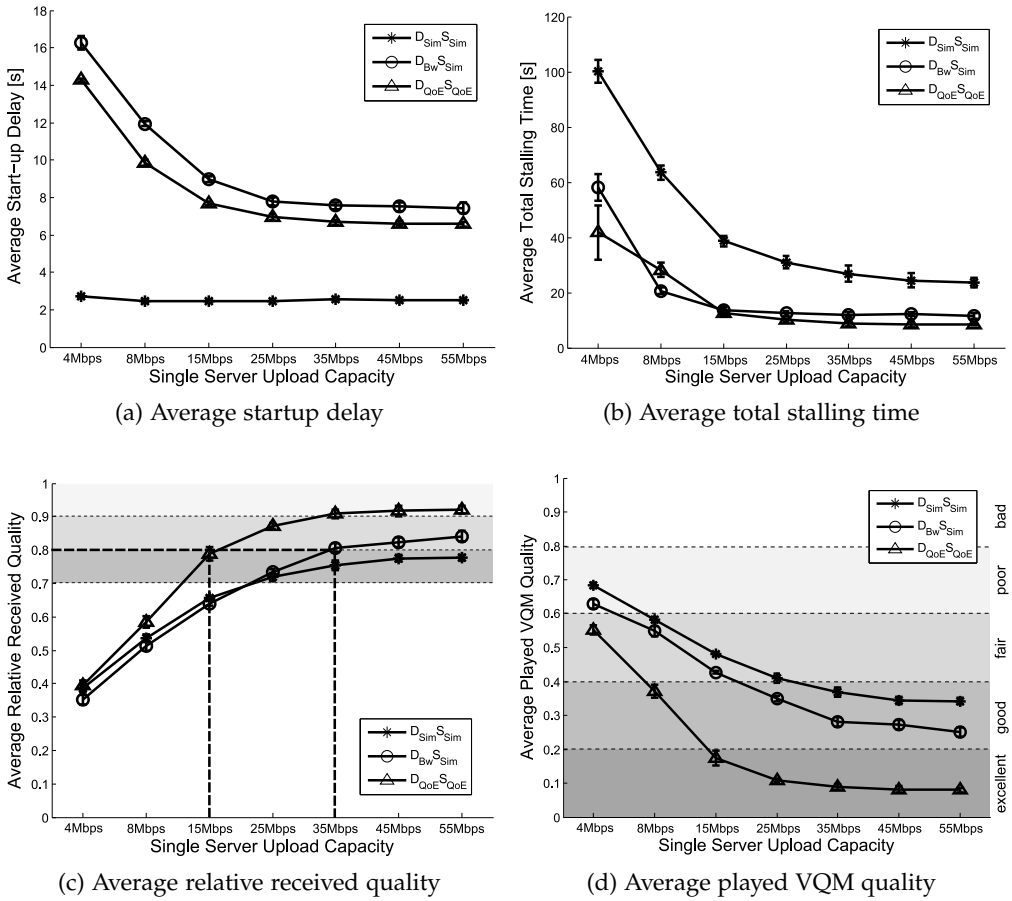
In this first scenario, we want to change the server capacity with two goals in mind. The first is to investigate the impact of server provisioning on the

performance. The second is to derive good values for the server capacity for our further investigations; in particular concerning the adaptation algorithms. Additionally, this would help content providers in understanding server provisioning in applications where the quality streamed by the different peers varies during playback.

For the evaluation, we vary the upload capacity of the servers from 4 Mbps to 55 Mbps, just below the upper limit of 61 Mbps derived above. For this set of simulations, we evaluate the session quality metrics: startup delay and average total stalling time. For the SVC video quality, the average relative received quality and the average played-out VQM quality are considered.

It is worth to remind the reader that the different mechanisms are named following the naming convention: D_xS_y, where D indicates the different quality decision algorithms, and S indicates the different quality switching algorithms. x and y are the different possibilities for both algorithms, as outlined in Subsection 5.3.6.

The results for this scenario are shown in Figure 46.



(a) Average startup delay

(b) Average total stalling time

(c) Average relative received quality

(d) Average played VQM quality

Figure 46: Comparison of the session and SVC video quality for changing system capacities.

Session Quality

Starting with the session quality, shown in the upper side of Figure 46, we observe that increasing the server capacity yields almost an exponential decrease in the startup delay for the $D_{Bw}S_{Sim}$ and $D_{QoE}S_{QoE}$ mechanisms. Only when starting from a 25 Mbps server capacity is the startup time smaller than eight seconds, a value commonly used as a target in Internet-based VoD systems [107]. Our $D_{QoE}S_{QoE}$ QoE-aware mechanism shows a clear advantage (of about one second) over the QoS-based $D_{Bw}S_{Sim}$ mechanism.

Starting from the point of 25 Mbps server capacity, we observe a saturation effect: adding more server capacity does not decrease the startup delay. This can be attributed to the time the buffer needs to be filled where this time is limited by the download speed of the peers and not by the server capacity.

The $D_{Sim}S_{Sim}$, which works by always first requesting the base layer, shows the fastest startup time of only two seconds. This can be explained by the fact that the SVC base layer is rather small, enabling quite a fast buffer filling and a faster startup of video playback. Additionally, the base layer is needed by all peers, making it easier to find uploaders to provide the base layer blocks.

Moving on to the total stalling time, the $D_{Sim}S_{Sim}$ mechanism does not show any advantage over our QoE-aware adaptation mechanism. This can be attributed to the frequent stalling events the $D_{Sim}S_{Sim}$ is inflicting. We again observe saturation in performance at a 25 Mbps server capacity. Starting from this saturation point, our mechanism outperforms the $D_{Bw}S_{Sim}$ mechanism by 50%.

Video Quality

For the video quality, the results are depicted in the lower part in Figure 46.

For the SVC video quality the following observation can be made. We first observe the intuitive effect of increasing server capacity, which enables the peers to better maintain the initially selected layer, i.e., achieved near 100% of the relative received quality. When a content provider is offering high server upload capacity of 55 Mbps, the relative received quality reaches 70% and 80% for the $D_{Sim}S_{Sim}$ and $D_{Bw}S_{Sim}$ mechanisms respectively.

Evidently, our QoE-aware adaptation mechanism performs the best. It is able to provide the peers with 90% of their initial layer starting from a 25 Mbps server capacity. Additionally, if the content provider provides only 15 Mbps capacity, the peers are still able to achieve an 80% relative received layer. On average, and for the different server capacities, the QoE-aware strategy performed 20% better than the QoS-based one.

From the content provider point of view, it is possible to save precious server resources while using our mechanism. If a target average received layer is 80%, server resources while using the $D_{Bw}S_{Sim}$ mechanism have to be 35 Mbps. When using our $D_{QoE}S_{Sim}$, only 15 Mbps of server capacity is required. In other words, the content provider can save up to 60% of server capacity while using our mechanism. This can be explained by the fact that since the peers are favoring layers with higher VQM ratings, those layers are better replicated,

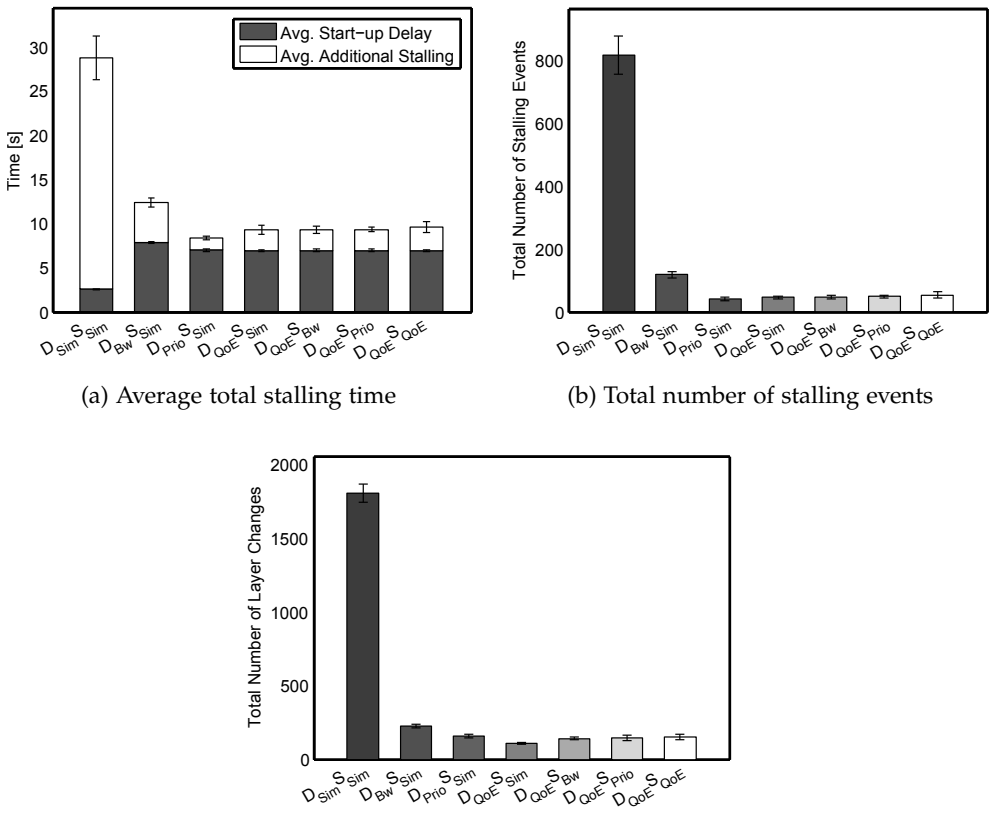
making it easier for the peers to get video data from other peers instead from the server.

We now go to the other metric that assesses the video quality, shown in Figure 46d. We can observe that starting from a 15 Mbps server capacity our $D_{QoE}S_{QoE}$ yields an average played VQM quality of 0.2. This maps to an *excellent* perceived quality on the MOS scale. Evidently, the $D_{QoE}S_{QoE}$ outperforms the other strategies and was the only strategy that was able to reach the *excellent* score on the MOS scale.

6.5.3 *Comparing the QoE-aware Algorithms*

We now come to the main evaluation scenario in which we compare the different adaptation mechanisms proposed in Section 5.3. All configurations described in Subsection 5.3.6 are simulated using the configurations presented in Subsection 6.5.1 and the video *crowd run*.

The simulated configurations in this experiment are: $D_{Sim}S_{Sim}$, $D_{Prio}S_{Sim}$, $D_{Bw}S_{Sim}$, $D_{QoE}S_{Sim}$, $D_{QoE}S_{Bw}$, $D_{QoE}S_{Prio}$, and $D_{QoE}S_{QoE}$. The results of the simulations are presented in Figure 47, Figure 48, and Figure 49.



(a) Average total stalling time

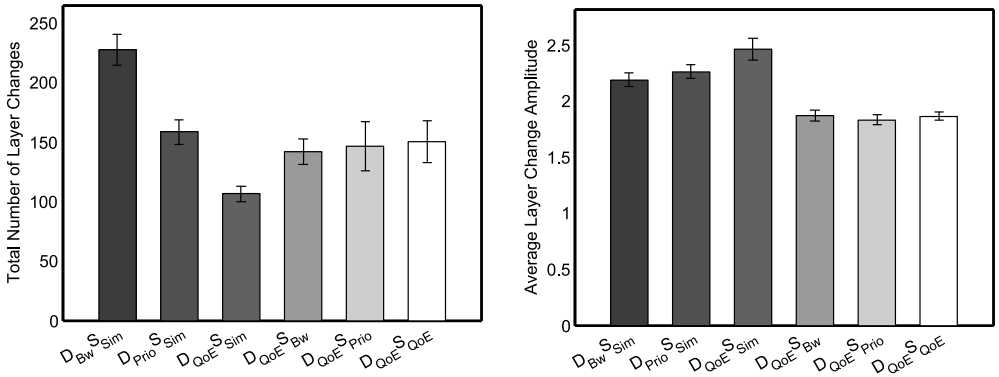
(b) Total number of stalling events

(c) Total number of layer changes

Figure 47: Comparison of the session and SVC video quality for different adaptation variants.

Regarding the session quality, the average total stalling time, presented in Figure 47a, shows improvements when using our QoE-aware adaptation mechanisms in comparison to the standard QoS-based one (i.e., $D_{Sim}S_{Sim}$). Accordingly, our algorithms were able to keep the number of stalling events lower (see Figure 47b). The frequent stalling events of the $D_{Sim}S_{Sim}$ mechanism has, although of very short startup time, created long total playback delays.

Regarding the video quality metrics, we observe that the $D_{Sim}S_{Sim}$ mechanism does not perform very well mainly because this mechanism is too slow in its adaptation. As shown in Figure 47c, the $D_{Sim}S_{Sim}$ mechanism had to perform much more layer changes, 1800 changes for all peers, in comparison to only 300 for the other mechanisms. These too frequent layer changes can be explained by the fact that this mechanism changes quality in steps and not according to the active resources.



(a) Total number of layer changes

(b) Average Layer Change Amplitude

Figure 48: Layer change characteristics, including the total number of SVC layer changes, and the average change amplitude.

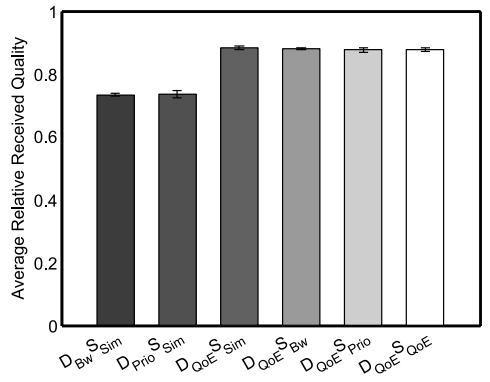
We have seen that the $D_{Sim}S_{Sim}$ mechanism, which is based on the stepwise switching of video quality, was not able to provide good performance, as the playback delay was too high. In order to better compare the other mechanisms, we excluded this mechanism for the next set of graphs.

In Figure 48, we show the number of layer changes as well as the average change amplitude. What we directly notice is that the $D_{QoE}S_{Sim}$ mechanism generates the least amount of layer changes. On the other hand, it also account for the highest average change amplitude of about 2.5 layers. Therefore, using no switching mechanism would generate the highest jumps. This can be reduced to just below 2 layers by using a layer switching algorithm.

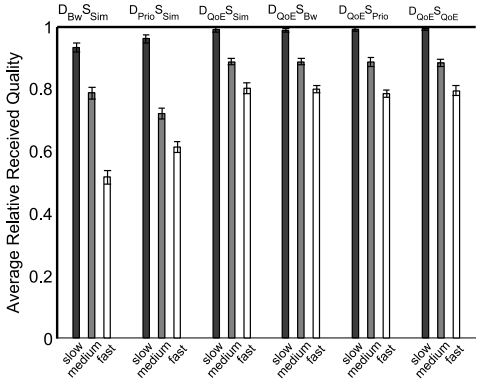
We notice that the three switching algorithms, S_{Bw}, S_{Prio}, and S_{QoE}, do not diverge from the performance point of view, they generated almost similar results. Nonetheless, our intention to use layer switching in order to decrease the amplitude of switching has been confirmed. The amplitude has been decreased on the expense of having a slightly higher number of layer changes. Recall that according to the work in [111], one should keep the amplitude and frequency of layer changes at a minimum. By applying our switching mechanisms, the amplitude could be reduced at the cost of an increased number of

layer changes. This shows that there is a tradeoff between the frequency and the amplitude of layer changes.

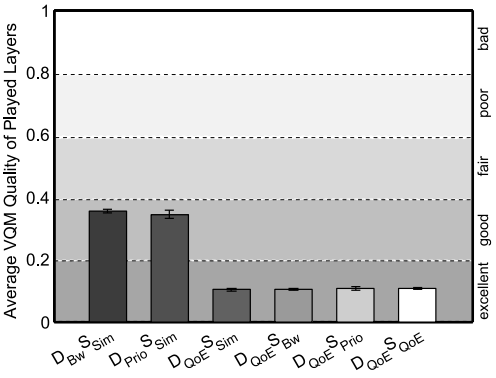
We now present the other SVC video quality metrics, the average relative received quality and VQM ratings, in order to get a better understanding on the performance of the quality adaptation mechanisms. These are shown in Figure 49, where we present the results for each mechanism (left side) and also results for each peer group (right side).



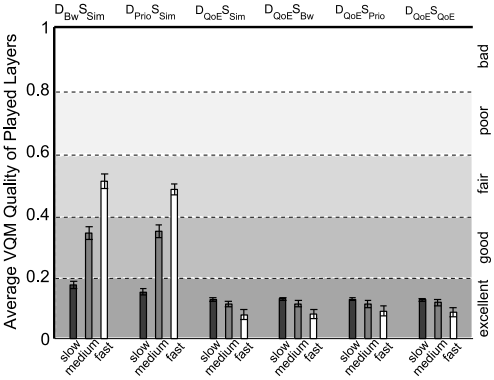
(a) Average relative received quality per peer



(b) Average relative received quality per peer group



(c) Average played VQM quality per peer



(d) Average played VQM quality per peer group

Figure 49: Comparison of the SVC video quality for the different adaptation mechanisms.

Starting with the average relative received quality, shown in Figure 49a and Figure 49b, we observe the following. The non-QoE-aware mechanisms provide an average received quality of only 70%, while the QoE-aware mechanisms reach about 90%. Second, for all mechanisms, the peers within the slow group are able to attain almost the highest possible relative received quality in comparison to the other groups. Keeping in mind that the calculation of the relative received quality depends on the layer selected by the IQA, a higher initial quality yields, in general, a smaller relative quality (see SVC video quality in Subsection 6.1.2). We notice the trend that the faster a peer is, the harder it gets for it to reach the high quality. This can be explained by the less abundant high quality blocks, where the fast peers have to resort to the server to get

them. Another reason for this trend is that the IQA layer of the fast group has a much higher bit rate than that of the slow group, which is difficult to sustain.

Going to the measured average VQM quality, shown in Figure 49c and Figure 49d, we get another view of the system.

First, we observe that the non-QoE-aware adaptation mechanisms generate a significantly worse quality than our QoE-aware mechanisms, which show an average VQM value of around 0.1 (with zero being the best quality over a scale from zero to one). Therefore, our QoE-aware mechanisms are able to provide a video with a better-estimated user perceived quality. Inspecting the results per peer group, presented in Figure 49d, we see that for the non-QoE-aware mechanisms, faster peers receive a worse estimated video quality than the slow peers. However, when using the QoE-aware adaptation mechanisms, the VQM is consistent and homogenous over the heterogeneous peers groups, where all peers achieved an *excellent* video quality on the MOS scale.

6.5.4 *Impact of the Adaptation Interval*

Now we assess the impact of the adaptation interval on the QoE aware adaptation mechanisms. As we have presented in Subsection 6.3.3, the adaptation interval is important especially for the frequency of layer changes.

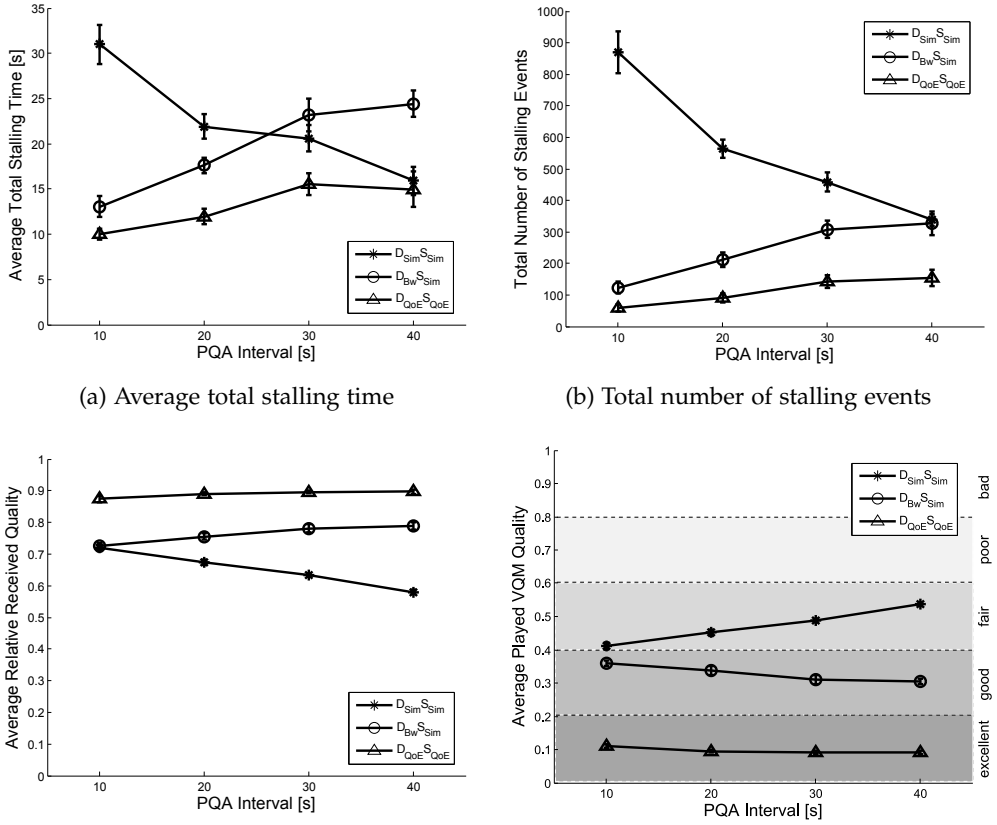
Now we limit our evaluation to four different adaptation intervals: 10, 20, 30, and 40 seconds. We compare the following algorithms: $D_{Sim}S_{Sim}$, $D_{Bw}S_{Sim}$, and $D_{QoE}S_{QoE}$.

In Figure 50, we show the selected metrics that represent the session and video quality.

Starting with the average total stalling time, presented in Figure 50a, and the total number of stalling events, presented in Figure 50b, we observe that the total stalling time and number of stalling events increases with an increased adaptation interval for the mechanism $D_{Bw}S_{Sim}$, and $D_{QoE}S_{QoE}$. However, for the $D_{Sim}S_{Sim}$ mechanism, the same metrics decrease with an increased adaptation interval. In order to explain this, we have a look at the number of stalling events and see that those are decreasing for an increased adaptation interval. This indicates a bad adaptation algorithm, since performance is improving when it is not used. Since more adaptations, in the case of $D_{Sim}S_{Sim}$, lead to more stalling events, much of the selected layers do not fit to the resources of the peers or the network. Therefore, the peers are not able to sustain those decisions and have to frequently switch the layers.

We now consider the SVC video quality metrics with focus on the average relative received quality and the played VQM quality.

We observe that both the relative received quality and the VQM quality degrade with an increased adaptation interval for the $D_{Sim}S_{Sim}$ mechanism. This in turn generates unacceptable ratings on the MOS scale. However, for both $D_{Bw}S_{Sim}$ and $D_{Bw}S_{Sim}$ mechanisms, the performance is not affected by the changed adaptation interval, where the two metrics are almost stable over the different adaptation intervals. Comparing all strategies, we directly see that our $D_{QoE}S_{QoE}$ mechanism substantially outperforms the other mechanisms for both the relative received quality and the VQM quality.



(a) Average total stalling time

(b) Total number of stalling events

(c) Average relative received quality

(d) Average played VQM quality

Figure 50: Comparison of session and SVC video quality for different adaptation intervals.

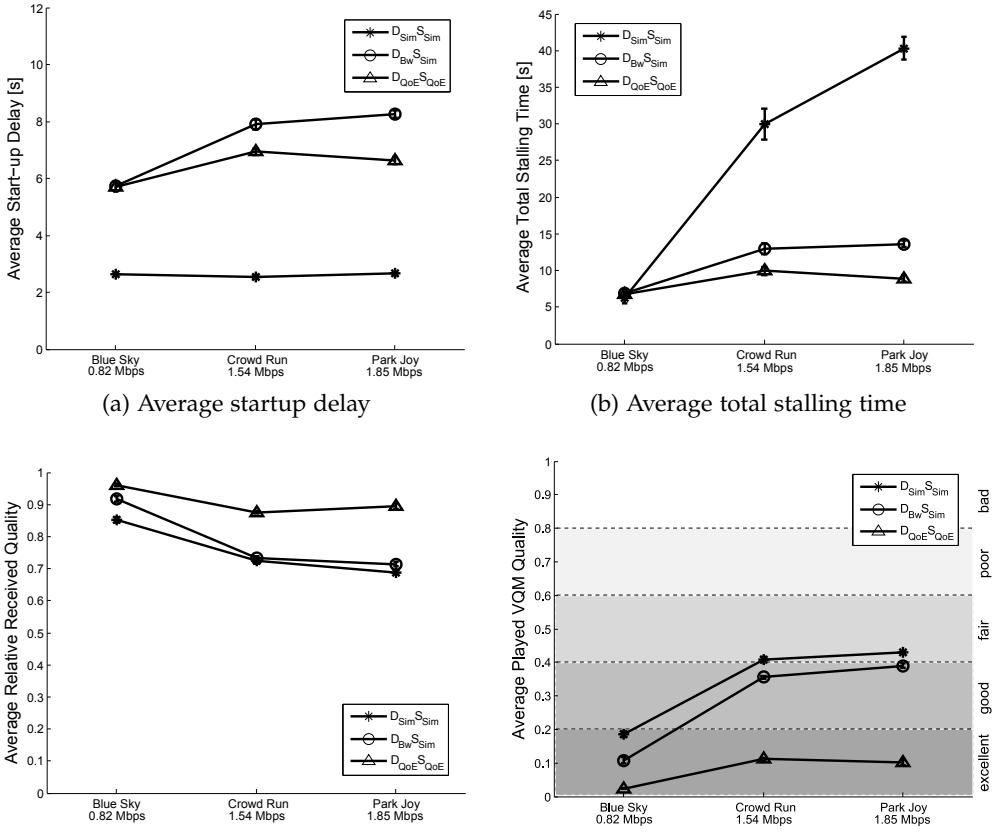
6.5.5 *Impact of SVC Video Bit Rate*

This last set of graphs deals with the impact of the SVC video bit rate on system performance. We use the three SVC video files, which were presented in Subsection 6.5.1. These videos, named *Blue Sky*, *Crowd Run*, and *Park Joy*, have the same number of spatial and temporal layers. However, they have differing bit rates due to different video content and motion levels. Evaluating the system with different video bit rates would greatly benefit VoD system providers when provisioning their system since our analysis helps in better understanding the interaction between video bit rates, quality adaptation, and performance.

We again restrict our evaluations to the following mechanisms. $D_{Sim}S_{Sim}$, $D_{Bw}S_{Sim}$, and $D_{QoE}S_{QoE}$. The server capacity is fixed to 25 Mbps.

In Figure 51, the most important session and SVC video quality metrics are presented.

Starting with Figure 51a, we observe that $D_{Sim}S_{Sim}$ inflicts the lowest start delay which is again attributed to the strategy to always stream the base layer at the beginning of the streaming session. Although the different SVC video files have different bit rates, the base layers almost are the same within the range from 80 to 100 KBps (See Subsection 6.5.1). Therefore, the $D_{Sim}S_{Sim}$



(a) Average startup delay

(b) Average total stalling time

(c) Average relative received quality

(d) Average played VQM quality

Figure 51: Comparison of session and SVC video quality for SVC videos with different bit rates.

the startup delay of this mechanism is constant over the different video bit rates.

Regarding the other two mechanisms, we see that the startup time is increasing with a higher bit rate video. This is because a higher bit rate video directly translates into more data that has to be downloaded. The same is noticed for the total stalling time (Figure 51b) Regarding the session quality, the $D_{QoE}S_{QoE}$ mechanism again outperforms the other two for all video bit rates, providing a total stalling time in the range of only six to ten seconds.

We now consider the SVC video quality depicted in Figure 51c and Figure 51d.

For the average relative received quality, both the $D_{Sim}S_{Sim}$ and $D_{Bw}S_{Sim}$ mechanisms inflict a decrease in relative quality. $D_{QoE}S_{QoE}$, however, starts with a value of about 95%, falling to about 88% for the second, and rising back to about 90% for the park joy video. Therefore, our algorithms are able to offer consistent relative received quality. The same tendencies are observable for the average VQM. Again our $D_{QoE}S_{QoE}$ mechanism was the only one able to sustain an *excellent* video quality over the MOS scale for the different video bit rates.

6.6 PROTOTYPE EVALUATION

We now present the evaluation of our system that we have performed in additional to the simulations. These evaluations are based on a prototype of the P2P VoD system along with the quality adaptation mechanisms. Other implementation issues that had to be addressed are detailed in Appendix A.2. The prototype was evaluated using the German Lab test bed [5], which is presented next. Later in this section, we present the major findings of our prototypical evaluation.

6.6.1 *The German-Lab Test Bed*

For the evaluation, we used the German Lab (GLab) test bed, which is funded by the *Federal Ministry of Education and Research* of Germany and was created in 2008. This test bed, which is similar to PlanetLab, is based in Germany with more than 150 homogenous physical nodes distributed over five major German cities (Berlin, Darmstadt, Karlsruhe, Munich, Kaiserslautern, and Würzburg).

In order to control and orchestrate experiments over the GLab test bed, we have developed a Central Monitoring and Control (CMC) software. The CMC, is used to configure GLab nodes, configure experiments, display real time information, as well as to collect log data at the end of each experiment. Also, the CMC initializes the peers, assigns them with resources, and makes them join and leave the network.

6.6.2 *Evaluation of Quality Adaptation*

The goal of our evaluation is to analyze the feasibility and performance of the quality adaptation mechanisms using SVC in a realistic environment. Additionally, we want to compare the results with the simulation study. Performance is expressed using the metrics presented in Section 6.1.

PEER RESOURCES. Similar to the simulative evaluation, we consider three peer sets with different resources. Peer resources are configured as shown in Table 8.

	Servers	Clients_{slow}	Clients_{medium}	Clients_{fast}
Number	1	5	5	5
Upload BW (Kbps)	12000	150	550	1175
Download BW (Kbps)	-	300	2210	4700
Player resolution (Pixels)	-	176×144	352×288	704×576

Table 8: The used peer resources for the prototype evaluation.

SVC FILE PROPERTIES AND WORKLOAD. We use the same SVC file properties that were used for the simulations (see Table 4), but with higher overall

video quality (see Table 11). This generates generally higher bit rates that fit well with real video bit rates.

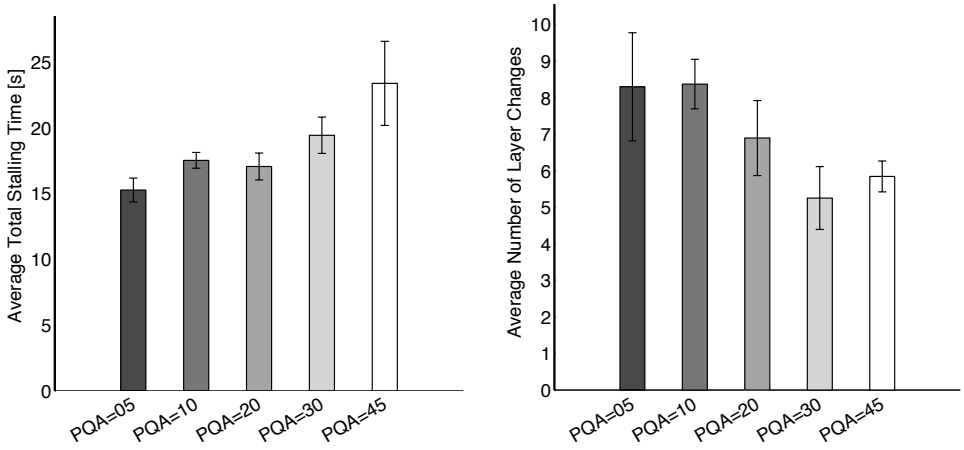
The workload parameters used in our prototype evaluation are the same as those used for the simulation study (see Table 5) except for using 15 peers as shown in Table 8.

Evaluation Results

We ran the prototype with the above mentioned configuration and workload over the G-Lab test bed. We repeat each experiment five times to rule out random effects, where 95th percentile confidence intervals are calculated.

In Figure 52 and Figure 53, we show the performance of the prototype in terms of the most important metrics.

Starting from Figure 52a, where we show the average total stalling time per peer for the different PQA intervals, we notice that the total stalling time starts from 15 seconds and slowly increases to 24 seconds for PQA of 45 seconds. Therefore, a larger PQA interval means that the system is slower in reacting to changes in the network resulting in more frequent video stalling.



(a) Number of layer changes

(b) Relative received layer

Figure 52: Prototype performance in terms of average stalling time (left side) and average number of layer changes per peer (right side) for different PQA adaptation intervals.

In Figure 52b, we present the number of layer changes per peer over different PQA intervals. This number is steadily decreasing from eight changes down to around five changes. Therefore, using a larger PQA interval, less adaptations are performed, leading to a lower number of layer changes.

In Figure 53, the average received quality is depicted for the different peer groups, i.e., peers that were classified as slow, medium and fast. We again evaluate the system with different PQA intervals. We first notice the performance is consistent over the different PQA intervals. Therefore, the system was successful in sustaining high relative qualities to the different peers. We additionally notice that while slow peers were able to get the full IQA layer (relative quality almost one), fast peers achieved only around 0.7 of the IQA layer. This seems

to indicate that high throughputs are difficult to sustain in real P2P networks. Thereby, although fast peers had enough download resources, they were not able to saturate their download capacity.

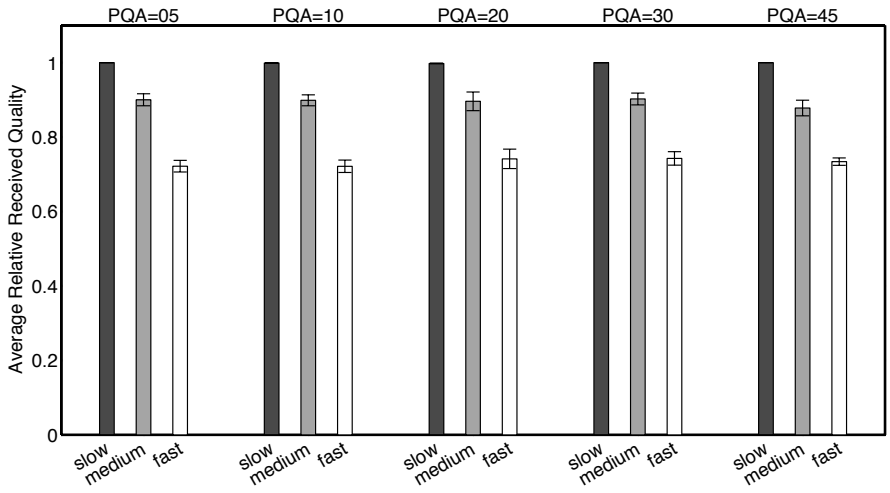


Figure 53: Prototype performance in terms of relative received quality with different PQA adaptation intervals. For each PQA value we show the performance of slow, medium, and fast peers separately.

We conclude from the above evaluations that, in general, the prototype performs as expected and provides good session and video qualities. It also demonstrates the feasibility of combining SVC and P2P VoD systems, in which peers with heterogeneous resources can co-exist in the same system and sustain the required data rates.

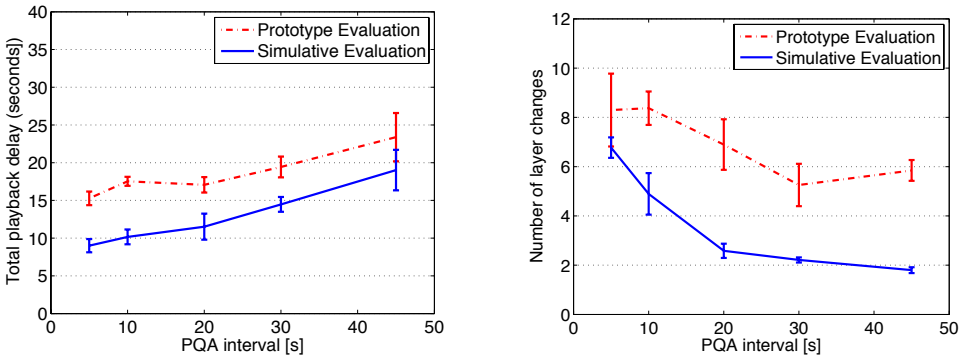
Next, we want to evaluate how well did the prototype fit our simulative evaluation.

Prototype Versus Simulation

We now assess how well the test bed study matches the simulation results. Therefore, in Figure 54 we show the test bed results along with the simulation results presented in Subsection 6.3.4. We restrict the presented metrics to the total playback delay (left side) and the number of layer changes (right side).

Regarding the session quality, we first notice the slightly higher playback delay of the prototype compared to the simulations. This can be attributed to two factors. The first is the slightly more tight resources of the system since the SVC file in consideration has a higher bit rate than that used for simulations. The second is the use of real networks that include additional effects such as: packet loss, delay, and communication overhead. Nonetheless, the same trend can be observed. With an increased PQA interval, the playback delay increases with the same rate.

Regarding the video quality presented through number of layer changes, we observe that the prototype behaves quite similar to the simulative study. With an increasing PQA interval, the number of layer changes decreases. Here the slightly higher number of layer changes for the prototype is attributed to the additional dynamics found in real system. Since the traffic of the G-Lab



(a) Total playback delay

(b) Number of layer changes

Figure 54: Performance of the quality adaptation mechanisms with different adaptation intervals using both test bed and simulation studies.

test bed goes over real networks, the system reacts to fluctuations in resources not found in simulations. Therefore, the system had to react more, thereby, performing more layer changes.

We conclude that the prototype and simulative study exhibit similar trends, thereby confirming the accuracy of our simulative evaluation.

6.7 SUMMARY

This chapter has provided an extensive simulative evaluation of our system with respect to different scenarios and evaluation metrics. We now provide a summary of the major achieved results.

In Section 6.3 we have evaluated our mechanisms for quality adaptation using SVC. We have compared our SVC-based quality-adaptive VoD system with one that does not use SVC. We were able to demonstrate the superiority of our system where the slow peers were able to be part of the system and retrieve the video, although with a lower video quality, but with much lower playback delays. We have shown how both the IQA and PQA were essential in achieving better session quality and more homogeneous performance across heterogeneous peers. Additionally, we have assessed the impact of changing the adaptation interval. We have seen that when the adaptation interval is larger, the peers become slower in reacting to network changes, thereby session quality degrades. On the other hand, the SVC video quality is improved, since the peers try to maintain a high video quality. Evidently, we have seen that those two metrics exhibit a tradeoff. Therefore, depending on the application area and which aspect is more important for the users, the adaptation interval can be adjusted accordingly to meet the given requirements.

In Section 6.4, we have evaluated our system for media-aware networking. We have compared the media-aware network with both a standard media-agnostic and a DiffServ networks. We were able to show that playback stalls could be reduced by up to 52% during congestion. Regarding the best router configuration, we have seen that a more weighted temporal priority (T70/Q30) yielded the best performance. Further, we have evaluated the interdependen-

cies between adapting at the edge and adapting in the network by using the media-aware system. We varied the adaptation interval at the edge and measured the performance. We found out that the advantage of using a media-aware network is consistent over the different adaptation intervals. Additionally, the relation between playback delay and adaptation interval is more predictable when having a media-aware network. Therefore, media awareness is quite crucial especially in applications where the system provider would change the adaptation interval during runtime.

In Section 6.5, we have evaluated our QoE-aware quality adaptation mechanisms. We have first demonstrated that if all peers favor the high QoE-rated SVC layers, those layers become better replicated. Thereby, the peers can better help in offloading servers. In this case, we have demonstrated that the content provider can save up to 60% of server costs while using our mechanism. Further, we have shown the superiority of the QoE-aware quality adaptation mechanism, the $D_{QoE}S_{QoE}$, which outperformed the other QoS-based strategies and was able to provide an *excellent* score on the MOS scale. In this regard, we have shown that a simple layer decision strategy based on choosing the base layer and increasing slowly did not perform as good as the QoE-based strategy. Subsequently, we evaluated the impact of QoE-based layer switching. The simple switching strategy ($D_{QoE}S_{Sim}$) generated the least amount of layer changes, but had the highest layer change amplitude. However, when using our QoE-aware switching mechanism, lower change amplitudes were achieved along with better overlay VQM quality. When varying the adaptation interval, the $D_{QoE}S_{QoE}$ mechanism showed consistent results and outperformed the other mechanisms for both relative received quality and VQM quality.

Finally, in Section 6.6, we have evaluated a prototype of the developed mechanisms using the German Lab test bed. We have seen that the prototype evaluations fit very well with our simulations. Additionally, we demonstrated the feasibility of the presented quality adaptation mechanisms in real networks and systems.

7 CONCLUSION

In this thesis, we have explored the potential, performance, and impact of quality adaptation using Scalable Video Coding (SVC) in Peer-to-Peer (P2P) Video-on-Demand (VoD) systems. In this last chapter, we summarize this thesis, highlighting the contributions of each chapter. Subsequently, we present the major contributions we have made in the context of a quality-adaptive VoD system. We then finish this chapter by presenting possible directions for future work.

7.1 THESIS SUMMARY

The use of P2P technologies to support VoD systems is very appealing since it enhances the capacity of the P2P network and, thus, either increases the achievable bit rate or allows the system to support more peers simultaneously. Nonetheless, existing P2P streaming systems suffer from major limitations when it comes to supporting the wide spectrum of end user devices and heterogeneous network resources. This thesis addresses this issue by improving the playback delay and video quality using *quality adaptation* mechanisms and SVC that are aware of the Quality of Experience (QoE).

Chapter 1 *"Introduction"* motivated the need for using alternative delivery architectures based on P2P techniques so that the cost for delivering multimedia content can be reduced. It also highlighted the problem of peer heterogeneity, and how quality adaptation is essential for such streaming systems. Furthermore, in Chapter 1, we presented our research goal, which is to improve video quality and playback performance of P2P VoD systems by using SVC and supporting peer heterogeneity.

Chapter 2 *"Background"* provided background information on major research areas required to understand the content of this thesis. We have provided an overview of the basics in multimedia delivery and how P2P can help in reducing costs for content providers. In particular, we highlighted the major pieces of information regarding P2P techniques, such as topologies and scheduling algorithms. Since a major contribution of this thesis is QoE-aware quality adaptation mechanisms, we provided information about prominent multi-layer codecs and how video quality and QoE can be assessed and measured.

Chapter 3 *"Related Work"* detailed pieces of work related to ours. Those were classified according to four categories: adaptive P2P VoD, P2P video streaming and SVC, media-aware networking, and objective QoE in video streaming. Different pieces of work in these four categories were presented, highlighting the major differences from the system developed in this thesis.

Chapter 4 *"Basic System Design"* presented the design of a P2P VoD system upon which we built our quality adaptation mechanisms. In this regard, this chapter detailed all algorithms and mechanisms we have developed in our basic system design. Additionally, this chapter explained the prefetching and upload strategies that helped to increase the benefit of using P2P techniques in VoD systems.

Chapter 5 *"Quality Adaptation in P2P Video-on-Demand"* presented the major contributions of this thesis. It was organized in three sections. The first section, quality adaptation using SVC, presented our two-stage quality adaptation algorithm used to adapt the video quality to various static and dynamic parameters of the peers and the system. This section additionally elaborated on how SVC videos have to be handled in the context of a P2P system, thereby, detailing various peer and block management techniques. The second section, media-aware networking, presented our design for a network that takes into account the priority of SVC videos when making resource allocation decisions. Subsequently, we developed a possible router application architecture that can better handle congestion by considering deadlines of video blocks and SVC file characteristics. The third section, QoE-aware quality adaptation, detailed a set of algorithms and techniques that extend QoS-based quality adaptation mechanisms to also consider QoE characteristics of SVC videos. There, we used a light-weight objective QoE metric, namely the Video Quality Metric (VQM), which helped peers in making better choices as to which SVC quality to choose. We divided the adaptation algorithms into two steps: layer decision and layer switching, where both use information about the QoE to have a smoother adaptation achieving better performance with less server capacity.

Chapter 6 *"System Evaluation"* evaluated the devised quality adaptation mechanisms using an extensive simulation study. In four sections, we have demonstrated the superiority of our system compared to the state of the art.

First, we have shown how our two-stage quality adaptation mechanism was able to drastically reduce playback delay compared to a non-adaptive VoD system. Peers with weak resources were thereby able to take part in the system while all peers had homogenous performance in terms of playback delay and continuous video playback. We have additionally identified an important tradeoff in SVC-based P2P VoD systems: session quality (playback delay and stalls) and SVC video quality (layer changes, received SVC layer) behave differently when the adaptation interval is changed. Nonetheless, a value of 10 seconds provided good session and SVC video quality. This value is regarded as an adaptation speed that fits well with the dynamics of P2P networks.

Second, using the developed media-aware solution, playback delay was reduced by 52% during congestion. In essence, we have seen that it is more important for routers to prioritize video blocks near their deadline rather than other blocks. The improved system performance was consistent over different adaptation speeds of the peers, and, therefore, can be used in scenarios where adaptation algorithms have to be adjusted by the system provider.

Third, our evaluations have shown that making the peers perform quality adaptation based on objective QoE metrics allowed content providers to re-

duce their bandwidth costs by up to 60%. Additionally, the developed mechanism provided the peers with better playback delays and SVC video quality compared to earlier work. We have shown that an approach based on starting always with the lowest quality and gradually increasing it [74] performs poorly as it does not adapt well to available system resources.

Finally, the test bed evaluation of the prototype implementation of the quality adaptation mechanisms has demonstrated the accuracy of our simulation models. Thereby, the same trends and tradeoffs were found for both prototype and simulation studies regarding the session and SVC video quality.

7.2 CONTRIBUTIONS

We now detail the major contributions of our work in this thesis.

- The first major contribution is a set of *Quality adaptation* mechanisms that leverages the power of SVC to improve the performance in terms of playback delay. The designed mechanisms have demonstrated that SVC is essential in matching the video quality to peer and system resources. The highest impact of using SVC was evident for the peers with weak resources as they were able to take part in the system and have a playback delay performance comparable to fast peers. We have shown the superiority of the designed system in comparison to non-quality-adaptive systems, such as the case when MPEG4/AVC or similar video codec is used [20, 38, 11, 75]. We have further identified major tradeoffs in the context of SVC-based VoD system that have to be considered by system providers.

- The next major contribution is a group of *Media-aware network management* techniques that make use of information about the video data to improve system performance during network congestion. The designed mechanisms do not cause any additional bandwidth costs. Rather, by simply using the priority of the transmitted blocks, better resource allocation decisions are taken. We have seen that traditional methods [73] were not able to provide the same performance as a full media-aware network that understands and reacts to SVC video traffic. The proposed mechanisms were able to reduce playback delay by 52% while showing a consistent improvement with different adaptation intervals at the peers.

- Another major contribution of this thesis is a set of *QoE-aware quality adaptation* mechanisms that make use of QoE ratings of the different SVC qualities to improve the performance perceived by the users. By making peers favor getting SVC layers with high QoE-ratings, those layers become better distributed and easier to find. This directly reduced the server traffic by 60%, as the peers were able to satisfy their own demand without resorting that often to the server to get video data. Additionally, performance at the peers improved in terms of session quality and SVC video quality compared to non-QoE adaptation mechanisms [13, 55, 70, 27, 66, 26, 103]. We use a light-weight objective QoE estimation method that does not re-

quire any bandwidth and processing resources at the peers. This avoids substantial overhead as reported in related work [105].

In addition to the contributions mentioned above, there are further findings and insights that we have gained throughout our work. Thereby, the following additional contributions were also made.

- A P2P VoD system that makes use of advanced prefetching and upload strategies has been designed and presented in Section 4.1. Using those mechanisms, peers with excess download bandwidth prefetch video pieces that are useful for other peers. Playback stalls were reduced by 50% compared to the state of the art used in other systems [96, 24]. Evaluation results for those mechanisms are presented in Appendix A.3.
- A P2P VoD system capacity model was developed that considers the major parameters in such systems while using both SVC (see Section 6.2) and non-SVC videos (see Subsection 4.2.1). This model is capable of calculating either the maximum number of supported peers or the required server capacity for given network resources. Such a model is essential for content providers to better provision their server resources, especially when the number of active peers is fluctuating.
- We have performed an in-depth classification of multi-layer video coding techniques (see Section 2.3). This classification helps to better understand the differences between various techniques that aim at dividing a video file into multiple layers that can be independently played.
- New metrics were developed that capture the performance of SVC-based VoD systems. Typical systems restrict their analysis to the playback delay and do not investigate other important metrics in the context of systems where the video quality is changing. We have, therefore, devised SVC video quality metrics that additionally take the SVC video performance into account (see Section 6.1).
- A prototype of the SVC-based P2P VoD system has been implemented (see Appendix A.2). The prototype employs the quality adaptation mechanisms explained in this thesis. Additionally, it encompasses the methods used to parse SVC videos and the required mechanisms.
- The prototype has been extensively evaluated in the German-Lab test bed [5]. We detailed the test bed and evaluation results in Section 6.6. There, we showed that the results fit very well with our simulations. Additionally, we demonstrated the feasibility of the presented quality adaptation mechanisms in real networks and systems.

7.3 OUTLOOK

This thesis has provided various contributions in the context of using SVC to enable quality adaptation in P2P VoD systems. The problem of quality adaptation stretches to different areas of the VoD system, where we have considered required mechanisms at the edge, in the core, and in relation to user QoE.

A P2P VoD system still relies on and interacts with many other system aspects that range from adaptive server allocation [79], to locality-awareness [16, 9], to incentive mechanisms [49, 10, 56], and to commercial considerations [3, 8]. Therefore, there is much potential for future work to assess and evaluate the impact and interactions of such mechanisms with the system designed in this thesis.

Regarding adaptive server allocation, an interesting question arises as to how can two major adaptation entities interact in one system. On one hand, an adaptive server usually reacts to a resource deficit by assigning more server resources. On the other hand, quality adaptation reacts to a resource deficit by reducing the streamed video quality. More work has to be done in order to investigate the complex interactions of these two adaptation techniques within one system to design appropriate mechanisms.

Locality awareness is a major trend when it comes to P2P overlay applications. Various projects [89, 9, 85] have investigated mechanisms to keep P2P traffic local within the same network. This is quite relevant to the media-aware solution we presented in this thesis. Further work in this area can deal with the tradeoff and performance of media awareness versus locality awareness.

Although this thesis assumed collaborative users in a closed system, the devised mechanisms can, with slight modification, be used in an open system. Here, SVC can be used to provide incentives for users to contribute resources. For example, the system would provide only the lowest video quality to free riders, i.e., users that do not contribute to others. The video quality may be increased when the peers upload more data. Therefore, users would have a concrete incentive to contribute to the system, since only then would they be able to get the highest video quality.

Further, we have seen that if the content provider wants to use minimal server resources, video stalling events are very probable. Here, more degrees of freedom can be added to the system. For example, during long stalling events, pre-loaded commercial spots could be displayed while the video is loading. Such commercial spots could also be related to the video content, creating a profitable platform for content providers.

7.4 FINAL REMARKS

This thesis has identified and highlighted the need for using SVC in the context of P2P VoD. Evidently, much of today's VoD systems rely on single-layer video codecs. Thereby, playback performance suffers greatly when peers with weak resources use the system. This thesis has proposed novel quality adaptation mechanisms using SVC that address the above-mentioned issue. While multi-layer video coding techniques have been proposed by system designers, much research regarding the required mechanisms that run in different areas in the system was required.

After presenting the different contributions and insights of this thesis, we are sure that the use of multi-layer video coding, such as SVC, is an important step towards the future generation of quality-adaptive P2P VoD systems.

[1] Dissecting the Gap Between Downloading and Streaming Video. http://www.ipsos-ideas.com/article.cfm?id=3804, Accessed January 2012.

[2] Multichannel News - YouTube May Lose $470 Million In 2009. http://www.multichannel.com/article/191223-YouTube_May_Lose_470_Million_In_2009_Analysts.php, Accessed January 2012.

[3] Octoshape. http://www.octoshape.com/, Accessed January 2012.

[4] Ookla - Net Index. http://www.netindex.com/, Accessed January 2012.

[5] The G-Lab Project. http://www.german-lab.de/, Accessed January 2012.

[6] Visual Networking Index: Forecast and Methodology. http://ciscovni.com/vni_forecast/advanced.html, Accessed January 2012.

[7] Youtube - Online Video Sharing. http://www.youtube.com/, Accessed January 2012.

[8] Zattoo - TV to Go. http://www.zattoo.com/, Accessed January 2012.

[9] V. Aggarwal and A. Feldmann. Locality-Aware P2P Query Search with ISP Collaboration. *Networks and Heterogeneous Media*, 3(2), 2008.

[10] N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu. Influences on Cooperation in BitTorrent Communities. In *ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems (P2PECON)*, 2005.

[11] S. Annapureddy, C. Gkantsidis, P. Rodriguez, and L. Providing Video-on-Demand using Peer-to-Peer Networks. In *Proc. of the Internet Protocol TeleVision (IPTV) workshop*, 2006.

[12] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. Rodriguez. Is High-quality VoD Feasible Using P2P Swarming? In *16th ACM International Conference on World Wide Web (WWW)*, 2007.

[13] P. Baccichet, T. Schierl, T. Wiegand, and B. Girod. Low-delay Peer-to-Peer Streaming using Scalable Video Coding. In *16th International Packet Video Workshop (PV)*, 2007.

[14] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. Analyzing and Improving a BitTorrent Networks Performance Mechanisms. In *25th IEEE International Conference on Computer Communications INFOCOM*, 2006.

[15] M. Bieberich. IP Core Network Intelligence: Essential Components and Drivers. *Cisco, Yankee Group*, 2007.

[16] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang. Improving Traffic Locality in BitTorrent via Biased Neighbor Selection. In *Proc. of the ICDCS Conference*, 2006.

[17] Z. Bozakov. An Open Router Virtualization Framework using a Programmable Forwarding Plane. In *SIGCOMM*, 2010.

[18] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth Multicast in Cooperative Environments. In *Symposium on Operating Systems Principles (SOSP)*. ACM, 2003.

[19] J. Chakareski, S. Han, and B. Girod. Layered Coding vs. Multiple Descriptions for Video Streaming over Multiple Paths. *Multimedia Systems*, 10:275–285, 2005.

[20] B. Cheng, L. Stein, H. Jin, and Z. Zhang. A Framework for Lazy Replication in P2P VoD. In *18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV, 2008.

[21] H. Chi, Q. Zhang, J. Jia, and X. Shen. Efficient Search and Scheduling in P2P-based Media-on-Demand Streaming Service. *IEEE Journal on Selected Areas in Communications*, 25(1):119–130, 2007.

[22] V. R. M. K. L. Chikkerur, S. Sundaram. Objective Video Quality Assessment Methods: A Classification, Review, and Performance Comparison. *IEEE Transaction on Broadcasting*, 2011.

[23] Y. H. Chu, G. Rao, and H. Zhang. A Case for End System Multicast. In *SIGMETRICS*, 2000.

[24] B. Cohen. Incentives Build Robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[25] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, editors. *Introduction to Algorithms*. The MIT Press, 2 edition, 2001.

[26] Y. Cui and K. Nahrstedt. Layered Peer-to-Peer Streaming. In *13th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2003.

[27] Y. Ding, J. Liu, D. Wang, and H. Jiang. Peer-to-Peer Video-on-Demand with Scalable Video Coding. *Computer Communications*, 33:1589–1597, 2010.

[28] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu. Characterizing Residential Broadband Networks. In *7th ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2007.

[29] U. Engelke and H.-J. Zepernick. Perceptual-based Quality Metrics for Image and Video Services: A Survey. In *3rd EuroNGI Conference on Next Generation Internet Networks*, 2007.

[30] M. Fidler. Differentiated Services Based Priority Dropping and its Application to Layered Video Streams. In *IFIP Networking*, 2002.

[31] M. Fiedler, T. Hoßfeld, and P. Tran-Gia. A Generic Quantitative Relationship between Quality of Experience and Quality of Service. *IEEE Network Special Issue on Improving QoE for Network Services*, 24(2):36 – 41, 2010.

[32] R. Fortuna, E. Leonardi, M. Mellia, M. Meo, and S. Traverso. QoE in Pull Based P2P-TV Systems: Overlay Topology Design Tradeoffs. In *Proceedings of the 10th International Conference on Peer-to-Peer Computing (P2P)*, 2010.

[33] J. Ghoshal, L. Xu, B. Ramamurthy, and M. Wang. Network Architectures for Live Peer-to-Peer Media Streaming. Technical report, Department of Computer Science and Engineering, University of Nebraska-Lincoln, 2007.

[34] D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O. Jr. Overcast: Reliable Multicasting with an Overlay Network. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2000.

[35] C. Griwodz. *Wide-area True Video-on-Demand by a Decentralized Cache-based Distribution Infrastructure*. PhD thesis, Technische Universität Darmstadt, 2000.

[36] B. Guedes, R. Pereira, T. Vazao, and A. Varela. Simple Media-Aware Packet Discard Algorithms. *International Conference on Information Networking*, 2009.

[37] J. Gustafsson, G. Heikkila, and M. Pettersson. Measuring Multimedia Quality in Mobile Networks with an Objective Parametric Model. In *15th IEEE International Conference on Image Processing (ICIP)*, 2008.

[38] M. Hefeeda, B. Bhargava, and D. Yau. A Hybrid Architecture for Cost-effective On-demand Media Streaming. *Computer Networks*, 44(3):353–382, 2004.

[39] T. Hossfeld, P. Tran-Gia, and M. Fiedler. Quantification of Quality of Experience for Edge-Based Applications. In *20th International Teletraffic Congress (ITC)*, 2007.

[40] C. Huang, J. Li, and K. W. Ross. Can Internet Video-on-Demand be Profitable? In *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2007.

[41] International Telecommunication Union. Recommendation ITU-T P.910: Subjective Video Quality Assessment Methods for Multimedia Applications. http://www.itu.int/rec/T-REC-P.910, 2008. [Accessed January 2012].

[42] International Telecommunication Union. Recommendation ITU-R BT.500: Methodology for the Subjective Assessment of the Quality of Television Pictures. http://www.itu.int/rec/R-REC-BT.500, 2009. [Accessed January 2012].

[43] International Telecommunication Union. Recommendation ITU-T H.264: Advanced Video Coding for Generic Audiovisual Services. http://www.itu.int/rec/T-REC-H.264, 2010. [Accessed January 2012].

[44] R. Iqbal and S. Shirmohammadi. DAg-stream: Distributed Video Adaptation for Overlay Streaming to Heterogeneous Devices. *Peer-to-Peer Networking and Applications*, 2009.

[45] X. Jiang, Y. Dong, D. Xu, and B. Bhargava. GnuStream: A P2P Media Streaming System Prototype. In *International Conference on Multimedia and Expo (ICME)*. IEEE, 2003.

[46] D. Jurca, J. Chakareski, J.-P. Wagner, and P. Frossard. Enabling Adaptive Video Streaming in P2P Systems. *IEEE Communications Magazine*, 45(6):108–114, 2007.

[47] D. Jurca, S. Petrovic, and P. Frossard. Media aware routing in large scale networks with overlay. In *Proc. IEEE ICME*, 2005.

[48] J. Karlsson and H. Li. P2P Video Multicast for Wireless Mobile Clients. In *The Fourth International Conference on MobiSys*, 2006.

[49] S. Kaune, K. Pussep, G. Tyson, A. Mauthe, and R. Steinmetz. Cooperation in P2P Systems through Sociological Incentive Patterns. In *3rd International Workshop on Self-Organizing Systems (IWSOS)*, 2008.

[50] S. Kaune, G. Tyson, K. Pussep, A. Mauthe, and R. Steinmetz. The Seeder Promotion Problem: Measurements, Analysis and Solution Space. In *IEEE International Conference on Computer Communications and Networks (ICCCN)*, 2010.

[51] C. S. Kim, H. Sohn, W. D. Neve, and Y. M. Ro. An Objective Perceptual Quality-Based ADTE for Adapting Mobile SVC Video Content. *IEICE Transactions on Information & Systems*, E92-D(1):93–96, 2009.

[52] S. Kopf, F. Lampi, T. King, and W. Effelsberg. Automatic Scaling and Cropping of Videos for Devices with limited Screen Resolution. In *Proceedings of the 14th Annual ACM International Conference on Multimedia*, 2006.

[53] R. Kumar, Y. Liu, and K. W. Ross. Stochastic Fluid Theory for P2P Streaming Systems. In *26th IEEE International Conference on Computer Communications (INFOCOM)*, 2007.

[54] J. Lee, F. D. Simone, and T. Ebrahimi. Subjective Quality Evaluation via Paired Comparison: Application to Scalable Video Coding. *IEEE Transactions on Multimedia*, 5(99):1, 2011.

[55] T.-C. Lee, P.-C. Liu, W.-L. Shyu, and C.-Y. Wu. Live Video Streaming Using P2P and SVC. In *11th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS)*, 2008.

[56] D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee. BitTorrent is an Auction: Analyzing and Improving BitTorrent's Incentives. In *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2008.

[57] R. Liao, S. Yu, and J. Yu. Synchronization-based Overlay Construction for Resilient P2P Streaming. In *World Congress on Intelligent Control and Automation (WCICA)*, 2008.

[58] J. Liu, B. Li, and Y.-Q. Zhang. Adaptive Video Multicast over the Internet. *IEEE MultiMedia*, 10(1), 2003.

[59] J. Liu, S. G. Rao, B. Li, and H. Zhang. Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast. *Proceedings of the IEEE*, 96(1):11–24, 2008.

[60] Y. Liu, Y. Guo, and C. Liang. A Survey on Peer-to-peer Video Streaming Systems. *Peer-to-Peer Networking and Applications*, 1(1):18–28, 2008.

[61] Z. Ma and Y. Wang. Complexity Modeling of Scalable Video Decoding. In *IEEE ICASSP 2008*, 2008.

[62] N. Magharei and R. Rejaie. Adaptive Receiver-driven Streaming from Multiple Senders. *Multimedia Systems*, 11:1–18, 2006.

[63] N. Magharei and R. Rejaie. PRIME: Peer-to-Peer receiver-driven mesh-based Streaming. In *IEEE INFOCOM*, 2007.

[64] N. Magharei, R. Rejaie, and Y. Guo. Mesh or Multiple-tree: A Comparative Study of Live P2P Streaming Approaches. In *IEEE INFOCOM*, 2007.

[65] V. Menkovski, G. Exarchakos, and A. Liotta. Machine Learning Approach for Quality of Experience Aware Networks. In *2nd International Conference on Intelligent Networking and Collaborative Systems 2010 (INCOS)*, 2010.

[66] K. Mokhtarian and M. Hefeeda. Analysis of Peer-assisted Video-on-Demand Systems with Scalable Video Streams. In *1rst ACM Conference on Multimedia Systems (MMSys)*, 2010.

[67] T. Mori, N. Kamiyama, and S. Harada. Improving Deployability of Peer-assisted CDN Platform with Incentive. In *IEEE Global Telecommunications Conference (GLOBECOM)*, 2010.

[68] M. Mu. An Interview with Video Quality Experts. *ACM SIGMultimedia Records*, 1:4–13, 2009.

[69] M. Mu, E. Cerqueira, F. Boavida, and A. Mauthe. Quality of Experience Management Framework for Real-time Multimedia Applications. *International Journal of Internet Protocol Technology (IJIPT)*, 2009.

[70] M. Mushtaq and T. Ahmed. Smooth Video Delivery for SVC based Media Streaming over P2P Networks. In *5th IEEE Consumer Communications and Networking Conference (CCNC)*, 2008.

[71] A. T. Nguyen, B. Li, and F. Eliassen. Chameleon: Adaptive Peer-to-Peer Streaming with Network Coding. In *IEEE INFOCOM*, 2010.

[72] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. Halvorsen. Flicker Effects in Adaptive Video Streaming to Handheld Devices. In *Proceedings of the ACM International Multimedia Conference (ACM MM)*, 2011.

[73] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, RFC, 1998.

[74] S. Oechsner, T. Zinner, J. Prokopetz, and T. Hoßfeld. Supporting Scalable Video Codecs in a P2P Video-on-Demand Streaming System. In *21th ITC Specialist Seminar on Multimedia Applications - Traffic, Performance and QoE*, 2010.

[75] V. N. Padmanabhan, H. J. Wang, and P. A. Chou. Supporting Heterogeneity and Congestion Control in Peer-to-Peer Multicast Streaming. In *International Workshop on Peer-to-Peer Systems (IPTPS)*, 2004.

[76] M. Piatek and R. Yang. Contracts: Practical Contribution Incentives for P2P Live Streaming. In *NSDI*, 2010.

[77] M. Pinson and S. Wolf. A New Standardized Method for Objectively Measuring Video Quality. *IEEE Transactions on Broadcasting*, 50(3):312–322, 2004.

[78] T. Plagemann, V. Goebel, C. Griwodz, P. Halverson, L. Mathy, N. Race, and M. Zink. Towards Scalable and Affordable Content Distribution Services. In *7th IEEE International Conference on Telecommunications (ConTEL)*, 2003.

[79] K. Pussep, O. Abboud, F. Gerlach, R. Steinmetz, and T. Strufe. Adaptive Server Allocation for Peer-assisted VoD. In *International Parallel and Distributed Processing Symposium (IPDPS)*, 2010.

[80] J. Reichel, H. Schwarz, and M. Wien. Joint Scalable Video Model JSVM-9. Doc. JVT-V202, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, 2007.

[81] R. Rejaie and A. Ortega. PALS: Peer-to-Peer Adaptive Layered Streaming. In *ACM NOSSDAV*, 2003.

[82] I. Richardson. *Video Codec Design: Developing Image and Video Compression Systems*. Wiley, 2002.

[83] 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(9):1103–1120, 2007.

[84] H. Schwarz and M. Wien. The Scalable Video Coding Extension of the H.264/AVC Standard [Standards in a Nutshell]. *IEEE Signal Processing Magazine*, 25(2):135–141, 2008.

[85] J. Seedorf, S. Kiesel, and M. Stiemerling. Traffic Localization for P2P-applications: The ALTO Approach. In *9th IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2009.

[86] P. Seeling, F. Fitzek, G. Ertli, A. Pulipaka, and M. Reisslein. Video Network Traffic and Quality Comparison of VP8 and H.264 SVC. In *3rd ACM Workshop on Mobile Video Delivery*, 2011.

[87] E. Setton, P. Baccichet, and B. Girod. Peer-to-Peer Live Multicast: A Video Perspective. *Proceedings of the IEEE*, 96(1):25–38, 2008.

[88] R. Steinmetz and K. Wehrle. *Peer-to-Peer Systems and Applications*. Springer, 2004.

[89] B. Stiller and W. Kellerer. Simple Economic Management Approaches of Overlay Traffic in Heterogeneous Internet Topologies (SmoothIT). 2008.

[90] D. Stingl, C. Groß, J. Rückert, L. Nobach, A. Kovacevic, and R. Steinmetz. PeerfactSim.KOM: A Simulation Framework for Peer-to-Peer Systems. In *Proceedings of the 2011 International Conference on High Performance Computing and Simulation (HPCS)*, 2011.

[91] T. Strufe, G. Schäfer, and A. Chang. BCBS: An Efficient Load Balancing Strategy for Cooperative Overlay Live-Streaming. In *Proceeding of the IEEE International Conference on Communications (ICC)*, 2006.

[92] P. Tran-Gia, T. Hossfeld, M. Menth, and R. Pries. Emerging Issues in Current Future Internet Design. *e&i Elektrotechnik und Informationstechnik*, 126(7):241–249, 2009.

[93] K. Tutschku and P. Tran-Gia. Traffic Characteristics and Performance Evaluation of Peer-to-Peer Systems. In R. Steinmetz and K. Wehrle, editors, *Peer-to-Peer-Systems and Applications*. Springer, 2005.

[94] V. Venkataraman, K. Yoshida, and P. Francis. Chunkyspread: Heterogeneous unstructured tree-based peer to peer Multicast. In *IEEE International Conference on Network Protocols (ICNP)*, 2006.

[95] Video Quality Experts Group. Final Report from the Video Quality Experts Group on the Validation of Objective Models of Video Quality Assessment, FR-TV Phase II. `http://www.its.bldrdoc.gov/vqeg/projects/frtv_phaseII/`, 2003. [Accessed January 2012].

[96] A. Vlavianos, M. Iliofotou, and M. Faloutsos. BiToS: Enhancing BitTorrent for Supporting Streaming Applications. In *9th IEEE Global Internet Symposium 2006*, 2006.

[97] L. Vu, I. Gupta, K. Nahrstedt, and J. Liang. Understanding Overlay Characteristics of a Large-scale Peer-to-Peer IPTV System. *ACM Transactions on Multimedia Computing, Communications and Applications*, 6, 2010.

[98] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[99] M. Welzl, M. Mühlhaüser, and L. Franzens. Scalability and Quality of Service: A Trade-off? *IEEE Communications Magazine*, 41(6):32–36, 2003.

[100] S. Winkler. *Digital Video Quality: Vision Models and Metrics*. Wiley, 1 edition, 2005.

[101] S. Winkler. Video Quality and Beyond. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, 2007.

[102] S. Wolf and M. Pinson. Application of the NTIA General Video Quality Metric (VQM) to HDTV Quality Monitoring. In *Proceedings of the Third International Workshop on Video Processing and Quality Metrics for Consumer Electronics (VPQM)*, 2007.

[103] X. Xiao, Y. Shi, Y. Gao, and Q. Zhang. LayerP2P: A New Data Scheduling Approach for Layered Streaming in Heterogeneous Networks. In *INFOCOM*, 2009.

[104] Y. T. Yu and S. R. Tong. Adaptive Transmission Control Protocol-trunking Flow Control Mechanism for Supporting Proxy-assisted Video on Demand System. *International Journal of Communication Systems*, 2011.

[105] G. Zhai, J. Cai, W. Lin, X. Yang, and W. Zhang. Three Dimensional Scalable Video Adaptation via User-End Perceptual Quality Assessment. *IEEE Transactions on Broadcasting*, 54(3):719–727, 2008.

[106] X. Zhang, J. Liu, B. Li, and T. P. Yum. CoolStreaming/DONet: a Data-driven Overlay Network for Peer-to-Peer Live Media Streaming. In *INFOCOM*. IEEE, 2005.

[107] Y. Zheng, D. Huang, W. Zhu, X. Zhang, Y. Chen, and C. Chen. A Measurement Study of P2P VoD System. In *International Conference on Research Challenges in Computer Science (ICRCCS)*, 2009.

[108] X. Zhu, P. Agrawal, J. Singh, T. Alpcan, and B. Girod. Distributed Rate Allocation Policies for Multi-homed Video Streaming Over Heterogeneous Access Networks. *IEEE Transactions on Media*, 2009.

[109] M. Zink. *Scalable Internet Video-on-Demand Systems*. PhD thesis, Technische Universität Darmstadt, 2003.

[110] M. Zink, O. Kuenzel, J. Schmitt, and R. Steinmetz. Subjective Impression of Variations in Layer Encoded Videos. In *International Workshop on Quality of Service*, 2003.

[111] M. Zink, O. Künzel, J. Schmitt, and R. Steinmetz. Subjective Impression of Variations in Layer Encoded Videos. In *Proceedings of the 11th International Conference on Quality of Service (IWQoS)*, 2003.

[112] T. Zinner, O. Abboud, O. Hohlfeld, T. Hoßfeld, and P. Tran-Gia. Towards QoE Management for Scalable Video Streaming. In *21th ITC Specialist Seminar on Multimedia Applications - Traffic, Performance and QoE*, 2010.

[113] T. Zinner, O. Hohlfeld, O. Abboud, and T. Hoßfeld. Impact of Frame Rate and Resolution on Objective QoE Metrics. In *Second International Workshop on Quality of Multimedia Experience (QoMEX)*, 2010.

A

APPENDIX

In this chapter, we provide further details that are part of the contributions of this thesis. Thereby, we present how Scalable Video Coding (SVC) videos are parsed to be used in a Peer-to-Peer (P2P) Video-on-Demand (VoD) system. Subsequently, the prototypical implementation of our system is presented highlighting the major challenges while implementing the VoD system beyond simulations. Further, some additional simulations results regarding the developed prefetching and upload strategies are presented. Finally, we provide the Video Quality Metric (VQM) data used for the Quality of Experience (QoE)-aware quality adaptation mechanisms.

A.1 ANALYZING SVC FOR P2P VOD

Using SVC with a P2P VoD system is challenging, specifically, parsing SVC videos to be used with P2P mechanisms. We now detail the methods and approaches we took in order to make use of SVC and quality adaptation in the P2P VoD prototype.

SVC provides a video with different qualities in one scalable bitstream. In order for the peers to decide which video quality to choose, SVC metadata information is needed. The metadata is extracted using the *BitStreamExtractor* tool, which is part of the JSVM software [80]. The extracted metadata includes: the number of layers, their bit rates, as well as the size of the different SVC transmission units.

Each layer is marked with an ID that starts with zero for the base layer and to $n-1$ for the highest layer, given that there are n layers. Each quality in the SVC file has a spatial layer with a *spatialID*, a temporal layer with a *temporalID*, and a quality layer with a *qualityID*. Therefore, each layer is denoted by the IDs of the three scalability dimensions as a tuple:

$$(\textit{spatialID}, \textit{temporalID}, \textit{qualityID}). \tag{A.1}$$

Each layer can have its own unique ID denoted by the *layerID*, defined as:

$$\begin{aligned} \textit{layerID} = \quad & \textit{spatialID} \cdot (\text{num. of quality layers}) \cdot (\text{num. of temporal layers}) \\ & + \textit{qualityID} \cdot (\text{num. of temporal layers}) \\ & + \textit{temporalID}. \end{aligned} \tag{A.2}$$

Temporal scalability is simple since it merely divides the video file into video frames. Since the *temporalID* for all spatial and quality layers within the same

frame are the same, we define a simplified layer index for those layers. This is denoted by the *sqID*, which is defined as:

$$sqID = spatialID \cdot (\text{num. of quality layers}) + qualityID. \tag{A.3}$$

Using Equation A.2 and Equation A.3, we can write the *layerID* as:

$$layerID = sqID \cdot (\text{num. of temporal layers}) + temporalID. \tag{A.4}$$

A.1.1 *NAL Units, Frames, and SQ-Layers*

The smallest entity of an SVC file (or any H.264/AVC video stream) is a Network Abstraction Layer (NAL) unit [80]. There is usually one NAL unit for each layer of each video frame. Additional information about the video format and layer dependencies are sent using so-called header NAL units. The most important NAL unit types within SVC are depicted in Table 9.

ID	NAL Unit Type	Description
1	Non-IDR picture	Base picture of a frame in the lowest quality ($sqID = 0$)
5	IDR picture	Same as above, but for a key picture (IDR frame)
6	SEI	Supplemental enhancement information (header)
7	Seq parameter set	Sequence parameter set (header)
8	Pic parameter set	Picture parameter set (header)
14	Prefix NAL unit	Information about dependencies between the layers (header)
15	Subset seq parameter set	Subset sequence parameter set (header)
20	Scalable extension	Additional picture data of a frame for higher spatial and quality layers ($sqID > 0$)

Table 9: Most important NAL unit types.

A video file is divided into many frames, each representing one image of the video sequence. Each frame is identified with a certain *temporalID*. The way temporal scalability is realized is important for the decoding and playback of the SVC video. Although three frames F_1, F_2, and F_3 will be played in that order, F_2 might be a B-frame and, therefore, can only be decoded after having received F_1 and F_3. Subsequently, the decoding order in this example is F_1, F_3, and F_2. The decoding order is important when selecting blocks for transmission, as it plays a major role in the priority calculation for block selection mechanisms.

Each frame contains picture data for each combination of spatial and quality layers, which are represented by the *sqID* index. Figure 55 illustrates a single frame with its spatial and quality layers (SQ-layers).

A frame always starts with a Supplemental Enhancement Information (SEI) NAL unit, sequence parameter set, picture parameter set, and subset sequence

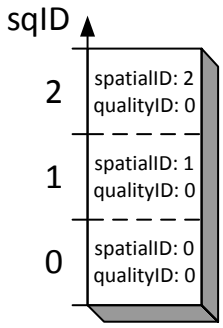


Figure 55: A single frame and its SQ-layers.

parameter set. These NAL units are standard H.264/AVC header NAL units. They are preceded with a prefix NAL unit, which contains header information for decoding the layers of the frame. All these header NAL units are part of the base layer, which means they belong to $sqID = 0$ for each frame. The payload picture data for the base layer is stored in a picture NAL unit, which is either a non-IDR or an IDR picture. The difference between both is described in Subsection A.1.2.

For proper playback of SVC files the base layer has to be available for each IDR period. This means the lowest resolution ($spatialID = 0$) and lowest quantization quality ($qualityID = 0$), or in other words: $sqID = 0$, for all frames have to be available. The reason behind this is that this base layer constitutes the H.264/AVC basic video stream, essential for playback[1].

The prefix NAL unit along with the picture NAL units hold all the information required to decode the base layer for a certain frame for $sqID = 0$.

All enhancement layers with $sqID > 0$ are stored in so-called scalable extension NAL units. There is one scalable extension for each enhancement layer $sqID$.

Table 10 shows an example on the NAL units of the first five frames of an SVC file with 3 spatial, 4 temporal, and 1 quality layers.

A.1.2 *IDR Periods*

An IDR period is defined as the set of frames which all have the same SQ-layer level or quality. An IDR period always starts with an IDR frame, i.e., a key frame that can be decoded independently. The rest of the frames within the IDR period are mostly prediction frames that are generated based on prediction information and are denoted as non-IDR frames. Nonetheless, other key frames can be found in a single IDR period.

Figure 56 shows an example of three IDR periods of a video file. This file has 49 frames per IDR, 24 frames per second frame rate, and, therefore, 2.042 seconds long IDR periods.

Choosing the correct size of an IDR period is quite challenging. On one hand, having longer IDR periods increases the efficiency of the video coding

1 For the actual playback, we use a modified version of the MPlayer: http://www.mplayerhq.hu, based on the open SVC decoder: http://sourceforge.net/projects/opensvcdecoder/.

Frame	NAL Unit ID	NAL Unit Type	*layerID*	(s, t, q)	*temporalID*	*sqID*
0	6	SEI	0	(0,0,0)	0	0
0	7	Seq parameter set	0	(0,0,0)	0	0
0	15	Subset seq par set	0	(0,0,0)	0	0
0	15	Subset seq par set	0	(0,0,0)	0	0
0	8	Pic parameter set	0	(0,0,0)	0	0
0	8	Pic parameter set	0	(0,0,0)	0	0
0	8	Pic parameter set	0	(0,0,0)	0	0
0	14	Prefix NAL unit	0	(0,0,0)	0	0
0	5	IDR picture	0	(0,0,0)	0	0
0	20	Scalable extension	4	(1,0,0)	0	1
0	20	Scalable extension	8	(2,0,0)	0	2
1	14	Prefix NAL unit	0	(0,0,0)	0	0
1	1	Non-IDR picture	0	(0,0,0)	0	0
1	20	Scalable extension	4	(1,0,0)	0	1
1	20	Scalable extension	8	(2,0,0)	0	2
2	14	Prefix NAL unit	1	(0,1,0)	1	0
2	1	Non-IDR picture	1	(0,1,0)	1	0
2	20	Scalable extension	5	(1,1,0)	1	1
2	20	Scalable extension	9	(2,1,0)	1	2
3	14	Prefix NAL unit	2	(0,2,0)	2	0
3	1	Non-IDR picture	2	(0,2,0)	2	0
3	20	Scalable extension	6	(1,2,0)	2	1
3	20	Scalable extension	10	(2,2,0)	2	2
4	14	Prefix NAL unit	3	(0,3,0)	3	0
4	1	Non-IDR picture	3	(0,3,0)	3	0
4	20	Scalable extension	7	(1,3,0)	3	1
4	20	Scalable extension	11	(2,3,0)	3	2

Table 10: All NAL units of the first 5 frames of the video used for the prototype evaluation.

process [83]. On the other hand, one IDR is the basic unit for quality adaptation since all frames within one IDR must be played with the same quality[2]. Therefore, having a long IDR period increases coding efficiency but makes the system slow in reacting to resource changes. Having a short IDR enables fast quality adaptation, but suffers from low coding efficiency.

2 For example, if an IDR is available in $sqID = 1$ and $temporalID = 2$, all SQ-layer parts with $sqID \leqslant 1$ must be available of all frames with $temporalID \leqslant 2$. Frames and SQ-layers with $sqID > 1$ and $temporalID > 2$ are only stored to the video file on disk if whole IDR period is available in the cache.

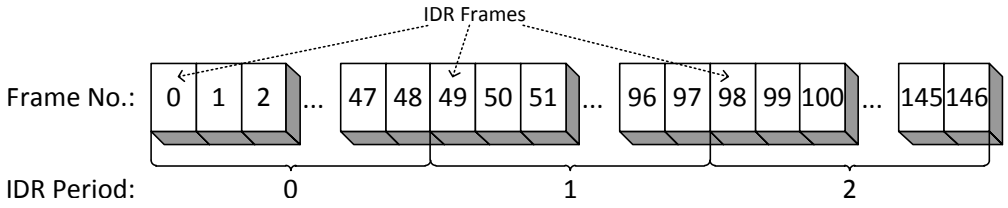


Figure 56: Frames are merged together to IDR Periods.

While trying to address this tradeoff, we ask the question: how fast should quality adaptation be performed? Motivated by the findings presented in [72], which indicates that a too frequent layer change, i.e., faster than two seconds, is not recommended, we decided to set the limit of the IDR period to two seconds.

A.1.3 *Packing NAL Units Into Blocks*

P2P blocks are used for the actual transmission of the video data over the P2P network. In P2P systems it is common to use blocks of fixed sizes. Therefore, we need to pack multiple NAL units into fixed-sized blocks.

In order to maximize the filling of blocks, we need to put as much NAL units as possible in a single block. Nonetheless, each block should only carry NAL units of the same layer in order to make the whole block useful for any peer. Therefore, multiple SQ-layer parts of a certain *sqID* of several frames with the same temporal layer are packed together. In this way, we can assign a certain *layerID* for each block, making it easier for the peers to map the block number to an SVC layer.

Since we cannot perfectly pack NAL units into blocks, we try to maximize the utilization of blocks. Therefore, NAL units are packed into blocks until they cannot hold any other NAL units of the same layer. The only exception is when a single NAL unit is larger than the block itself. In this case, those NAL units will be packed into special large blocks.

In Table 11, we show the number of used P2P streaming blocks for each layer for the SVC file used in the test best evaluation. As we notice, enhancement layers of temporal layers, i.e., layers with index 1, 2, 3, 5, 6, 7, 9, 8, 10, and 11 can be packed into less blocks since they are quite small.

Additional information about the SVC file used for the testbed evaluation presented in Section 6.6, namely: frames per IDR, number of IDR periods, and IDR period length, are shown in Table 12.

A.2 PROTOTYPE DESIGN AND IMPLEMENTATION

In this section, we present the prototype implementation of our system. This section is divided into two parts. The first details the prototype implementation of the basic P2P VoD system described in Chapter 4. The second part deals with using SVC and the required quality adaptation mechanisms and their realization in the basic P2P VoD prototype as described in Chapter 5.

Layer Index	SVC Level (d,t,q)	Picture size (Pixels)	Frame rate (fps)	Total Bit Rate (Kbps)	Number of P2P Blocks
0	(0,0,0)	176 × 144	3.75 fps	167.78 kbps	55
1	(0,1,0)	176 × 144	7.50 fps	197.73 kbps	10
2	(0,2,0)	176 × 144	15.00 fps	218.81 kbps	7
3	(0,3,0)	176 × 144	30.00 fps	237.91 kbps	7
4	(1,0,0)	352 × 288	3.75 fps	984.39 kbps	300
5	(1,1,0)	352 × 288	7.50 fps	1278.66 kbps	91
6	(1,2,0)	352 × 288	15.00 fps	1545.65 kbps	80
7	(1,3,0)	352 × 288	30.00 fps	1764.70 kbps	65
8	(2,0,0)	704 × 576	3.75 fps	2332.70 kbps	560
9	(2,1,0)	704 × 576	7.50 fps	2893,37 kbps	90
10	(2,2,0)	704 × 576	15.00 fps	3349.10 kbps	62
11	(2,3,0)	704 × 576	30.00 fps	3721.95 kbps	50

Table 11: SVC video structure with respective quality levels, total bit rates, and number of P2P blocks.

Parameter	Value
Highest bit rate	3721.95 kbps
Block size	64 KB
Total number of blocks	1377
Max frames per second	30
Frames per IDR	50
Number of IDR Periods	200
IDR period length	1.67 seconds

Table 12: Additional information on the used SVC video file.

A.2.1 *Basic P2P VoD Prototype*

This section presents the design of the prototype of our P2P VoD system called P2PStream.KOM. A screen shot of the streaming application is presented in Figure 57.

For the sake of brevity, we restrict the descriptions in this section to the more high-level algorithms and architecture. These are shown in Figure 58. The algorithms are executed in the following order: joining the network, regular state exchange, video search, peer discovery, connection establishment, initial choking, block selection, choking and unchoking, and finally download and playback.

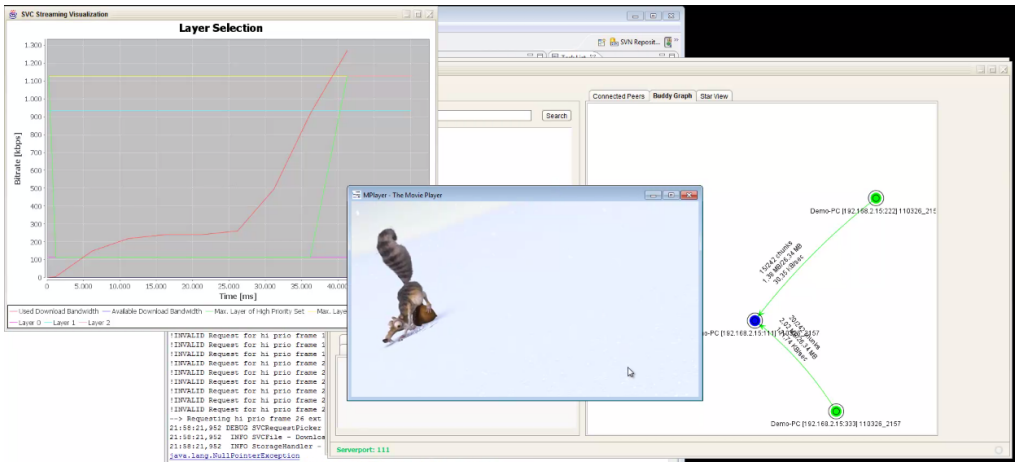


Figure 57: Screenshot of the P2PStream.KOM application during runtime.

Joining the Network

Joining the network is performed as soon as the user starts the P2PStream.KOM application. The basic configuration used to initialize the streaming application is retrieved from a configuration file.

It is important for the tracker to know the public IP of each peer, therefore, the Session Traversal Utilities for NAT (STUN) protocol is used. The next step is to prepare for receiving connections from other peers. Therefore, a Transmission Control Protocol (TCP) socket for incoming connections is opened. If the default port of 1234 is blocked, then a random port is chosen. By combining the IP address and the port number of the peer, a unique peer identifier is created. This peer ID is used for all further communication with the tracker, server, and other peers.

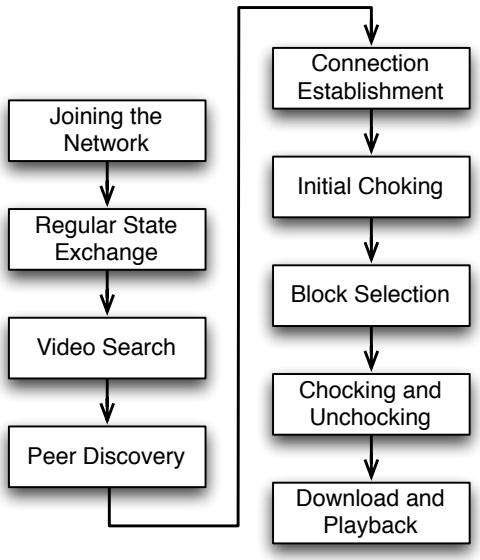


Figure 58: Algorithm workflow of the P2PStream.KOM prototype.

The next step is to launch the Graphical User Interface (GUI) of the P2PStream.KOM application. A connection to the tracker is established, and a list of locally shared videos is announced. The streaming application scans within a specific video directory for the videos shared at the peer.

To keep track of all received video blocks even after the streaming application is restarted, an index file and properties file are maintained at each peer for every video. The index file contains the block map, that keeps record of all video blocks that haven been downloaded. The properties file, on the other hand, contains the metadata of the video. This includes information on the used video codec, for example whether single or multi-layer. Additionally, it contains its length, resolution, and bit rate. If the index or property files do not exist for a certain video, they are created.

The peer that is the first to share a certain video has to generate its metadata so that other peers can correctly play the video. The metadata of a video is created using the MediaInfo[3] library for single-layer videos and using the BitStreamExtractor tool [80] for SVC videos. Each video has a unique identifier based on the MD5 checksum. Thereby, changing the file name does not lead to a new video in the system.

Regular State Exchange

From the time a peer joins until it leaves, the peer will regularly, default every 5 seconds, report its status to the tracker. This is similar to other systems, e.g. BitTorrent [24], where this is known as the keep-alive message. A peer status message contains the following information.

- The peer's identifier and address (IP and port).
- Its current status within the streaming system, whether it is a downloader or uploader.
- Its maximum user set download and upload bandwidth capacities.
- Its current download and upload bandwidth utilization.
- Its current streaming state: the video being watched, playback position, and download progress.
- How well are the playback buffer and high priority set are filled with video data.

The tracker stores the data along with a timestamp so that to detect peers that do not send status reports on time. Usually, all peers that do not report fast enough to the tracker, will be removed from the database.

Based on the number of online peers, a dynamic upload bias probability is calculated by the tracker. This probability, is used by the peers to balance low and high priority requests (see Section 4.2). After a peer has joined the network, the tracker directly assigns it to a server so that it can always have a reliable data transmission.

3 www.mediainfo.sourceforge.net

After the tracker has received the announce message, it proceeds by processing the peer state and returns the following information:

- The address of the assigned server and an access permit.
- The new dynamic upload bias probability P.

Video Search

After the peer has connected and received the basic information from the tracker, the next step is for the user to search for some video content. This is done using a simple keyword search. Thereby, the search string entered by the user is sent to the tracker as a part of a search message. The tracker runs a search query on its local database and returns back a list of matches. The list of hits contains additional metadata about the videos in order to help the peer decide on the correct video file. This includes:

- Video name and ID
- Video duration in minutes
- Video bit rate
- Number of blocks

The GUI will then display the list of matches in the search results box. It is worth mentioning that since we use the *MD5* hash of the video files, announcing the same file with different names does not lead to duplicates.

The next step takes place when the user selects one of the returned videos and starts the streaming process. This will trigger the following steps in the streaming application: peer discovery, connection establishment, initial choking, block selection, chocking and unchoking, and, finally, download and playback. These are explained next.

Peer Discovery

In order to find out which other peers are also streaming the same selected video, a request message is sent to the tracker. This message contains the selected video ID, where the tracker returns a list of 50 peers that are also streaming the same video file. This list is randomized in the sense that it may contain both downloader and uploader peers, as well as peers with an advanced playback position and others with a close playback position.

In order to perform better peer discovery, the peers can additionally be sorted according to some criteria. For example, the tracker can perform load balancing by sorting the peers by their current upload utilization. Thereby, peers that are not uploading a lot are put on the top of the list and are preferred for selection.

Since some of the discovered peers may leave the system either gracefully (playback finished) or ungracefully (hardware failures), the peer may request new peers from the tracker so that video delivery is sustained.

Connection Establishment

After the peer has discovered other peers streaming the same video, the requesting peer starts to establish connections to those peers. This is done until the download connections limit has been reached. The point of having a certain number of connections is that having too less leads to suboptimal utilization of the download capacity, while having too many increases the chances of block timeout as the capacity of each individual connection is too low. The default used, and which has shown to exhibit the best results, is 10 connections. Such a value has also been reported in [63].

When a downloader peer attempts to establish a connection to another uploader peer, a handshake protocol is used to have the ability to reject connections as presented in Chapter 4. Connections are rejected in the case the number of upload connections is reached.

As soon as the first connection has been established, the downloader peer requests the properties or metadata file of the video file. Additionally, a dummy file with the same size reported in the metadata is created. This file will be filled progressively with the video data received from other peers.

For every uploader peer, the *block map* is requested. This block map, as explained in Chapter 4, contains a list of blocks already downloaded. The block map of the connected uploader peers are updated regularly so that the information available at the downloader peers is fresh.

Initial Choking

Since having too many simultaneous upload transmissions would lead to low effective capacity for each individual connection, it is important to keep a limit on the number of peers that download video data at once. Therefore, all connected peers are constantly evaluated whether they are eligible to receive an active upload connection. The initial connection does not directly mean that the peer can download data. This is denoted by the initial choking, i.e., no video data is sent from the uploader to the downloaded peer. As long as the downloader peer is in the choked state, it regularly sends a list of needed blocks and asks to be un-choked. Next we go over the block selection mechanisms that determine the list of needed blocks.

Block Selection

Before we detail the block selection algorithms, we first present the video structure used in the prototype.

VIDEO STRUCTURE. A video file is virtually divided into n blocks of a default size of 64 KB. Having fixed sized-blocks makes the block management algorithms simpler. In Subsection A.1.3 we have presented how SVC NAL units with variable sizes can be packed into the fixed-sized blocks.

As shown in Figure 59, all blocks of the video files are divided into four sets.

- Already played blocks are those that have already been played and are located behind the playback position.

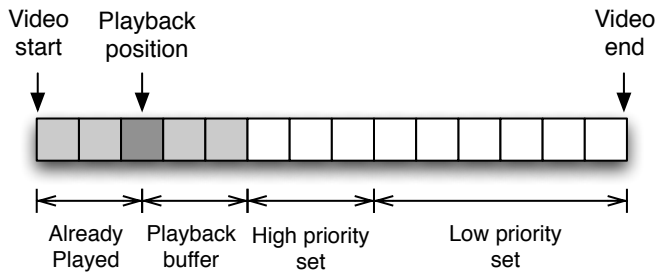


Figure 59: Video file structure in P2PStream.KOM.

- The playback buffer blocks are those immediately starting at the playback position and covering an equivalent of seven seconds of video. This playback buffer is used to absorb any short-term fluctuations in the download throughput. Playback is stopped once the buffer is completely empty and is only resumed once the buffer is full.
- The high priority set starts at the playback buffer and has a configurable length. By default, it is twice as large as the playback buffer. The blocks within this set are requested in sequence so that to match the need of the video player.
- The low priority set constitutes the rest of the blocks that are not directly important for playback. This set is used to prefetch video blocks that are interesting for other peers in the system.

Block selection is performed every time the peer gets choked and un-choked as follows:

- After the downloader peer is choked, the peer has to determine its needed video blocks and announces them at the uploader. This information is used by the uploader to re-evaluate whether it should unchoke the downloader peer or not.
- In case the peer gets unchoked, the peer has to determine the actual blocks it wants to retrieve from the uploader. Since the block that was reported in the previous choking round might have already been requested from some other uploader, this step is essential to keep the uploader informed concerning the required blocks.

For the actual block selection, the peer always selects from the high priority set as long as it is not full. Otherwise, the peer would select blocks from the low priority set. Details regarding the block selection algorithms were presented in Section 4.1.

Choking and Unchoking

Every five seconds, the uploader peer has to reevaluate its choking decisions, i.e., to which peers should the upload connections be assigned. Since the high

and low priority sets have different roles in the streaming process, both requests are separately reported in the unchoking process. Within each set, high and low priority unchoking requests are evaluated separately.

For the high priority blocks, the requests are sorted according to the playback deadline. This means that peers needing blocks closer to their playback positions are unchoked first, thereby reducing the chance of having a stalled playback.

For the low priority blocks, the blocks are sorted according to the Soon Most Needed (SMN) score as calculated in Subsection 4.2.2. This means that blocks that are more urgently needed in certain peer neighborhoods are uploaded first.

Regular (i.e., non-server) peers serve high priority requests with a dynamic probability, the so called upload bias probability P, which is computed at the tracker as explained in Subsection 4.2.3.

The number of concurrent upload connections is calculated based on the upload capacity of the peer. Therefore, the faster is a peer, the more simultaneous upload transmissions it can support. This mechanism helps in better supporting the heterogeneous resources available in the P2P network.

Download and Playback

Once a peer receives an unchoke message from the uploader, the block selection algorithms is executed to determine the actual block to request, which is sent to the uploader. After the block has been received and given that the peer is still unchoked, further requests are sent the uploader. This is done until the peer has finished downloading the whole video file or if the peer gets choked, for example due to an advanced download progress compared to other peers.

When the buffer has been filled, playback is started. At this time the *playback monitor* is executed so that to launch the video player. In case the buffer runs low, playback is paused until the buffer is full.

A.2.2 *Quality Adaptation and SVC Block Selection*

Quality adaptation mechanisms are essential in matching the video quality with available peer and system resources. We have extensively described our quality adaptation algorithms in Subsection 5.1.4.

Based on the static peers resources, the Initial Quality Adaptation (IQA) selects a target layer with which the download bandwidth is maximized. Additionally, the Progressive Quality Adaptation (PQA) adapts to the real time resources and changes the video quality according to available throughput.

Within the prototype configuration, it is possible to define a minimum and a maximum layer for the three dimensions of scalability where it is possible for the user to set a *minLayer* and a *maxLayer*. Thereby, the user may provide preferences regarding the desired video quality. While the *minLayer* and *maxLayer* are used as global limitations, the IQA and PQA take those constraints into account while making the layer selections.

Playback of the SVC video is started only of the playback buffer is full with at least the layer selected by the PQA or the IQA in case the PQA has not been made yet. If a NAL unit belonging to the selected layer is missing, playback is paused.

Block Selection Order

The basic idea behind the order of the block selection in P2PStream.KOM is shown in Figure 60.

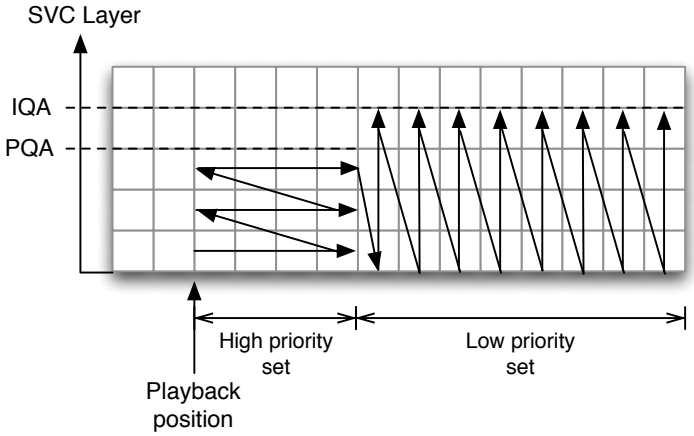


Figure 60: Order of block requests based on SVC. High priority requests are determined by the PQA while the low priority ones are determined by the IQA.

For the high priority set, we perform a base-layer-first strategy with the PQA-selected layer used as a target layer. This is done since the PQA reflects real time resources, thereby, its selected layer reflects the streamable quality. Additionally, the high priority set is requested, as shown in Figure 60, starting with the base layer and gradually going up to higher layers. The goal of this approach is to have the possibility to play the video as soon as possible. Further, since the PQA may decrease the layer when available resources degrade, this mechanism helps in minimizing the amount of useless layers in case the PQA drops down the quality.

Regarding the low priority set, the IQA-selected layer is used as a target layer. The reason behind this is that if a peer is able to download from the low priority set, it indicates that it is enjoying a relatively good throughput. Therefore, this peer should try to maximize the downloaded layer according to the highest possible quality, i.e., the quality selected by the IQA. Since the high priority set can have a rather large size, we do not perform a base-layer-first selection strategy, as shown in Figure 60,

A.3 EVALUATION OF THE PREFETCHING AND UPLOAD STRATEGIES

In this section, we present additional evaluation results that compliment those presented in Chapter 6. Subsequently, we focus on assessing the impact of the prefetching and upload strategies presented in Section 4.2.

In order to show the benefits of the proposed approach we conduct two groups of experiments: (1) comparison of the rarest-first and SMN prefetching strategies and (2) evaluation of the best upload bias probability p. In the following, we make use of relatively long videos, such as television series or movies. The content provider is interested in guaranteeing playback performance up to a certain number of users depending on server capacity and content properties. It is also interested in maximizing revenues by minimizing traffic it is serving.

We use a peer bandwidth model with uniform values for all peers with 1024 kbps download and 256 kbps upload link capacities, which are reported to be the average speed of today's Internet users [14]. The rest of the simulation workload is configured according to Table 13.

Parameter	Value
Video length	60 minutes
Video bit rate	512 kbps
Available servers	4
Server capacity (up)	4086 kbps
Peer capacity (up)	256 kbps
Peer capacity (down)	1024 kbps
Playback buffer size	7 seconds
Piece size	64 KB
Number of pieces	1800
Neigborhood size	16

Table 13: Simulation setup used to evaluate the prefetching and upload strategies.

The main scenario we use to evaluate our prefetching and upload algorithms is based on a double flash crowd scenario, shown in Figure 61.

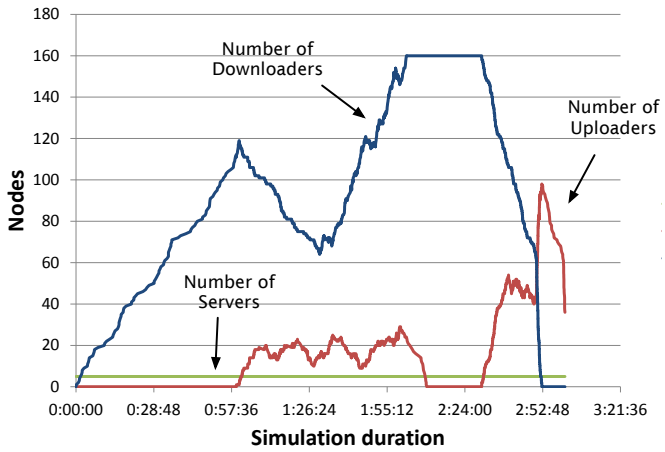
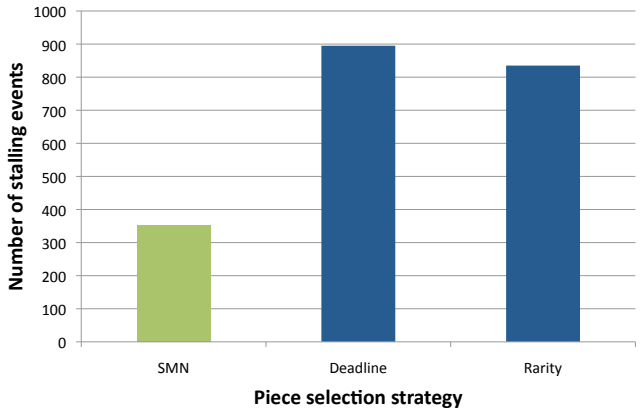


Figure 61: The double flash crowd scenario.

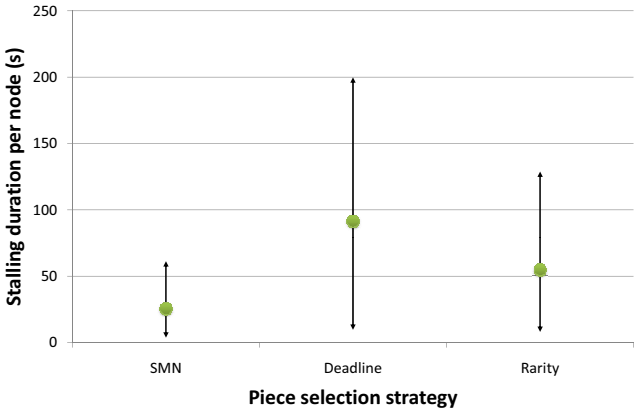
This scenario is based on the idea that people go home at 6pm, they would start streaming some series or similar content creating a first flash crowd effect. After some drop in active users, another flash crowd takes place at around 8pm which is known as the primetime, where many users join the system to watch the video content. In our simulation, 120 and 180 peers join the system during the first and second flash crowds respectively, resulting in a total of 280 peers. Both crowds follow an exponential distribution.

A.3.1 *Soon-Most-Needed Prefetching*

To have a fair comparison, piece selection of high priority pieces is based on an in-order selection, while prefetching is based on either SMN or rarest-first. The results of this comparison are shown in Figure 62, where Figure 62a shows the number of stalling events for the whole simulation, and Figure 62b shows average stalling duration for each stalling event.



(a) Number of stalling events for all peers over the whole simulation run.



(b) Average, minimum and maximum stalling duration per stalling event.

Figure 62: Performance comparison of the different prefetching strategies.

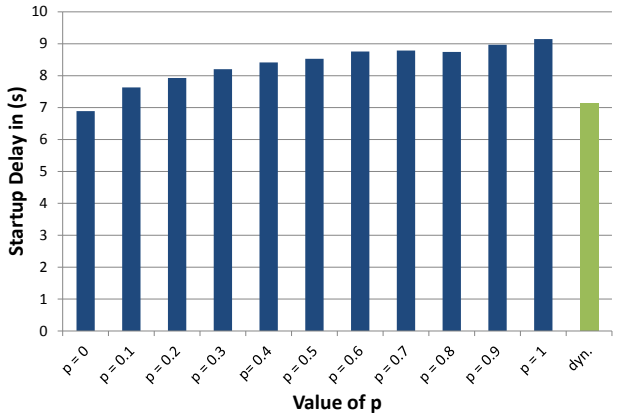
We found that using SMN resulted in better performance than rarest-first, which in turn performed better than deadline. Although the fact that deadline-based prefetching performs the worst might be surprising at first, it can be ex-

plained by the weak piece distribution this strategy inflicts. Such performance was also reported in [40].

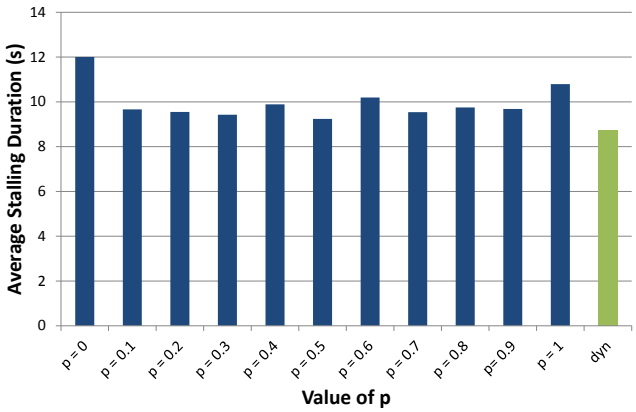
Comparing SMN and rarest-first, rarest-first generated 835 stalling events while SMN generated only 353 (around 50% better performance). In addition, the average duration of each stalling event is 45% shorter when using SMN. The performance advantage for SMN can be explained by the fact that this strategy prepares the system, already from the beginning, for flash crowds by prefetching those pieces that will soon be needed by the neighborhood. Therefore, peers already longer in the system have good piece distribution in terms of how much those are needed. Hence, those peers would offload servers by exchanging those pieces already prefetched, while servers can serve new comers until they can start prefetching and support other peers.

A.3.2 *Dynamic Upload Strategy*

The second set of simulations aims at evaluating the proposed dynamic upload strategy. We compare the adaptive strategy that controls the probability p (as presented in Subsection 4.2.3) with static upload bias probability p ranging from zero to one with 0.1 increments. The results are depicted in Figure 63.



(a) Average startup delay.



(b) Average stalling duration per stalling event.

Figure 63: Evaluation and impact of static and dynamic upload bias probability p on system performance.

As can be seen in Figure 63, session quality indicators (i.e., start-up delay, and average stalling duration) show better performance when using our adaptive strategies.

A.4 VQM EVALUATIONS OF TEST VIDEOS

Table 14, Table 15, and Table 16 detail the VQM ratings of the three SVC video used for the simulative evaluation of the adaptation mechanisms presented in Section 6.5. Details on how this data was derived are presented in [113].

Spatial level (d)	Temporal level (t)	SNR level (q)	Data rate (Mbyte/s)	VQM rating
0	0	0	0.099502	0.7080
1	0	0	0.237859	0.6980
2	0	0	0.344678	0.6991
3	0	0	0.483708	0.7017
0	1	0	0.116962	0.5987
1	1	0	0.278904	0.5755
2	1	0	0.405544	0.5699
3	1	0	0.571487	0.5720
0	2	0	0.133664	0.4173
1	2	0	0.319317	0.3905
2	2	0	0.467042	0.3859
3	2	0	0.662858	0.3886
0	3	0	0.148941	0.2617
1	3	0	0.356181	0.2279
2	3	0	0.521065	0.2230
3	3	0	0.742194	0.2253
0	4	0	0.164336	0.0557
1	4	0	0.394226	0.0188
2	4	0	0.574822	0.0089
3	4	0	0.820000	0.0000

Table 14: The VQM ratings of the blue sky video.

Spatial level (d)	Temporal level (t)	SNR level (q)	Data rate (Mbyte/s)	VQM rating
0	0	0	0.083343	0.7838
1	0	0	0.196833	0.7635
2	0	0	0.326427	0.7580
3	0	0	0.508360	0.7605
0	1	0	0.124848	0.6141
1	1	0	0.295807	0.5816
2	1	0	0.486092	0.5734
3	1	0	0.749941	0.5762
0	2	0	0.174872	0.4276
1	2	0	0.423468	0.3874
2	2	0	0.688186	0.3803
3	2	0	1.046318	0.3846
0	3	0	0.216091	0.2345
1	3	0	0.534859	0.1730
2	3	0	0.881488	0.1635
3	3	0	1.333625	0.1676
0	4	0	0.235070	0.1170
1	4	0	0.591817	0.0318
2	4	0	1.003290	0.0123
3	4	0	1.540000	0.0000

Table 15: The VQM ratings of the crowd run video.

Spatial level (d)	Temporal level (t)	SNR level (q)	Data rate (Mbyte/s)	VQM rating
0	0	0	0.089190	0.7204
1	0	0	0.195647	0.6977
2	0	0	0.338438	0.6896
3	0	0	0.568799	0.6913
0	1	0	0.134219	0.5629
1	1	0	0.298488	0.5298
2	1	0	0.528358	0.5213
3	1	0	0.890658	0.5230
0	2	0	0.180994	0.4292
1	2	0	0.418088	0.3900
2	2	0	0.778516	0.3863
3	2	0	1.328312	0.3825
0	3	0	0.197295	0.2924
1	3	0	0.473345	0.2418
2	3	0	0.956511	0.2317
3	3	0	1.703705	0.2350
0	4	0	0.202510	0.1064
1	4	0	0.487983	0.0306
2	4	0	1.004883	0.0126
3	4	0	1.850000	0.0000

Table 16: The VQM ratings of the park joy video.

B

AUTHOR'S PUBLICATIONS

B.1 MAIN PUBLICATIONS

[1] Osama Abboud, Julius Rückert, and David Hausheer. QoE-aware Quality Adaptation in Peer-to-Peer Video-on-Demand. Technical Report PS-TR-2012-01, Peer-to-Peer Systems Group, Technische Universität Darmstadt, 2012.

[2] Julius Rückert, Osama Abboud, Thomas Zinner, David Hausheer, and Ralf Steinmetz. Quality Adaptation in P2P Video Streaming Based on Objective QoE Metrics. In *IFIP Networking*. 2012.

[3] Osama Abboud, Konstantin Pussep, Aleksandra Kovacevic, Katharina Mohr, Sebastian Kaune, and Ralf Steinmetz. Enabling Resilient P2P Video Streaming: Survey and Analysis. *Multimedia System Journal (MMSJ)*, 17(3), 2011.

[4] Osama Abboud, Konstantin Pussep, Dominik Stingl, and Ralf Steinmetz. Media-aware Networking for SVC-based P2P Streaming . In *Network and Operating System Support for Digital Audio and Video (NOSSDAV)*. 2011.

[5] Osama Abboud, Thomas Zinner, Konstantin Pussep, Sabah Al-Sabea, and Ralf Steinmetz. On the Impact of Quality Adaptation in SVC-based P2P Video-on-Demand Systems. In *ACM Multimedia Systems Conference (MMSys)*. 2011.

[6] Osama Abboud, Konstantin Pussep, Markus Mueller, Aleksandra Kovacevic, and Ralf Steinmetz. Advanced Prefetching and Upload Strategies for P2P Video-on-Demand. In *ACM Workshop on Advanced Video Streaming Techniques for Peer-to-Peer Networks and Social Networking*. 2010.

[7] Osama Abboud, Thomas Zinner, Eduardo Lidanski, Konstantin Pussep, and Ralf Steinmetz. StreamSocial: A P2P Streaming System with Social Incentives. In *IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*. 2010.

[8] Osama Abboud, Thomas Zinner, Konstantin Pussep, Simon Oechsner, Ralf Steinmetz, and Phuoc Tran-Gia. A QoE-Aware P2P Streaming System Using Scalable Video Coding. In *IEEE International Conference on Peer-to-Peer Computing (P2P)*. 2010.

[9] Osama Abboud, Aleksandra Kovacevic, Kalman Graffi, Konstantin Pussep, and Ralf Steinmetz. Underlay Awareness in P2P Systems: Techniques and Challenges. In *23th IEEE International Parallel and Distributed Processing Symposium, Workshops and Phd Forum (IPDPSW)*. 2009.

[10] Osama Abboud, Konstantin Pussep, Aleksandra Kovacevic, and Ralf Steinmetz. Quality Adaptive Peer-to-Peer Streaming Using Scalable Video Coding. In *12th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS)*. 2009.

B.2 OTHER PUBLICATIONS

[1] Konstantin Pussep, Osama Abboud, Florian Gerlach, Ralf Steinmetz, and Thorsten Strufe. Adaptive Server Allocation for Peer-assisted VoD. In *24th IEEE International Parallel and Distributed Processing Symposium, Workshops and Phd Forum (IPDPSW)*. 2010.

[2] Konstantin Pussep, Sebastian Kaune, Osama Abboud, Christian Huff, and Ralf Steinmetz. On Energy-Awareness for Peer-assisted Streaming with Set-Top Boxes. In *6th International Conference on Network and Service Management (CNSM)*. 2010.

[3] Thomas Zinner, Osama Abboud, Oliver Hohfeld, and Phuoc Tran-Gia. Towards QoE Management for Scalable Video Streaming . In *21st ITC Specialist Seminar on Multimedia Applications - Traffic, Performance and QoE*. 2010.

[4] Thomas Zinner, Oliver Hohfeld, Osama Abboud, and Tobias Hossfeld. Impact of Frame Rate and Resolution on Objective QoE Metrics. In *Second International Workshop on Quality of Multimedia Experience*. 2010.

[5] Konstantin Pussep, Simon Oechsner, Osama Abboud, Miroslaw Kantor, and Burkhard Stiller. Impact of Self-Organization in Peer-to-Peer Overlays on Underlay Utilization. In *4th International Conference on Internet and Web Applications and Services (ICIW)*. 2009.

C

CURRICULUM VITAE

PERSONAL

Name	Osama Abboud
Date of Birth	August 07, 1982
Place of Birth	Deir Ames, Lebanon

EDUCATION

Since 3/2008	Technische Universität Darmstadt, Germany Doctoral candidate at the Department of Electrical Engineering and Information Technology
3/2006–2/2008	University of Ulm, Germany Communication Technology Degree: Masters of Science
9/2000–8/2005	Lebanese University, Lebanon Computer and Communication Engineering Degree: Diplom

WORK EXPERIENCE

Since 3/2008	Technische Universität Darmstadt, Germany Research assistant at the research group *Peer-to-Peer Systems* at the Multimedia Communications Lab (KOM)
9/2005–2/2006	Heya TV, Lebanon Systems Engineer

TEACHING ACTIVITY

Since SS 2008	Technische Universität Darmstadt, Germany Tutor for various Bachelor-, Diploma, and Master theses
Since SS 2008	Technische Universität Darmstadt, Germany Organizer of the lecture *Industrial Colloquium*
Since WS 2010	Technische Universität Darmstadt, Germany Tutor for the Seminar *Advanced Topics in Future Internet Research*

SCIENTIFIC ACTIVITIES

Reviewer	**Journals** Image Communications Journal, 2011

Multimedia System Journal (MMSJ), 2011
Computer Networks Journal (ComNet), 2008
Computer Communications (ComCom), 2009–2011

Selected Conferences and Workshops
ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), 2010,2011
ACM Multimedia Systems (MMSys), 2011-2012
IEEE International Conference on Peer-to-Peer Computing (P2P), 2008–2011
ACM Multimedia (MM), 2011
IEEE Conference on Communications (ICCC), 2009

MEMBERSHIPS

Since 2009	Institute of Electrical and Electronics Engineers (IEEE) student member
Since 2010	Association for Computing Machinery (ACM) student member

Darmstadt, 2012

Osama Abboud

D

ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe.

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 2012

Osama Abboud