



Dipl.-Ing. Benjamin Rainer, Bakk. techn.

# **Self-Organized Inter-Destination Multimedia Synchronization**

## **DISSERTATION**

submitted in fulfilment of the requirements for the degree of  
Doktor der technischen Wissenschaften

Alpen-Adria-Universität Klagenfurt  
Fakultät für Technische Wissenschaften

### **Mentor**

Priv.-Doz. Dr. Dipl.-Ing. Christian Timmerer  
Alpen-Adria-Universität Klagenfurt, Austria  
Department of Informationstechnology (ITEC)

### **First Evaluator**

Priv.-Doz. Dr. Dipl.-Ing. Christian Timmerer  
Alpen-Adria-Universität Klagenfurt, Austria  
Department of Informationstechnology (ITEC)

### **Second Evaluator**

Priv.-Doz. Dr. Matthias Klusch  
German Research Center for Artificial Intelligence (DFKI) GmbH  
Agents and Simulated Reality Department

Klagenfurt, Juni 2015



## Eidesstattliche Erklärung

Ich versichere an Eides statt, dass ich

- die eingereichte wissenschaftliche Arbeit selbständig verfasst und andere als die angegebenen Hilfsmittel nicht benutzt habe;
- die während des Arbeitsvorganges von dritter Seite erfahrene Unterstützung, einschließlich signifikanter Betreuungshinweise, vollständig offengelegt habe;
- die Inhalte, die ich aus Werken Dritter oder eigenen Werken wortwörtlich oder sinngemäß übernommen habe, in geeigneter Form gekennzeichnet und den Ursprung der Information durch möglichst exakte Quellenangaben (z.B. in Fußnoten) ersichtlich gemacht habe;
- die Arbeit bisher weder im Inland noch im Ausland einer Prüfungsbehörde vorgelegt habe und dass
- die zur Plagiatskontrolle eingereichte digitale Version der Arbeit mit der gedruckten Version übereinstimmt.

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

## Affidavit

I hereby declare in lieu of an oath that

- the submitted academic paper is entirely my own work and that no auxiliary materials have been used other than those indicated;
- I have fully disclosed all assistance received from third parties during the process of writing the paper, including any significant advice from supervisors;
- any contents taken from the works of third parties or my own works that have been included either literally or in spirit have been appropriately marked and the respective source of the information has been clearly identified with precise bibliographical references (e.g., in footnotes);
- to date, I have not submitted this paper to an examining authority either in Austria or abroad and that
- the digital version of the paper submitted for the purpose of plagiarism assessment is fully consistent with the printed version.

I am aware that a declaration contrary to the facts will have legal consequences.

Unterschrift/Signature:

---

Klagenfurt, 19. Juni 2015



# Contents

<b>Acknowledgments</b>	<b>V</b>
<b>Abstract</b>	<b>VII</b>
<b>Kurzfassung</b>	<b>IX</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Research Objectives . . . . .	5
1.3 Contributions . . . . .	7
1.4 Structure . . . . .	8
<b>2 Technical Background and Related Work</b>	<b>11</b>
2.1 Multimedia Synchronization . . . . .	11
2.2 Inter-Destination Multimedia Synchronization Schemes . . . . .	14
2.3 Adaptive Media Playout . . . . .	18
2.4 MPEG-DASH . . . . .	20
2.5 Subjective Quality Assessment of Audio-Video Content using Crowdsourcing	27
<b>3 Distributed negotiation on a Reference Playback Timestamp</b>	<b>35</b>
3.1 Introduction . . . . .	35
3.2 Session Management . . . . .	37
3.3 Signalling of Timing Information and Negotiation of the Playback Timestamp	42
3.3.1 Unstructured Peer-To-Peer Overlay Construction and Coarse Syn-	
chronization . . . . .	43
3.3.2 Aggregate . . . . .	48
3.3.3 Merge and Forward . . . . .	52
3.4 Experimental Results . . . . .	61
3.4.1 Overhead Analysis . . . . .	61
3.4.2 Synchronization Time Analysis . . . . .	69
3.5 Conclusion . . . . .	74

<b>4</b>	<b>Adaptive Media Playback for achieving IDMS</b>	<b>77</b>
4.1	Introduction . . . . .	77
4.2	AMP using Content Features vs. plain AMP . . . . .	78
4.2.1	Random Selection of Content Sections . . . . .	79
4.2.2	QoE- and Context-aware Adaptive Media Playback . . . . .	79
4.2.3	Stimuli and Stimulus Presentation . . . . .	85
4.2.4	Crowdsourcing Platform and Participants . . . . .	87
4.2.5	Evaluation Methodology . . . . .	89
4.2.6	Filtering of Participants . . . . .	91
4.2.7	Statistical Analysis of the Responses . . . . .	94
4.2.8	Discussion on the results . . . . .	97
4.3	Quantification of the Distortion caused by AMP . . . . .	98
4.3.1	Quantifying the Distortion of Multimedia Content in the Temporal Domain . . . . .	98
4.3.2	Participants, Stimuli, Methodology, and Assessment Platform . . . . .	101
4.3.3	Statistical Analysis of the Results . . . . .	102
4.3.4	QoE Utility Model for AMP . . . . .	105
4.3.5	Instantiation and Validation of the Utility Model . . . . .	108
4.3.6	Discussion and Conclusion . . . . .	109
4.4	Dynamic AMP . . . . .	110
4.4.1	Dynamic AMP for IDMS . . . . .	111
4.4.2	Evaluation of Dynamic AMP for IDMS . . . . .	114
4.5	Conclusion . . . . .	117
<b>5</b>	<b>Applications</b>	<b>121</b>
5.1	Introduction . . . . .	121
5.2	Mobile DASH Encoder . . . . .	121
5.2.1	Architecture . . . . .	123
5.2.2	Mobile DASH Encoder on Android . . . . .	125
5.2.3	General Usage Scenarios . . . . .	128
5.2.4	Energy consumption of MDE . . . . .	129
5.2.5	Conclusion . . . . .	131
5.3	Web-based Assessment Platform for Subjective QoE Experiments . . . . .	132
5.3.1	Architecture . . . . .	133

5.3.2	Introduction and Questionnaires . . . . .	134
5.3.3	Stimuli Presentation and Rating Possibility . . . . .	137
5.3.4	Conclusion . . . . .	139
<b>6</b>	<b>Discussion and Conclusion</b>	<b>141</b>
6.1	Summary . . . . .	141
6.2	Findings . . . . .	142
6.3	Future Work . . . . .	145





# Acknowledgments

This work would not have been possible without the support of PD. Dr. Christian Timmerer and Univ.-Prof. Dr. Hermann Hellwagner. Thanks for giving me the opportunity to be part of your great team. Special thanks to PD. Dr. Christian Timmerer for his patience, time, understanding, and guidance. I would like to say thanks to my colleague Dipl.-Ing. Christian Raffelsberger for the fruitful discussions we had. Thanks PD. Dr. Stefan Rass for your help with mathematical problems and for your patience.

I would like to further thank PD. Dr. Matthias Klusch and Dipl.-Inf. Patrick Kapahnke for the excellent collaboration in the SocialSensor project. It was a honour for me to collaborate with you on the MyMedia application and on the resulting scientific contributions which have been published in high quality conferences.

My final words go to my family, without whose support none of this would ever have been possible.

This work was supported in part by the EC in the context of the SocialSensor (FP7-ICT-287975) and QUALINET (COST IC 1003) projects and partly performed in the Lakeside Labs research cluster at Alpen-Adria-Universität.



# Abstract

Experiencing multimedia content together has become common in the last four decades. For example, the traditional TV scenario with friends, colleges and/or the family. With the invention of mobile devices and the upraise of social networks this traditional scenario tends to drift more and more towards a distributed multimedia experience where the participating users are geographically distributed. Nevertheless, the users want to have the same experience and possibilities as if they all were in the same room, watching the multimedia content together using a TV. In order to provide nearly the same experience as one would have during a traditional TV session with friends, it is mandatory that the multimedia playback of the participating users is synchronized. This new type of synchronization is called Inter-Destination Multimedia Synchronization.

This thesis investigates how Inter-Destination Multimedia Synchronization can be achieved by using a self-organized and distributed approach for calculating the reference playback timestamp among a group of users. But, determining the reference playback timestamp is not enough. The actual process of carrying out the synchronization is another very important and often neglected part of Inter-Destination Multimedia Synchronization because it directly affects the Quality of Experience. Therefore, this thesis further investigates how to carry out the synchronization trying to minimize the impact on the Quality of Experience.

In the first part of this thesis we introduce Inter-Destination Multimedia Synchronization to MPEG Dynamic Adaptive Streaming over HTTP and introduce the notion of an Inter-Destination Multimedia Synchronization session object. We split the process of calculating the reference playback timestamps into two parts. The first part creates an application layer peer-to-peer overlay network among the participating peers and tries to make an educated guess where (in the multimedia content) to start the actual multimedia playback. The second part uses our distributed algorithm that calculates the reference playback timestamp among a group of peers that participate in a specific Inter-Destination Multimedia Synchronization session. Our approach maintains low overhead and allows to synchronize an arbitrary number of peers. We evaluate our approach against related work and discuss its properties in detail.

In the second part of this thesis we investigate how to actually carry out the synchronization by overcoming the identified asynchronism between the reference playback timestamp and the current playback timestamp at each peer with respect to the Quality of Experience. Related work proposes to naïve approaches such as skipping multimedia content and pausing the multimedia playback without considering the Quality of Experience. We propose to use Adaptive Media Payout which foresees an increase or decrease of the playback rate in order to achieve synchronization. Therefore, we investigate the impact of Adaptive Media Payout on the Quality of Experience. We further investigate how content sections have to be select such that the impact of Adaptive Media Payout on Quality of Experience is minimized. The insights that are obtained by these investigations are used to formulate a constrained optimization problem that allows to carry out the actual synchronization of the multimedia playback at each peer.



# Kurzfassung

Gemeinsam Multimediainhalte zu erleben ist bereits zu einer gesellschaftlichen Aktivität geworden. Seit der Erfindung des Fernsehgeräts gibt es gemeinsame TV Abende mit Freunden und/oder Familie. Durch das Entstehen von sozialen Netzen (z.B., Facebook) und den daraus resultierenden Möglichkeiten zur Echtzeitkommunikation immer und überall, bewegen sich diese traditionellen TV Sessions (im Wohnzimmer auf der Couch) immer mehr in Richtung eines (geographisch) verteilten Fernseherlebnisses. Die Nutzer wollen bei diesem verteilten Erlebnis jedoch die selbe Qualität und Immersion erfahren wie bei einem traditionellen TV Abend. Insbesondere müssen Multimediainhalte synchron wiedergegeben werden. Dieser neue Typ von Synchronisation wird als Inter-Destination Multimedia Synchronization bezeichnet.

Diese Dissertation untersucht wie man Inter-Destination Multimedia Synchronization mittels eines selbstorganisierten und verteilten Ansatzes, welcher den Referenzzeitpunkt für die Synchronisation berechnet, ermöglicht werden kann. Es genügt jedoch nicht nur einen Referenzzeitpunkt für eine Gruppe von Peers zu bestimmen. Sobald ein Referenzzeitpunkt gefunden wurde, kann sich jeder Peer zu diesem synchronisieren. Die Synchronisierung wirkt sich direkt auf die Qualität der Erfahrung (engl. Quality of Experience) aus.

In dem ersten Teil dieser Dissertation erläutern wir wie Inter-Destination Multimedia Synchronization mittels MPEG-DASH ermöglicht wird. Des Weiteren definieren wir ein sogenanntes Inter-Destination Multimedia Synchronization Sessionobjekt, welches synchronisationsspezifische Informationen speichert. Das Berechnen des Referenzzeitpunktes wird in zwei Etappen behandelt. Während der ersten Etappe wird ein Peer-to-Peer Application Overlay Netzwerk erstellt und es wird versucht mittels eines "Educated Guess" das Abspielen der Multimediainhalte zu starten. Die zweite Etappe ist für das Berechnen des Referenzzeitpunktes zuständig. Dies geschieht mittels eines verteilten Algorithmus der das erstellte Peer-to-Peer Netzwerk verwendet. Unser Ansatz verursacht wenig Overhead und ermöglicht somit die Synchronisation einer großen Anzahl an Peers.

Der zweite Teil dieser Dissertation beschäftigt sich mit dem Erreichen des Referenzzeitpunktes. Hierzu postulieren wir, die Abspielgeschwindigkeit adaptieren (Adaptive Media Playout), anstatt einfach zu pausieren oder zu dem Referenzzeitpunkt zu springen. Der naive Ansatz hat laut bereits durchgeführten Studien erheblichen negativen Einfluss auf die Qualität der Erfahrung. Wir untersuchen den postulierten Adaptive Media Playout Ansatz mittels subjektiven Studien und zeigen wie dieser Ansatz technisch umgesetzt werden kann. Des Weiteren zeigen wir, wie Adaptive Media Playout unter Einbeziehung von, einfach zu berechnenden, Inhaltseigenschaften zu sehr guten Ergebnissen bezüglich der Qualität der Erfahrung führen kann.



---

# 1 Introduction

“The important thing is not to stop questioning.”

---

— Albert Einstein, 1879 - 1955

## 1.1 Motivation

During the past decade social communication has evolved extensively through the introduction of platforms such as Facebook, Twitter, and Google+. These cutting-edge forms of social interaction demand new requirements on the underlying technologies that help us create, distribute and experience multimedia content. The traditional TV scenario as we know it, watching TV with friends and family, is becoming increasingly location independent with people wanting to experience multimedia together even if they are geographically distributed. This new form of *togetherness* utilizes real-time communication channels such as text, voice, or even video telephony in order to share the experience.

The demand for these new distributed social experiences requires new research and technologies. Picking up the use case of watching TV together with friends, colleagues, and the family while being geographically separated requires a certain synchronization between the participating entities in order to provide a high immersion and a strong feeling of togetherness. Recent research has shown that with an increase in asynchronism between the participating entities the togetherness/immersion decreases and can even lead to annoyance [1]. For example, Figure 1.1 depicts two friends that want to watch a soccer match together while being geographically distributed and having a real-time voice communication using Skype. Due to the

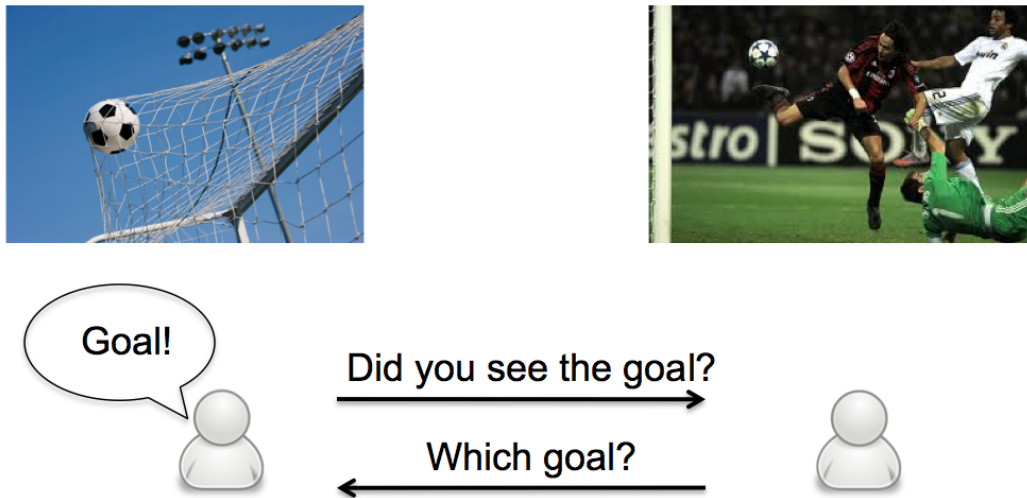


Figure 1.1: IDMS use case example.

missing synchronization between the multimedia playbacks of their peers, one experiences the goal before the other one and communicates the goal instantly via Skype. This asynchronism of the multimedia playback leads to a decrease in the Quality of Experience (QoE), more specifically, the asynchronism has an impacts the feeling of togetherness and annoyance as investigated in [1]. The required synchronization for such use cases is called Inter-Destination Multimedia Synchronization (IDMS).

The authors of [1] further provide the first preliminary thresholds for IDMS. They state that the asynchronism between the multimedia playback of the users shall be below one second. In IDMS it is further assumed that the asynchronism between users is, in general, by an order of magnitude higher than the delay of the voice or text channel. Especially, voice over IP applications have only a few milliseconds delay [2, 3] whereas, the multimedia playback of geographically distributed peers may suffer from several seconds of asynchronism.

The SocialSensor EU project aims at providing new means of aggregating, presenting, and experiencing user created media by analyzing popular social media platforms (e.g., Facebook, Twitter, and Google+). The research presented in this thesis has been conducted during the SocialSensor EU project, specifically in the work packages (WP) three and five [4]. WP three provides new means of searching, distributing and experiencing multimedia in unstructured peer-to-peer (P2P) overlay networks using MPEG - Dynamic Adaptive Steaming (MPEG-DASH) [5]. Figure 1.2 depicts the



components of the envisioned SocialSensor Semantic Middleware Suite (SSMS). This thesis covers the *Video Data Streaming* component and to a certain extent the *P2P interaction* component in the sense of our self-organized and distributed IDMS approach. The other components of the SSMS are: the *Semantic Service Coordination* component deals with providing semantic search, selection, planning, and replication with respect to services in unstructured P2P overlay networks; the *Semantic Data Retrieval* component handles the intelligent caching and retrieval of distributed semantic data; the *Predictive Data Caching* tries to predict when and how long disconnections may occur for a specific peer based on the already learned network characteristics. The SocialSensor EU project pursues two defined use cases, the *infotainment use case* which demands new means of presenting, experiencing, and searching user created multimedia content and the *news use case* which demands new ways of aggregating user created news and stories. Figure 1.3 depicts the simplified infotainment use case of SocialSensor tailored to fit the focus of this thesis. The use case foresees the sharing of user generated multimedia content either live or on-demand. In the context of an event (i.e., the Thessaloniki Film Festival, or any other film or music event) users want to record and share the impressions and experiences from the event with friends. These impressions shall be shared in a synchronized manner using IDMS while having a real-time voice communication between the users. The work presented in this thesis aims at the infotainment use case by providing IDMS using pull-based streaming and an unstructured P2P overlay for providing the possibility of experiencing multimedia content in a synchronized manner.

A very recent service, that has been introduced a few months ago that provides IDMS, is Google Hangouts [7]. Besides the possibility of having a distributed video chat with friends, the family, and colleges, it provides the possibility to watch multimedia content hosted by YouTube in a synchronized way. Google Hangouts employs WebRTC which provides an API for real time communication [8]. The synchronization of the multimedia playback of the participating users is done by skipping and pausing the playback. Nevertheless, there is no sophisticated synchronization protocol employed to keep the peers synchronized such that the users would not notice that there is a synchronization happening. If a user pauses the multimedia playback, the playback of all the other users in a Google Hangouts session is paused. We further

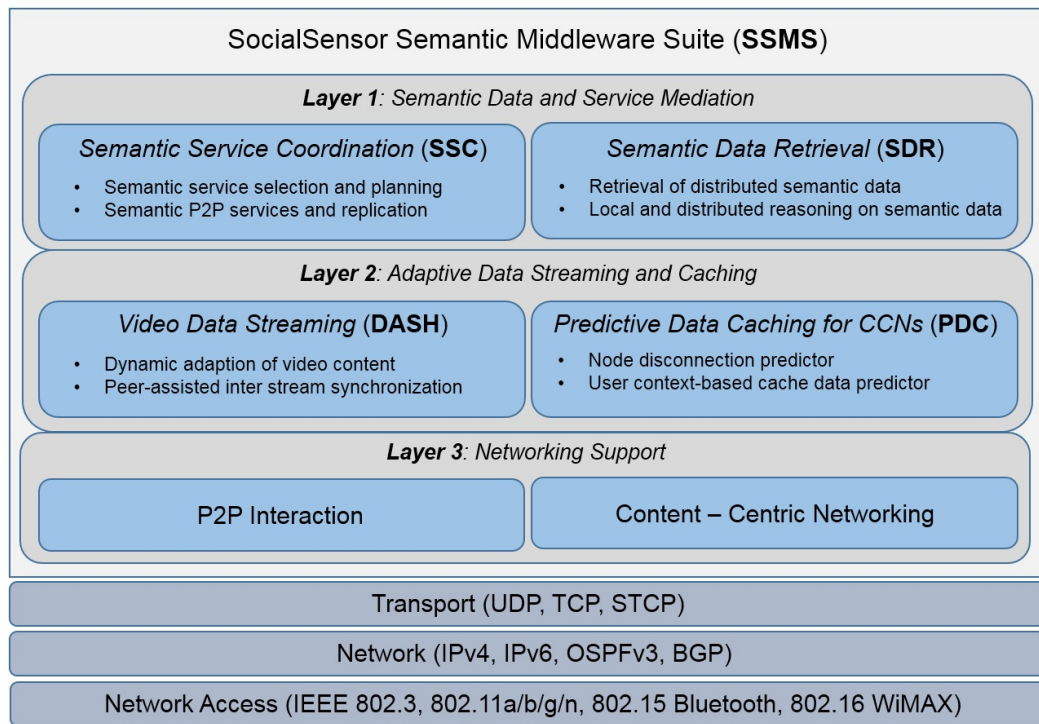


Figure 1.2: WP 3 components and modules of the SocialSensor Semantic Middleware Suite [6].

differ in the general principle of our approach. We assume completely autonomous peers which are not controlled by a central instance nor shall the other peers adhere to (trick mode control) actions of a single peer (which supervises the others) in an IDMS session. The reference playback timestamp shall be determined in a collaborative and fair manner. The actual adjustment of the multimedia playback to the agreed reference playback timestamp shall be carried out by each peer separately and such that the impact on the QoE is minimized.

Current state of the art IDMS solutions are tailored around push-based streaming such as RTP/RTCP [9] and most of these solutions demand a central instance that deals with exchanging timing and control information [10]. Calculating the reference among a group of peers is one aspect but, the other very important aspect, with emphasizes on the QoE, is how the synchronization is actually carried out at the peers. State of the art algorithms just pause or skip multimedia content in order to align the multimedia playback of all peers to the reference timestamp. In this thesis we will introduce an IDMS approach that calculates the reference timestamp in a distributed

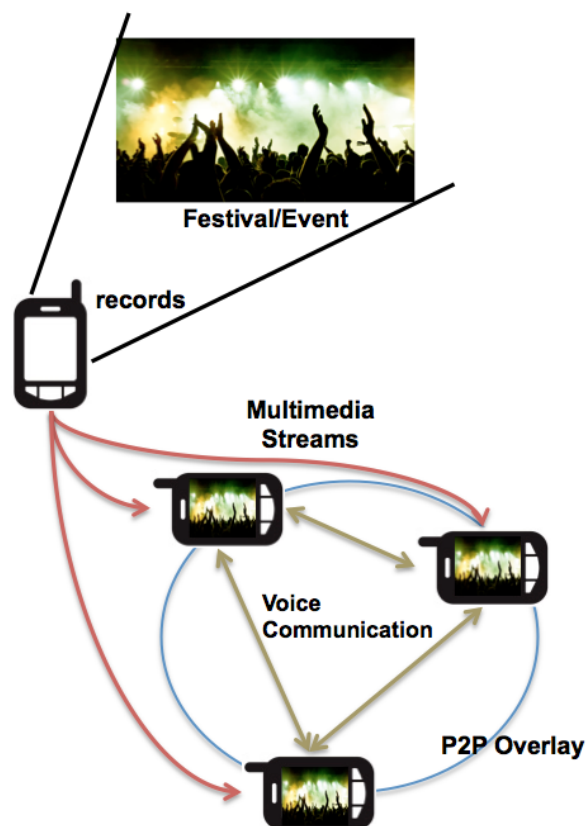


Figure 1.3: Infotainment use case in the context of SocialSensor and IDMS.

and self-organized manner. We will further investigate how to carry out the actual synchronization such that it is barely perceived by the user by taking a look at content features and conducting subjective quality assessments using crowdsourcing.

## 1.2 Research Objectives

In this thesis we focus on the use case of providing IDMS for on-demand and live multimedia streaming. In particular, we adopt MPEG-DASH [5] for IDMS and introduce a novel distributed algorithm that calculates the reference playback timestamp among the participating entities in a self-organized manner. We further investigate how to overcome the identified asynchronism by increasing or decreasing the playback rate of the multimedia stream.

The research objectives of this thesis are

- (1) to define the notion of an IDMS session.
- (2) to introduce session management for IDMS in the context of MPEG-DASH.
- (3) to provide an algorithm that identifies the asynchronism between the multimedia playback of different users in an IDMS session in a distributed and self-organized manner.
- (4) to evaluate the introduced distributed and self-organized approach.
- (5) to investigate the possibility of carrying out the synchronization in terms of adaptively changing the media playback by increasing or decreasing the playback rate of the multimedia playback.
- (6) to assess whether content features allow to decrease the impact of increasing or decreasing the playback rate on the QoE.
- (7) to analyze and quantify the impact of Adaptive Media Playout (AMP) on the QoE.
- (8) to utilize content features for dynamically selecting content sections that are appropriate to overcome the identified asynchronism using AMP.

This thesis will introduce and define the notion of an IDMS session (1). A state of the art multimedia streaming technology, MPEG-DASH, is utilized and adopted to realize session management for IDMS (2). It will be shown how an IDMS session can be signaled by using the Media Presentation Description (MPD) of MPEG-DASH and how newly arriving peers are added to an existing session. Furthermore, the creation of a peer-to-peer overlay network will be introduced in order to provide the basis for agreeing on a reference playback timestamp.

Agreeing on a reference playback timestamp and calculating the asynchronism of the multimedia playback of each participating peer is done by a novel distributed and self-organized algorithm which is decoupled from the employed multimedia streaming technology (3). The algorithm uses a probabilistic data structure in order to keep track which peers have already contributed to the reference playback timestamp,

but still providing an unique and deterministic computation of the reference playback timestamp among the peers in a peer-to-peer overlay network. The introduced distributed algorithm is evaluated with respect to overhead generated and the time needed until all participating peers have agreed on a reference playback timestamp (4).

Having agreed on a reference playback timestamp and having each peer calculated its asynchronism, the question arises how each peer is going to carry out the synchronization in order to overcome the identified asynchronism. This thesis will investigate the usability of AMP for carrying out the synchronization by conducting a subjective quality assessment (5). Furthermore, we introduce an algorithm that uses content features for selecting appropriate content section for which the multimedia playback rate is increased or decreased. This algorithm shall provide insights whether using content features provide a better QoE (6).

Finally, this thesis investigates the impact of AMP on the QoE by introducing quantitative measures (7). These measures are used to quantify the distortion caused by increasing or decreasing the playback rate in the audio and video domain, respectively. We derive a non-linear parametric utility model by correlating these measures with the results of a subjective quality assessment. This utility model allows to estimate the coefficient of degradation when AMP is employed. The resulting utility model is used to derive a generalized optimization problem called dynamic AMP (8).

### 1.3 Contributions

This thesis comprises several scientific contributions in the fields of QoE, distributed and self-organized systems, AMP and multimedia streaming which have been published in the proceedings of international high quality conferences and workshops, journals, project deliverables and in book chapters.

The research on adopting MPEG-DASH for IDMS and the distributed and self-organized negotiation on a reference playback timestamp has been published in [11]. An implementation using the proposed algorithm and MPEG-DASH has been published in [12], showing the practicability of our approach. Previous work utilizes

RTP/RTCP [9] for achieving IDMS and does not consider unicast and multicast separately [10]. Especially, in today's open Internet IP multicast is not in place. Therefore, we adopt MPEG-DASH for IDMS and show how IDMS can be achieved efficiently using a distributed control scheme.

The research on utilizing AMP for carrying out the synchronization has been published in [11, 13, 14] and [15]. AMP was previously used to smoothen the multimedia playback by maintaining a certain buffer fill state and avoiding buffer under-runs or overflows by increasing or decreasing the playback rate of the multimedia content. We have shown that using content features for selecting appropriate section where AMP is applied can mask the negative effects of increasing the playback rate in the audio and video domain. Other IDMS solution do not take care of the actual synchronization of the multimedia playback and just employ pausing and skipping of multimedia content in order to achieve synchronization.

In order to achieve the research objectives of this thesis a Web-based subjective assessment platform and an application for generating MPEG-DASH compliant live content has been developed. Research regarding these applications has been published in [16, 17] and [18].

## 1.4 Structure

This thesis is structured as follows. Chapter 2 gives an overview of the related work in the areas of IDMS, AMP, MPEG-DASH, subjective quality assessment methods, and subjective quality assessments using crowdsourcing.

Chapter 3 deals with the distributed and self-organized negotiation on a reference playback timestamp and, therefore, will introduce a novel distributed algorithm. Furthermore, MPEG-DASH is adapted for signaling session information and building a peer-to-peer overlay network. The proposed algorithm is compared to a baseline algorithm using unicast (and multicast) regarding the overhead generated and time needed for negotiating on a reference playback timestamp.

In Chapter 4 the usability of AMP for carrying out the synchronization is investigated and whether content features can be utilized to mask the negative effects caused

by AMP. This is done by conducting subjective quality assessments. Furthermore, measures are introduced which allow the quantification of the distortion in audio and video caused by AMP. An utility model for estimating the QoE for certain values of the distortion measures is derived and instantiated. This utility model is later on used to formulate an optimization problem in order to find content sections which minimize the impact of AMP on the QoE.

Applications developed during the course of this PhD project and during the SocialSensor project are presented in Chapter 5. This comprises a web-based subjective quality assessment platform and the mobile DASHEncoder which generates MPEG-DASH compliant multimedia streams on Android devices.

Finally, Chapter 6 concludes this thesis and correlates the research objectives with the work presented in this thesis. Furthermore, future work items are discussed and an outlook is provided.





# 2 Technical Background and Related Work

## 2.1 Multimedia Synchronization

In the last decade three major types of synchronization have emerged with respect to multimedia synchronization. The first type of synchronization that has been subject to intensive research is intra-stream synchronization. Figure 2.1 depicts the actual aim of intra-stream synchronization [19]. Intra-stream synchronization deals with the presentation of media units (MUs) of a single media stream and shall maintain the time dependency between consecutive MUs. If we consider a network, where the change in delay corresponds to jitter and, thus, if no mechanisms are employed at the multimedia playback the video/audio frames received will suffer from varying presentation times. Therefore, the playback buffer was introduced which accounts for the jitter and provides the possibility that the received frames can have their presentation times as intended. The impact of jitter on the resulting QoE has been studied by [19], [20], [21], and [22]. If no playback buffer is used, jitter at the multimedia playback, will have an immediate impact on the QoE if it exceeds the frame rate of the video [21].

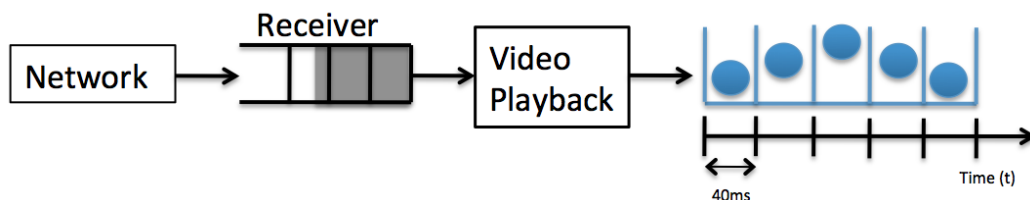


Figure 2.1: Intra-stream synchronization maintaining the dependency between multimedia frames of a single media stream.

Another well studied type of synchronization is the inter-stream synchronization, which is responsible for maintaining the causality between two or more streams depicted in Figure 2.2. Inter-stream synchronization has been very well studied in [19]

and [21]. Amongst others, the authors of [19] and [21] have studied the influence of asynchronism between audio and video on the QoE and derived upper limits for the asynchronism of audio and video. Therefore, a subjective quality assessment was conducted with different video sequences and a variety of skews in milliseconds between the audio and video playback. The subjective quality assessment revealed that a negative skew (audio behind video) lower than  $-80$  ms or having a positive skew (audio ahead video) higher than  $80$  ms is already significantly perceived by the users. If the synchronization between audio and video are within these limits the users could not significantly perceive the introduced skew. This type of synchronization is called *lip-synchronization*.

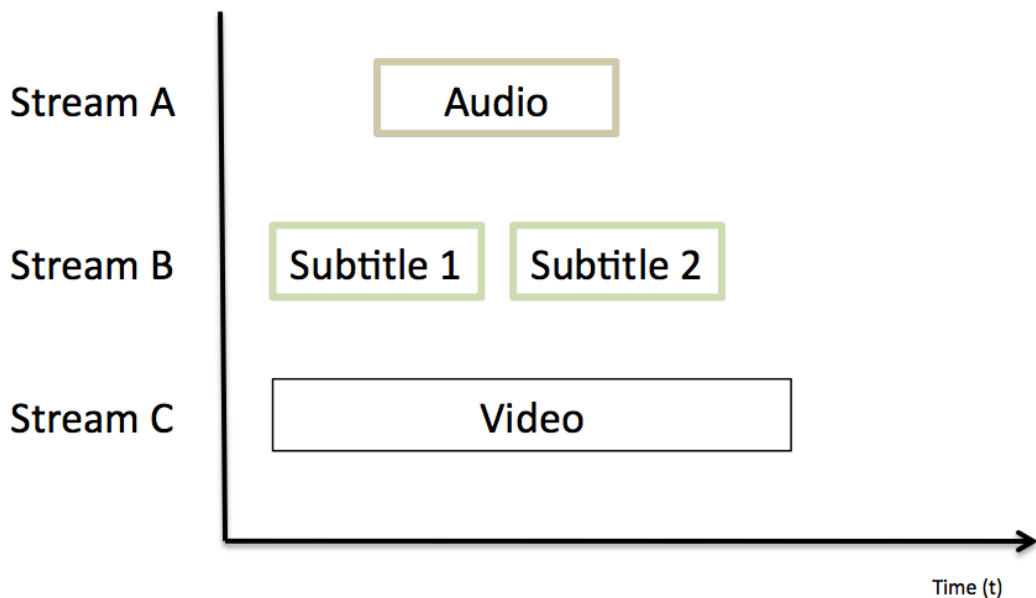


Figure 2.2: Inter-stream synchronization preserving synchronization between different modalities.

Recently, a new type of synchronization has emerged, which extends the already known synchronization types by another dimension, the so-called Inter-Destination Multimedia Synchronization [23]. In comparison to inter-stream synchronization where a stream (typically audio is used as synchronization reference) is select as synchronization reference, IDMS aims at synchronizing the multimedia playback of two or more peers consuming **the same multimedia content**. In this thesis **the same multimedia content** refers to exactly the same multimedia content with the same

encoding parameters and with identical presentation timestamps and video frames. In order to provide a pleasant viewing experience intra- and inter-stream synchronization have to be in place. Figure 2.3 depicts peers that may receive the multimedia content via different distribution channels but maintain IDMS. Besides intra-stream and inter-stream synchronization IDMS maintains the synchronization between the peers which may be geographically distributed (inter-destination). The thresholds for the synchronization are assessed in [1]. The authors of [1] conducted a subjective quality assessment where they tried to assess the synchronization thresholds for IDMS using different communication channels such as voice over IP and text chatting. The results of the subjective quality assessment show that for active text chatters the upper threshold is at about two seconds until a significant difference is noticed by the users. For active voice chatters the threshold is at about one second until the asynchronism is significantly perceived. Please note, that the assessed threshold depends strongly on the multimedia content used as stimulus because if there is a lot of action and motion these thresholds may not be anymore valid. Therefore, IDMS should try to synchronize the multimedia playback of the peers as accurate as possible. The IDMS approach we are going to introduce is able to synchronize the multimedia playback of the peers very accurately up to only a few milliseconds skew between the peers.

In comparison to intra- and inter-stream synchronization, IDMS demands that the following key mechanisms are in place:

- **Session management** which is responsible for managing the session to which peers belong;
- **Signaling of timing and control information** allows the exchange of timing information and, if necessary, control information between the peers;
- **Negotiation on a reference playback timestamp** deals with the selection of a playback timestamp within a session to which the peers have to synchronize their playback;
- **Carrying out the synchronization** overcomes the identified asynchronism by modifying the multimedia playback of each peer.

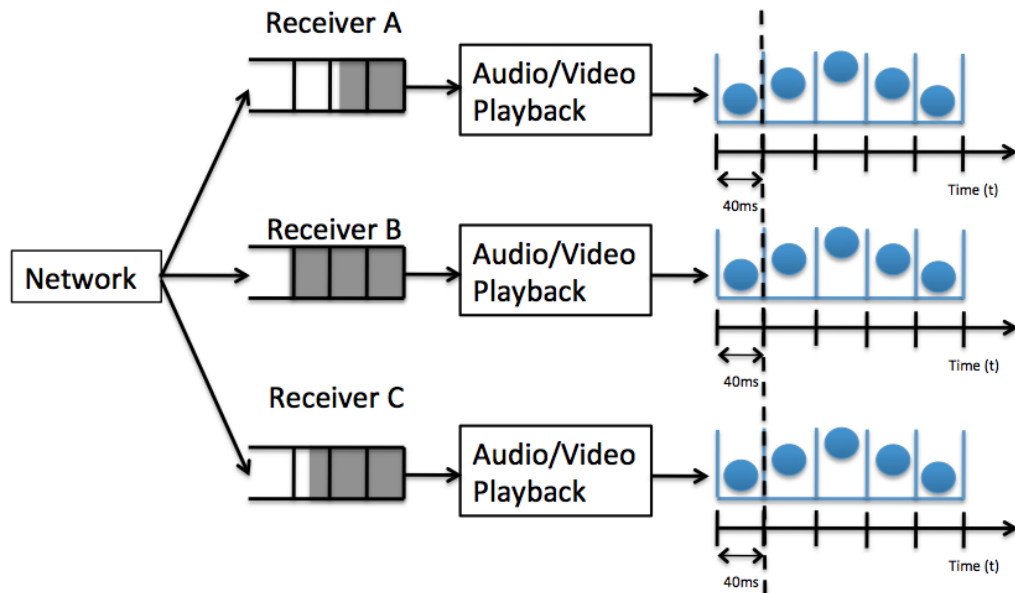


Figure 2.3: Inter-Destination Multimedia Synchronization has to provide synchronization among different distribution channels.

An IDMS system is only able to provide synchronization between the peers if and only if the mentioned mechanisms are in place. Each of these mechanisms is discussed in this thesis and novel research will be presented especially regarding the signaling of timing and control information, negotiating on a reference playback timestamp, and on carrying out the synchronization. Timing information consists of information about a peer's playback state (e.g., PTS). Control information consists of detailed instructions on how a peer shall modify its playback (e.g., skip multimedia content, pause, increase the playback rate, and decrease the playback rate).

## 2.2 Inter-Destination Multimedia Synchronization Schemes

The common assumption of most IDMS solutions is that clocks are already synchronized using the Network Time Protocol (NTP) [24] or the Precision Time Protocol (PTP) [25]. Most of the schemes only deal with the signaling of timing and control information to achieve IDMS among the participating peers [10, 23]. Current IDMS

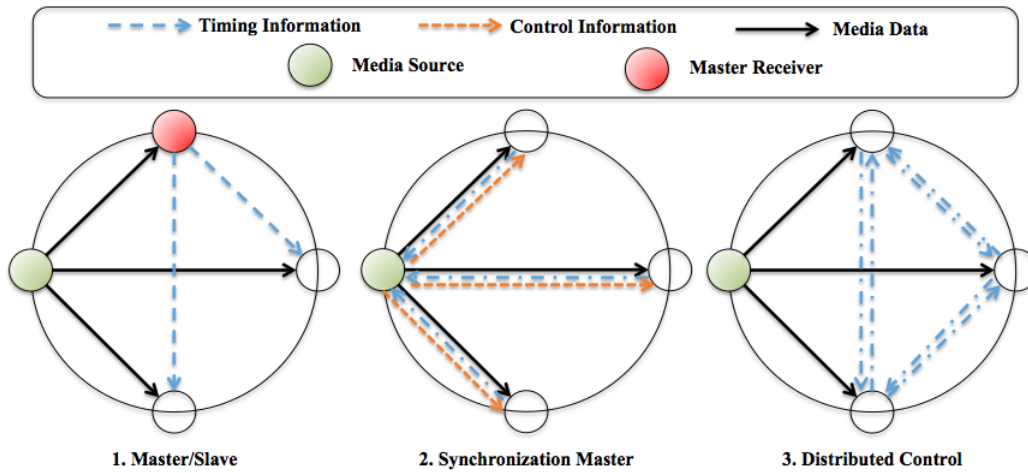


Figure 2.4: IDMS Schemes: Master/Slave Scheme, Synchronization Master Scheme, and Distributed Control Scheme.

solutions are very use case tailored, e.g., for multiplayer online games [26] or when participating in collaborative work [27]. In general, the existing solutions can be clustered into three different basic schemes [10, 23] depicted by Figure 2.4:

**Master-/Slave scheme (MS)** (cf. Figure 2.4.1): This scheme uses a dedicated master for signaling timing information. Figure 2.4.1 depicts a MS with a master (red node), the media source (green node), and two peers. The master may be elected among the participating peers or determined by the media source. If the selected master leaves the sessions a new master has to be selected or elected among the clients. Furthermore, the peers have to *trust* the master that control and timing information received by it is correct. The advantage of this scheme is that the instance which is responsible for signaling timing information is elected among the participating peers. An approach that follows the MS scheme was first proposed in [28]. The approach presented in [28] builds on top of multicast and the synchronization protocol employed also maintains intra- and inter-stream synchronization. The MS scheme suffers from scalability issues because timing information is exchanged between each peer and the selected master which may lead to bandwidth shortages using unicast and a certain number of peers. Therefore, most solutions that rely on a MS scheme require multicast.

**Synchronization Master Scheme (SMS)**(cf. Figure 2.4.2): is a centralized

approach where the synchronization is controlled by a synchronization master which may be the media source or a dedicated synchronization node (not a peer that consumes multimedia content). Figure 2.4.2 depicts the SMS with three peers and a media source (green node) that has the additional task of being the synchronization master. The synchronization master collects timing information and sends timing and control information to which the peers have to adhere. This approach suffers from scalability issues because a central instance can only handle a certain number of peers. Furthermore, if more than one synchronization master is used there has to be dedicated communication between them. The advantage compare to the MS scheme, is that the peers can trust the synchronization master because it is in control of the content provider. A SMS scheme is presented in [29], which adopts the local lag and time warp algorithm compensating for media playback inconsistencies [30]. Further SMS approaches are presented in [31] and [32]. Both approaches extend the Receiver and Sender Reports (RR and SR) defined within the RTP/RTCP protocol in order to carry the necessary timing and control information [9]. Furthermore, the extensions to the RTP/RCP protocol are standardized under RFC 7272 [33]. RFC 7272 supports multicast and unicast.

**Distributed Control Scheme (DCS)**(cf. Figure 2.4.3): uses distributed protocols to determine a common playback timestamp to which the peers may synchronize. Therefore, timing information is exchanged in a peer-to-peer manner among the peers. This scheme has the highest robustness in terms of overall failure probability. Figure 2.4.3 depicts a DCS with a media source (green node) and three peers which exchange only timing information in order to agree on a reference playback timestamp. The content provider has only to provide the multimedia content. Nevertheless, the peers have to trust each other in terms of faulty behavior. Furthermore, Network Address Translators (NATs) may cause problems, especially if the peers are behind symmetric NATs. [34] presents the *iNEM4U* approach where a DCS is used to achieve IDMS among heterogeneous network infrastructures. Furthermore, it introduces *iSession* for the session management, which provides an XML description of each session including the users and/or peers, content source, and other service-specific data. A very recent DCS for achieving IDMS which uses an extended version of RTCP messages is presented in [35]. The proposed DCS is designed on top of RTP/RTCP and

peers are assigned to a specific cluster. Within a cluster the peers regularly exchange RTCP RR packets including playback timestamps in order to achieve intra-cluster synchronization. This solution uses multicast for exchanging the RTCP RR packets (following the principles introduced by RFC 7272) between the peers. Another DCS is presented in [26] which uses multicast and provides intra-stream synchronization.

Most approaches are built on top of RTP/RTCP. Therefore, they cannot be decoupled from the underlying streaming technology. Our DCS approach (presented and described in Chapter 3) is not directly coupled to the streaming technology employed. The pull-based streaming, in particular MPEG-DASH, is used as an enabler and to store the session information that is needed to build the application layer peer-to-peer overlay. Therefore, our DCS approach may be used in conjunction with other streaming technologies. The common denominator of most of the DCS is that they use multicast which our approach does not require because we do not assume that multicast is in place. Furthermore, our DCS approach does not require a fully connected network where all peers can (directly) communicate with each other.

Besides the communication between the peers using a centralized or decentralized scheme, another very important decision in IDMS systems, specifically in distributed systems, is the calculation of the reference playback timestamp to which the peers synchronize their media playback. In [36] three different reference selection policies are discussed:

- **Synchronization to the slowest client**, i.e., the client that is displaying the lowest frame number among the group of peers;
- **Synchronization to the fastest client**, i.e., the client that is displaying the highest frame number; and
- **Synchronization to the average**, i.e., to the average frame number among a group of peers.

Each of these policies has its advantages and disadvantages which are discussed with respect to our DCS in Chapter 3. Nevertheless, the average ( $\bar{X}$ ) is the fairest selection among these three policies from a mathematical point of view because if

we aim on minimizing the error  $\epsilon$  between the reference playback timestamp and all current playback timestamps of the peers expressed by  $\epsilon = \frac{1}{2} \sum_{i=1}^N (x_i - \bar{X})^2$ ,  $N$  denotes the number of peers participating in an IDMS session and  $x_i$  denotes the current playback timestamp of peer  $i$ , where  $\forall x_i, 1 \leq i \leq N : x_i \in \mathbb{R}_+$ . With  $\frac{\partial \epsilon}{\partial \bar{X}} = \sum_{i=1}^N (x_i - \bar{X}) \stackrel{!}{=} 0$ , we have  $\bar{X} = \frac{1}{N} \cdot \sum_{i=1}^N x_i$ . Therefore, the average as reference point implies the lowest error. The mentioned policies assume that the playback is paused or audio/video frames are skipped to compensate for asynchronism which is in contrast to our approach (presented and discussed in Chapter 4, where the playback rate is dynamically increased or decreased).

## 2.3 Adaptive Media Playout

In addition to the selection of the reference and the type of the control scheme, there is ongoing work on how the synchronization should be carried out at each peer. Currently, the common denominator of the discussed IDMS solutions is that compensating the identified asynchronism is done by skipping or pausing the multimedia playback. In [37] the effect of stalls during media playback was subjectively assessed. The results indicate that the Mean Opinion Score (MOS) degrades with an increase in stalls during media playback. Thus, using stalls to overcome asynchronism may lead to a low QoE for the users. AMP was introduced to overcome these shortcomings, and the approach described in [38] deals with increasing or decreasing the playback rate of the multimedia playback without considering the influence on the QoE of the user. Furthermore, we assume that the playback rate of both the audio and the video domain are altered simultaneously; i.e., preserving the inter-stream synchronization between audio and video.

Initially, AMP was thought to be used to compensate for buffer under-flows or over-flows by decreasing or increasing the media playback rate to allow the stabilization of the playback buffer [39]. The impact of error prone networks, modeled by a Markov-Chain [40], on the continuity of the multimedia playback has been investigated in [41]. An approach that tries to maximize the quality of the multimedia in terms of Peak Signal-to-Noise Ratio (PSNR) [42] when using AMP, is presented



in [43]. The authors of [44] modeled the adaptation of the multimedia playback rate depending on the variance of the buffer fill state. In [45] the buffer fill state was used to decide whether the playback rate should be increased or decreased. All these schemes try to avoid buffer under-flows or buffer over-flows in an error prone environment taking properties of the buffer over time into account. Nevertheless, neither the impact of increasing or decreasing the playback rate on the QoE has not been taken into account, nor content features have been considered for selecting specific content section that may mask the effects of AMP for the human perception.

In [46] the authors pursuit the approach of combining the decision based on the buffer fill state with content features. Therefore, they use the spatial resolution of the video frames to influence the adaptive playback decision in wireless video streaming. An algorithm that tries to reduce the impact of AMP on the QoE is proposed in [47]. The proposed algorithm uses motion vectors and the buffer fill state for deciding for which frames the playback rate is increased or decreased.

Further research regarding the impact of AMP on the QoE or altering the temporal domain of the multimedia presentation has been investigated in [48]. The authors show that the QoE degrades with an increase in the number of frame drops (which corresponds to skipping multimedia content). Furthermore, a non-linear utility model is proposed that allows to estimate the resulting QoE for a certain frame rate and the average maximal motion of a frame. This model targets on the reduction of the frame rate by dropping frames while not altering the duration of the video. In [22] altering the frame rate by decreasing the frame rate has been subjectively assessed for short periods of video for low bit rate videos (without audio). Interestingly, the QoE does not degrade linearly with a decrease in the frame rate. Furthermore, in [49] the impact of reducing the frame rate for 100%, 50%, and 25% of the video sequence was investigated and revealed that *small deviations* from the nominal playback rate are not significantly perceived by the users.

Our findings will validate the findings that have been presented by related work and will extend it by taking audio and video into account. Most of the related work has investigated the audio and video domain separately. We will take a look on how the QoE is influenced if both modalities are present at the same time and if they are

altered simultaneously. This will provide new insights, especially for the interplay of the audio and video domain.

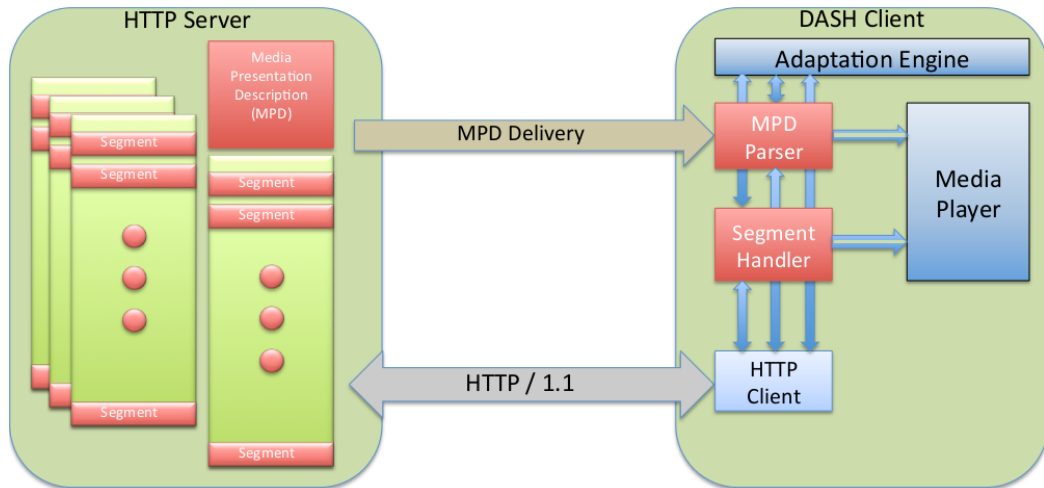


Figure 2.5: Building Blocks of MPEG-DASH [50].

## 2.4 MPEG-DASH

MPEG-DASH has been ratified and standardized by the ISO/IEC with grant number 23009-01 [5] which is a pull-based streaming technology. It is the standardized answer on the industrial solutions like Apple's HTTP-Live-Streaming (HLS) [51], Microsoft's Smooth Streaming [52] and Adobe's HTTP Dynamic Streaming (HDS) [53]. MPEG-DASH pursues the goal of providing a standardized way of describing scalable multimedia content such that the peers can decide which representation of the multimedia content has to be downloaded with respect to Quality of Service parameters such as, e.g., available bandwidth, delay and playback buffer fill state. Therefore, it tries to push the complexity from the server to the client which contradicts the traditional push-based streaming approach.

Figure 2.5 depicts the building blocks and the architecture of MPEG-DASH. The standardized components are depicted in red. The blue components are left open and shall provide the possibility for proprietary solutions or research. The MPEG-DASH standard provides a standardized way of describing the multimedia content and if available its different representation. This description is called Media Presentation

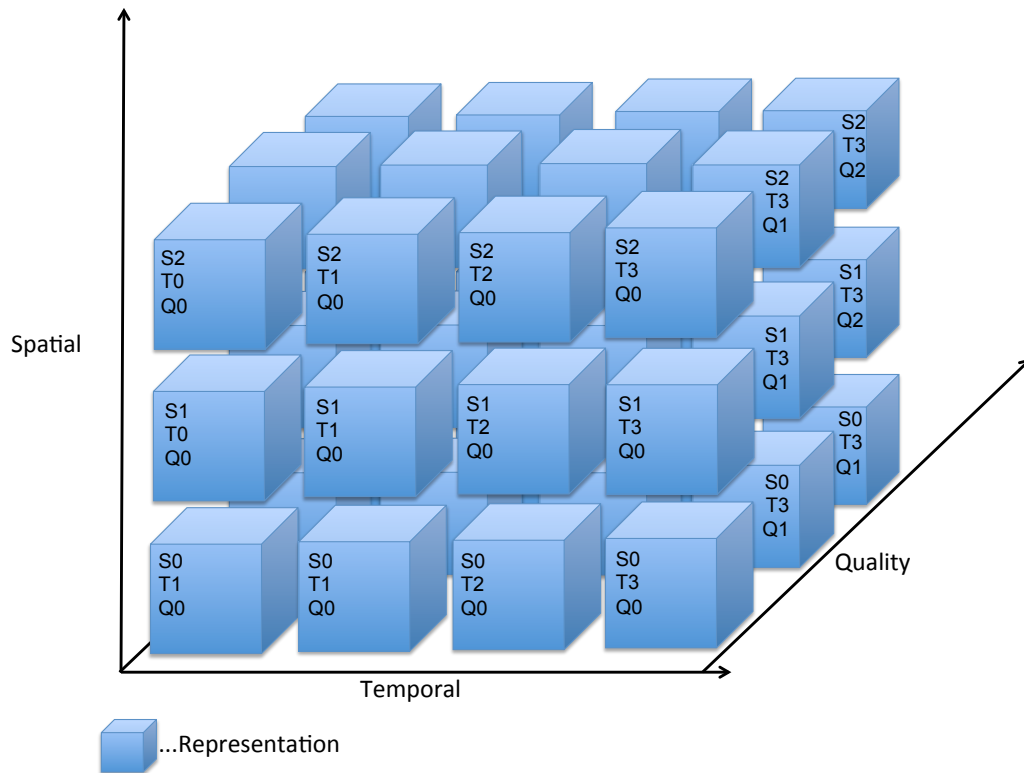


Figure 2.6: Adaptation space with four different spatial and temporal representations and three representations in the quality domain, respectively.

Description (MPD) and is defined using XML Schema. HTTP/HTTPS 1.0/1.1 is used as transport protocol for the multimedia content. MPEG-DASH demands that the multimedia content is present as so-called *segments*. These segments are equally sized in time (e.g., one seconds, two seconds, four seconds, eight seconds, or ten seconds) each of which are separately requested using HTTP/HTTPS. One container format that MPEG-DASH foresees is the ISO Base Media File Format (IBMF) [54] with the extension of separating a contiguous multimedia file into separate files. This is achieved by dividing the contiguous multimedia file into so-called fragments which are separately stored in files. These segments may be self-contained (the necessary decoding information is included in each of the segments) or a dedicated initial segment is provided that contains all the necessary decoding information. MPEG-DASH further supports elementary streams and MPEG-2 Transport Streams [55].

As already mentioned the MPD describes which versions/representations of the

multimedia content are available at the server side. Figure 2.6 depicts the three common adaptation/scalability dimensions, e.g., spatial, temporal and quality scalability. Spatial scalability refers to the resolution of the video domain of the multimedia content. Temporal scalability refers to the temporal resolution of multimedia content in terms of frames per second (fps). Scalability in the quality domain refers to an increase in quality of the different multimedia streams (e.g., audio and/or video) with respect to a certain measure (e.g., PSNR, Structured Similarity, bit-rate).

```

1 <?xml version="1.0"?>
2 <MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT1.5S" type="static"
   mediaPresentationDuration="PT1H26M" profiles="urn:mpeg:dash:profile:isoff-main:2011">
3 <BaseURL>http://example.com/content/</BaseURL>
4 <Period duration="PT1H26M">
5 <AdaptationSet segmentAlignment="true">
6 <Representation id="1" mimeType="video/mp4" codecs="avc1.42c00d" width="320" height
   ="240" frameRate="30000/1001" sar="1:1" startWithSAP="1" bandwidth="101355">
7 <SegmentList timescale="1000" duration="2000">
8 <Initialization sourceURL="100kbps_2sec_segmentinit.mp4"/>
9 <SegmentURL media="100kbps_2sec_segment1.m4s"/>
10 ..
11 </SegmentList>
12 </Representation>
13 <Representation id="2" mimeType="video/mp4" codecs="avc1.42c00d" width="320" height
   ="240" frameRate="30000/1001" sar="1:1" startWithSAP="1" bandwidth="150705">
14 <SegmentList timescale="1000" duration="2000">
15 <Initialization sourceURL="150kbps_2sec_segmentinit.mp4"/>
16 <SegmentURL media="150kbps_2sec_segment1.m4s"/>
17 ...
18 </SegmentList>
19 </Representation>
20 <Representation id="3" mimeType="video/mp4" codecs="avc1.42c01e" width="480" height
   ="360" frameRate="30000/1001" sar="1:1" startWithSAP="1" bandwidth="201237">
21 <SegmentList timescale="1000" duration="2000">
22 <Initialization sourceURL="200kbps_2sec_segmentinit.mp4"/>
23 <SegmentURL media="200kbps_2sec_segment1.m4s"/>
24 ...
25 </SegmentList>
26 </Representation>

```

```
27 ...
28 <Representation id="17" mimeType="video/mp4" codecs="avc1.42c032" width="1920"
    height="1080" frameRate="30000/1001" sar="1:1" startWithSAP="1" bandwidth="
    5942702">
29 <SegmentList timescale="1000" duration="2000">
30 <Initialization sourceURL="6000kbps_2sec_segmentinit.mp4"/>
31 <SegmentURL media="6000kbps_2sec_segment1.m4s"/>
32 ...
33 </SegmentList>
34 </Representation>
35 </AdaptationSet>
36 <AdaptationSet segmentAlignment="true" bitstreamSwitching="true">
37 <AudioChannelConfiguration schemeIdUri="urn:mpeg:dash:23003:3:
    audio_channel_configuration:2011" value="2"/>
38 <SegmentList>
39 <Initialization sourceURL="audio_2sec_init.mp4"/>
40 </SegmentList>
41 <Representation id="1" mimeType="audio/mp4" codecs="mp4a.67.02" audioSamplingRate=
    "48000" startWithSAP="1" bandwidth="63553">
42 <SegmentList timescale="1000" duration="2000">
43 <SegmentURL media="64kbps_segment1.m4s"/>
44 ...
45 </SegmentList>
46 </Representation>
47 ...
48 <Representation id="4" mimeType="audio/mp4" codecs="mp4a.67.02" audioSamplingRate=
    "48000" startWithSAP="1" bandwidth="164553">
49 <SegmentList timescale="1000" duration="2000">
50 <SegmentURL media="165kbps_segment1.m4s"/>
51 ...
52 </SegmentList>
53 </Representation>
54 </AdaptationSet>
55 </Period>
56 </MPD>
```

Listing 2.1: Example MPD with SegmentList.

Listing 2.1 provides an example MPD that uses the profile *urn:mpeg:dash:profile:isoff-main:2011* which states that the multimedia content is stored using the IBMFF and that the segments are described using the *SegmentList*. The root element *MPD* has the attribute *@type* which is set to *static*. This indicates that the MPD describes on-demand multimedia content. The attribute *@minBufferTime* states the duration which has to be buffered at minimum until the playback of the multimedia content starts. The *BaseURL* allows to specify where the multimedia content is hosted. There may be more than a single *BaseURL*. If this is the case, then the client has to decide from which servers it requests the segments. A MPD may contain one or more *Period* elements. A *Period* is a set of multimedia content components that have a common timeline and are related to each other. The *AdaptationSet* element contains one or more representations that correspond to the same media stream (e.g., audio, video, subtitles). The *Representation* element describes a single representation of a media stream. This may be a multiplexed multimedia stream that contains audio, video and/or subtitles or only elementary streams. The *@id* attribute uniquely identifies a representation. In this example the representations provide scalability in the quality and spatial domain. The attributes *@width* and *@height* specify the resolution of the representation. The *@bandwidth* attribute depicts the average bit-rate in bit per second (bps) of the representation. The frame rate is depicted by the *@frameRate* attribute which does not change among all available representations. The *@codecs* attribute specifies the codecs used for the representation and is used for compatibility reasons. This allows clients to select representations that can be decoded with the decoders available. Within the *Representation* the segments are listed using the *SegmentList* element. In the case that the segments are not self initializing the list contains the *Initialization* element which specifies the dedicated initialization segment. Each *SegmentURL* depicts a segment with the duration given by  $\frac{\text{@duration}}{\text{@timesacle}}$  in seconds. As already mentioned the segments can be stored as contiguous file which would imply that the single segments are addressed using byte ranges (with the *@mediaRange* attribute), or the segments are stored in separate files and we use the *@sourceURL* attribute to reference a single segment. In this example the audio and video streams are separated and, therefore, for both streams different representations are provided. The second *@AdaptationSet* provides the representation for the audio stream. Here,

we have only quality scalability. This is indicated by having only the *@bandwidth* attribute that changes throughout the representations for audio.

```

1 <?xml version="1.0"?>
2 <MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT1.5S" type="static"
   mediaPresentationDuration="PT1H26M" profiles="urn:mpeg:dash:profile:isoff-live:2011">
3 <BaseURL>http://example.com/content/</BaseURL>
4 <Period duration="PT1H26M">
5 <AdaptationSet segmentAlignment="true" group="1">
6 <Representation id="1" mimeType="video/mp4" codecs="avc1.42c00d" width="320" height
   ="240" frameRate="30000/1001" sar="1:1" startWithSAP="1" bandwidth="101355">
7 <SegmentTemplate timescale="1000" duration="2000" media="100
   kbps_2sec_segment$Number$.m4s" startNumber="1" initialization="100
   kbps_2sec_segmentinit.mp4"/>
8 </Representation>
9 <Representation id="2" mimeType="video/mp4" codecs="avc1.42c00d" width="320" height
   ="240" frameRate="30000/1001" sar="1:1" startWithSAP="1" bandwidth="150705">
10 <SegmentTemplate timescale="1000" duration="2000" media="150
   kbps_2sec_segment$Number$.m4s" startNumber="1" initialization="150
   kbps_2sec_segmentinit.mp4"/>
11 </Representation>
12 <Representation id="3" mimeType="video/mp4" codecs="avc1.42c01e" width="480" height
   ="360" frameRate="30000/1001" sar="1:1" startWithSAP="1" bandwidth="201237">
13 <SegmentTemplate timescale="1000" duration="2000" media="200
   kbps_2sec_segment$Number$.m4s" startNumber="1" initialization="200
   kbps_2sec_segmentinit.mp4"/>
14 </Representation>
15 ...
16 <Representation id="12" mimeType="video/mp4" codecs="avc1.42c01f" width="1280"
   height="720" frameRate="30000/1001" sar="1:1" startWithSAP="1" bandwidth="
   1993730">
17 <SegmentTemplate timescale="1000" duration="2000" media="2000
   kbps_2sec_segment$Number$.m4s" startNumber="1" initialization="2000
   kbps_2sec_segmentinit.mp4"/>
18 </Representation>
19 <Representation id="13" mimeType="video/mp4" codecs="avc1.42c01f" width="1280"
   height="720" frameRate="30000/1001" sar="1:1" startWithSAP="1" bandwidth="
   2475387">

```

```
20 <SegmentTemplate timescale="1000" duration="2000" media="2500
    kbps_2sec_segment$Number$.m4s" startNumber="1" initialization="2500
    kbps_2sec_segmentinit.mp4" />
21 </Representation>
22 <Representation id="14" mimeType="video/mp4" codecs="avc1.42c032" width="1920"
    height="1080" frameRate="30000/1001" sar="1:1" startWithSAP="1" bandwidth="
    2995671" >
23 <SegmentTemplate timescale="1000" duration="2000" media="3000
    kbps_2sec_segment$Number$.m4s" startNumber="1" initialization="3000
    kbps_2sec_segmentinit.mp4" />
24 </Representation>
25 <Representation id="15" mimeType="video/mp4" codecs="avc1.42c032" width="1920"
    height="1080" frameRate="30000/1001" sar="1:1" startWithSAP="1" bandwidth="
    3992758" >
26 <SegmentTemplate timescale="1000" duration="2000" media="4000
    kbps_2sec_segment$Number$.m4s" startNumber="1" initialization="4000
    kbps_2sec_segmentinit.mp4" />
27 </Representation>
28 <Representation id="16" mimeType="video/mp4" codecs="avc1.42c032" width="1920"
    height="1080" frameRate="30000/1001" sar="1:1" startWithSAP="1" bandwidth="
    4981983" >
29 <SegmentTemplate timescale="1000" duration="2000" media="5000
    kbps_2sec_segment$Number$.m4s" startNumber="1" initialization="5000
    kbps_2sec_segmentinit.mp4" />
30 </Representation>
31 <Representation id="17" mimeType="video/mp4" codecs="avc1.42c032" width="1920"
    height="1080" frameRate="30000/1001" sar="1:1" startWithSAP="1" bandwidth="
    5942702" >
32 <SegmentTemplate timescale="1000" duration="2000" media="6000
    kbps_2sec_segment$Number$.m4s" startNumber="1" initialization="6000
    kbps_2sec_segmentinit.mp4" />
33 </Representation>
34 </AdaptationSet>
35 <AdaptationSet segmentAlignment="true" bitstreamSwitching="true" >
36 <AudioChannelConfiguration schemeIdUri="urn:mpeg:dash:23003:3:
    audio_channel_configuration:2011" value="2" />
37 <SegmentTemplate initialization="audio_2sec_init.mp4" duration="0" />
```



```

38 <Representation id="1" mimeType="audio/mp4" codecs="mp4a.67.02" audioSamplingRate=
    "48000" startWithSAP="1" bandwidth="63553">
39 <SegmentTemplate timescale="1000" duration="2000" media="
    audio_64kbps_segment$Number$.m4s" startNumber="1"/>
40 ...
41 <Representation id="4" mimeType="audio/mp4" codecs="mp4a.67.02" audioSamplingRate=
    "48000" startWithSAP="1" bandwidth="164553">
42 <SegmentTemplate timescale="1000" duration="2000" media="
    audio_165kbps_segment$Number$.m4s" startNumber="1"/>
43 </Representation>
44 </AdaptationSet>
45 </Period>
46 </MPD>

```

Listing 2.2: Example MPD with SegmentTemplate.

Listing 2.2 depicts an example MPD that uses the profile *urn:mpeg:dash:profile:isoff-live:2011*. The difference to the previous MPD example is that the “live” profile allows to provide a more lightweight description of the scalable multimedia content. The *SegmentTemplate* provides the possibility to use so-called templates such as *\$Number\$* which indicates that the segment number shall be inserted here. The attribute *@startNumber* signals the client which one of the segment is the first segment to start with. Again, the *@type* attribute of the root element is set to static. MPEG-DASH supports live streams too. In order to indicate that the described multimedia content is live content, the *@type* attribute has to be set to dynamic. If the MPD is dynamic it is required to provide a starting time of the live content. This allows the clients to calculate the latest generated segment which is as nearest to the live event as possible.

## 2.5 Subjective Quality Assessment of Audio-Video Content using Crowdsourcing

The recommendations of the International Telecommunication Union (ITU), provided in ITU-T. P.910 [56] and ITU-R. BT.500-13 [57], provide very well documented recommendations on how to conduct in-lab subjective quality assessments. More

specifically, the ITU-R BT.500-13 covers the methodologies for assessing the quality of television pictures. For subjectively assessing the QoE of multimedia applications ITU-T. P.910 provides a set of proven methodologies. This even covers multi-modal stimulus presentations such as audio and video. The recommendation covers the following subjective quality assessment methodologies:

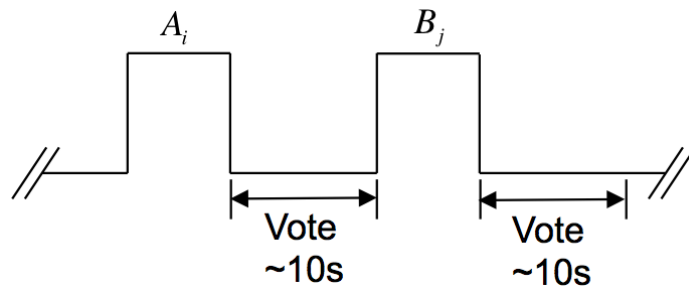


Figure 2.7: Absolute Category Rating, where  $A_i$  denotes sequence  $A$  under test condition  $i$  and  $B_j$  denotes sequence  $B$  under test condition  $j$  [56].

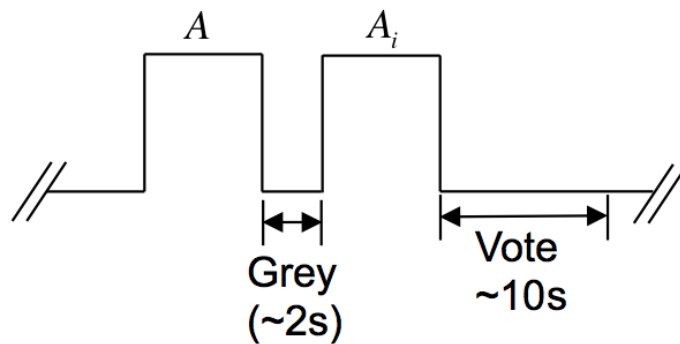


Figure 2.8: Degradation Category Rating, where  $A$  denotes the reference and  $A_i$  denotes the  $i$ -th test condition [56].

- **Absolute Category Rating (ACR):** The test sequences are presented one at a time and are rated independently on a categorical scale (cf. Figure 2.7). The ACR is a single stimulus methodology.
- **Absolute Category Rating with Hidden Reference (ACR-HR):** In general it is the same as the ACR method except that the reference condition is presented too but the participants do not know which is the reference condition. This allows to compute the differential Mean Opinion Score (DMOS) and

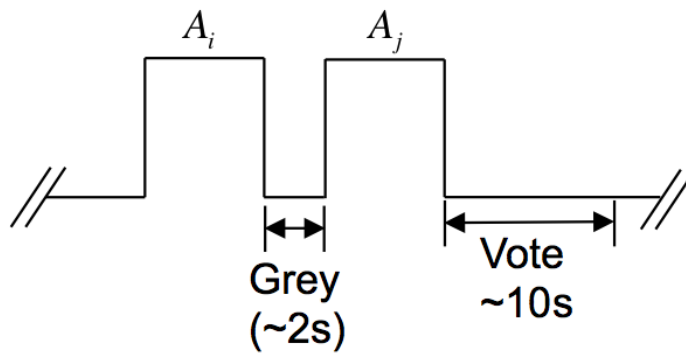


Figure 2.9: Pair Comparison, where  $A_i$  denotes sequence  $A$  under test condition  $i$  and  $A_j$  denotes sequence  $A$  under test condition  $j$  [56].

to identify how much the test conditions deviate from the reference condition in terms of QoE.

- **Degradation Category Rating (DCR):** Here, the test sequence is presented first and the reference condition is presented right after test sequence (cf. Figure 2.8) with a short pause of about two seconds. The DCR is a double stimulus methodology.
- **Pair Comparison (PC):** The test sequences are presented in pairs. The same sequence is presented under the first test condition and right after under the second test condition (cf. Figure 2.9). Often the participants are allowed to switch between the stimuli. The participants then rate which one of the presented stimuli is preferred. The Bradley-Terry-Luce model allows to predict the outcome of the pair comparison [58].

The voting phase (denoted by *Vote*) depicted in Figures 2.7, 2.8, and 2.9 provides the rating possibility for assessing the QoE or in the case of paired comparison the selection which stimuli is preferred. Furthermore, the category scale can be exchanged by numerical scales. In our subjective quality assessments using crowdsourcing (cf. Chapter 4) we use a discrete scale from 0 to 100 for rating the QoE. This rating scale is *continuous* in the sense that the discrete steps do not denote a certain category. This rating scale provides the illusion of a *continuous* rating (henceforth denoted as *continuous* rating scale) from very low QoE (0) to very high QoE (100).

Subjective quality assessments can be carried out either internally in the laboratories of the research facility (in-lab), where the researchers can control the environment or they can be carried out externally at the user where the researchers have no control about the environment. Conducting subjective quality assessments is often expensive and time consuming because enough participants have to be acquired and some sort of incentive has to be provided (e.g., money, gift cards) and at least one researcher has to be present during the subjective quality assessment to supervise the experiment. Conducting subjective quality assessments *at the user's home* has gained momentum in the last four years. Especially, with the upraise of so-called crowdsourcing platforms like Mechanical Turk [59] and Microworkers [60], where workers do micro tasks that consume typically a few minutes for very little money (e.g., 0.40\$).

Nevertheless, using the crowd for conducting subjective quality assessments poses completely new challenges to the test design. In contrast to in-lab subjective quality assessments where only a few persons participate in a subjective study, crowdsourcing provides the possibility to reach a vast amount of users, called the **crowd**. Apart from in-lab subjective quality assessments where the participants are invited and known, participants hired through crowd platforms are mostly **anonymous**. Furthermore, the subjective quality assessments using crowdsourcing are mostly conducted using a web-based platform. Therefore, there is no possibility to directly supervise the participants on the given tasks as it would be possible in-lab. This uncontrolled environment implies many further problems, beginning at the test design and the test validation. Completely different environments may lead to controversial results. Thus, the test design has to provide the possibility for identifying these effects and to investigate them separately. For example, different hardware such as different screens (e.g., size, color fidelity, audio devices) may lead to a different perception of the multimedia content and in particular of stimuli under the test conditions.

Another big issue is the motivation of the crowd. Especially, if participants are hired using platforms like Mechanical Turk [59] and/or Microworkers [60]. The payment does not only provide an extrinsic factor for the motivation. A too high payment may have an opposite effect as intended. For example, users may try to cheat in order to complete as many runs as possible to maximize their payment. Especially, in

low-wage countries the payment has to be carefully chosen. The web-based platform and the test design shall provide the possibility to identify such cases in order to filter them from the final results. A very interesting compilation of best practices for crowdsourcing subjective quality assessments is presented in [61]. In particular, different outlier screening methods are compared (including the one presented in [62]). The results clearly state that, detecting outliers can not only rely on taking user ratings into account. Thus, additional mechanism should be introduced in order to detect outliers. [63] and [64] propose further methods for screening outliers and for evaluating the results of subjective quality assessments using crowdsourcing. Therefore, an equivalent to the Mean Opinion Score (MOS) called CrowdMOS has been introduced. CrowdMOS is designed to overcome the issues of MOS for crowdsourcing and comprises a subset of the methodologies defined in [56] and [57] but tailored especially for the use with Mechanical Turk. The subset of methodologies are the single stimulus ACR in three variations, Multiple-Stimulus with Hidden Reference and Anchor (MUSHRA), and the the Double Stimulus Impairment Scale (DSIS) [63, 64]. The proposed CrowdMOS and its methodologies are limited in comparison to [56] and [57] because only a subset of the recommended test methods were adapted for crowdsourcing.

In [65] a crowdsourcing framework which is called QualityCrowd is presented. This framework directly communicates with Mechanical Turk and allows the definition of various test methodologies. Additionally, the authors of [65] discuss various challenges that arise when subjective quality assessments are *crowdsourced*, i.e., conceptual, technical, motivational, and reliability challenges. Another problem that arises is the cheat detection. In [62] a novel cheat detection mechanism for subjective quality assessments using the pair comparison is proposed. The cheat detection mechanism is evaluated using the data gathered by a subjective quality assessment. The proposed methods are not applicable to all evaluation methodologies. Thus, for our subjective quality assessment we had to come up with a cheat detection that fits the evaluation methodology used.

Another platform and approach for conducting subjective quality assessments using crowdsourcing is presented in [66], where an own platform was designed for

conducting user studies to assess the Quality of Experience. This platform uses Mechanical Turk to *hire* participants. The platform provides separate administration and experiment interfaces. Furthermore, the platform only supports pair comparison as test methodology. In [66] two network related user studies are presented. The user studies were conducted by means of the proposed platform and evaluate the effect of packet loss on voice over IP (VoIP) and the influence of packet loss on IPTV. During the experiment the participants are allowed to press the space-bar to switch the quality of the presented content (e.g., audio or video). The voting is done by asking the participants which of one of the space-bar states maps to a better perceived quality.

In [37] a YouTube QoE model and a subjective QoE assessment methodology based on crowdsourcing are presented. The authors of [37] describe how the task design for subjective quality assessments using crowdsourcing should look like. They introduce the following types of control questions:

- **Content based Questions**, the answers are already known in advance by the research. Thus, they can be easily cross checked. If participants do not provide the correct answers the whole results of these subjects are regarded as incorrect. This type of question is also used to check the participants attention.
- **Repeated Questions with differences**, the same question is asked multiple times with small differences. These difference may be different graphical elements for providing the answer. If the answers are not consistent to all these questions the results are rejected.

The authors of [37] propose that these questions and methods should provide trustworthy results by filtering the collected data according to the answers of the mentioned questions. However, this method of filtering the data has advantages and disadvantages. The advantages are that the participants are forced to pay attention to the questions and that participants that just randomly select the answers are filtered. But, the filtering scheme may be too strict. For example, asking the same question multiple times in a different manner can lead to the rejection of participants who did an acceptable job in the actual assessment but only provided a wrong answer to the control questions. Furthermore, asking questions about the content whether

there was something present which actually was not present during the presentation of the content, may confuse some participants. Changing the rating scale during the experiment may have a crucial impact on a subjective QoE assessment because participants may get used to the rating scale during the course of the experiment; changing the presentation of the rating possibility after each presentation of the actual test content may confuse the participants.

The discussed literature on crowdsourcing shows various methodologies for conducting subjective quality assessments. It can be concluded that there is no perfect test design that overcomes all problems that arise when conducting subjective quality assessments using crowdsourcing. The user test design strongly depends on the purpose of the assessment and how outliers should be detected, identified, and maybe rejected.





---

# 3 Distributed negotiation on a Reference Playback Timestamp

## 3.1 Introduction

The presence of a real-time communication within a group of users that want to watch the same multimedia content requires a synchronized playback among the participating users. Asynchronism may lead to an unpleasant viewing experience and may diminish the feeling of togetherness of the users as reported in [1]. Consider for example, if, out of a group of friends or colleges watching a soccer game together using multiple devices, some experience playback that is a few seconds ahead of the others. This kind of asynchronism between individual users may lead to a low system acceptance. The technical challenges of IDMS can be summarized based on the type of streaming technology employed (e.g., pull- or push-based), the selection of an appropriate synchronization point, and in cases where asynchronism does occur, an appropriate mechanism should be used to smoothly and imperceptibly synchronize the multimedia playback at the peers. The research presented in this chapter is based on [11].

Our approach differs from existing push-based IDMS approaches that utilize RTP/RTCP receiver reports to signal timing and control information which were discussed in Section 2.2 and instead extends IDMS to pull-based over-the-top streaming by adopting MPEG-DASH [50]. This chapter deals with the following mechanisms of IDMS:

- **Session management** which is responsible for managing the session to which peers belong;

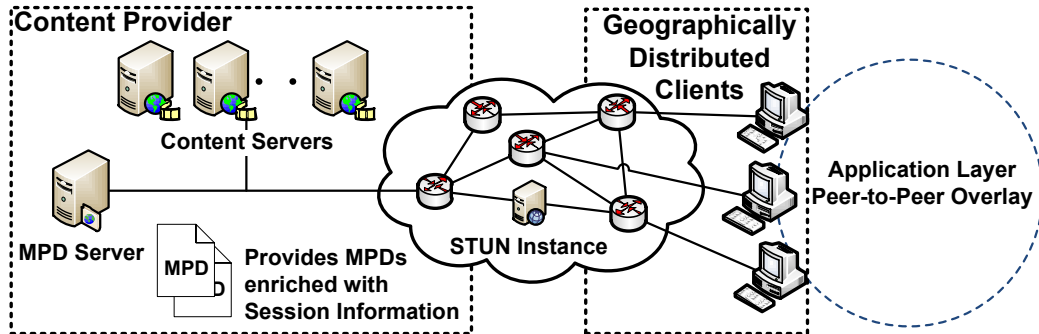


Figure 3.1: Architecture of IDMS for DASH.

- **Peer-to-Peer overlay creation** which allows to exchange data *directly* between peers;
- **Signaling of timing and control information** allows the exchange of timing information and, if necessary, control information between peers;
- **Negotiation on a reference playback timestamp** deals with the selection of a playback timestamp within a session to which the peers have to synchronize their playback.

In a RTP/RTCP-based environment these mechanisms are typically implemented at the server level whereas MPEG-DASH adopts a client-centric approach and, hence, migrates these mechanisms to the client. This facilitates simple HTTP servers that provide content in a MPEG-DASH compliant format (e.g., segmented ISO Base Media File Format or MPEG-2 Transport Stream).

Figure 3.1 illustrates our IDMS approach for pull-based streaming and, in particular, for MPEG-DASH. Rather than modifying the server side for IDMS, we introduce **session management** by defining IDMS Session Objects (ISOs). These ISOs are referenced from within the MPD and are stored at the server providing the MPD (i.e., *MPD Server*). We assume that there is a dedicated *MPD Server* that handles MPD requests from peers. The *Application Layer Peer-to-Peer (P2P) Overlay* is implicitly built by our **distributed synchronization protocol** that utilizes the

information contained in an ISO. The creation of the *Application P2P Overlay* may utilize the Session Traversal Utilities for NAT (STUN) [67] and its relay extensions specified in [68] in order to allow a P2P connection even if peers are behind Network Address Translators (NAT). Therefore, our IDMS architecture foresees a separate *STUN Instance* which provides the peers the possibility of detecting the type of their NAT. The **distributed synchronization protocol** consists of a two-stage protocol. First, it provides a **coarse synchronization** that is introduced on behalf of how multimedia content is segmented using MPEG-DASH. Second, it provides the **fine grained synchronization** that finally provides a reference playback timestamp for synchronizing the multimedia playback of the peers.

## 3.2 Session Management

We adopt MPEG-DASH [50] as an enabler for our IDMS approach to pull-based streaming, extending the MPD with so-called IDMS Session Objects (ISOs) that are matched against a session key provided by users. Nevertheless, our solution remains compliant to the MPEG-DASH standard because non-IDMS peers will just ignore the additional session description when parsing the MPD. Pull-based streaming such as MPEG-DASH represents multimedia content as equally sized, self-contained time units (e.g, 2s, 4s, 10s, etc.) which are referred to as segments (cf. Section 2.4). These segments may be stored as separate files or are indexed by byte ranges in a contiguous file. Additionally, the multimedia content may be provided in different representations – described with the MPD – offering various scalability (e.g., spatial, temporal, quality) of the multimedia content. The adaptation between representations takes place at segment boundaries. For more details on MPEG-DASH see Section 2.4.

**Definition 1 (IDMS Session Object)** An ISO is a time bounded entity to which a set of peers is assigned to. Each ISO shall have an unique identifier for a certain multimedia content. Furthermore, it shall allow a unique numbering of the peers and peers shall be unique addressable.

Each ISO shall have a Time-To-Live which indicates for how long a specific ISO is valid. Invalid ISOs may be deleted without any caution. According to Definition

1, we assume that an ISO is identified by an unique session key which is provided by the user or the application. The session key is signaled by adding it to the HTTP GET message that requests a MPD from the *MPD Server* along with the public IP (IPv4 or IPv6), port number and type of the NAT. For instance, the *MPD Server* employs a PHP-Script for responding with the appropriate MPD an example HTTP GET request may look as follows:

```

1 GET /MPDService.php?MPD=ExampleMPD&SessionKey=FDEA012345&IP=143.205.122.242&
   Port=8029&NAT=NoNAT HTTP/1.1
2 Host: www.example.com

```

Listing 3.1: Example HTTP GET Request.

The response to the example HTTP GET request depicted by Listing 3.1 could be the MPD with the requested session information that corresponds to the provided session key. Or, the PHP-Script generates a temporary MPD that includes the corresponding session information and responds with a redirection to the temporary MPD.

As peers may be behind a NAT, STUN is employed to determine the public IP address and port number to be used during the synchronization. Every peer has to follow a certain procedure before it requests a MPD containing session information. We use the same ports for STUN negotiation with the separate *STUN instance* (cf. Figure 3.1) as we do for our synchronization protocols. For further information on the procedure and how different types of NATs are traversed we refer the interested reader to Section 3.3.1.

The public IP (IPv4 or IPv6) address, port number, and NAT type is added along with the session key as URL parameters to the initial HTTP GET message that requests the MPD from the *MPD Server*. The initiation of an IDMS session and the provisioning of the session key is out of the scope of this work. We assume that the user or the application provides the session key. With the initiation of an IDMS session an ISO with a specific session key is created. The *MPD Server* adds the peers that request a certain MPD with a specific session key to the corresponding ISO. When a peer requests a MPD, the *MPD Server* adds the peer to the ISO associated with the session key and returns both. As peers may join the session at different

points in time, each peer may only have partial information about the actual number of peers in an IDMS session.

```
1 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.aau.at/  
  DASH/Session" targetNamespace="http://www.aau.at/DASH/Session" xmlns:xlink="http  
  ://www.w3.org/1999/xlink" >  
2 <xs:import namespace="http://www.w3.org/1999/xlink" schemaLocation="xlink.xsd"/>  
3 <xs:element name="IDMSSessionObject">  
4   <xs:complexType>  
5     <xs:sequence>  
6       <xs:element name="PeerList" type="PeerListType" minOccurs="0" maxOccurs  
       = "unbounded"/>  
7       <xs:element name="TTL" type="xs:dateTime" minOccurs="1" maxOccurs="1"  
       />  
8     </xs:sequence>  
9     <xs:attribute ref="xlink:href"/>  
10    <xs:attribute ref="xlink:actuate" default="onLoad"/>  
11  </xs:complexType>  
12 </xs:element>  
13 <xs:complexType name="PeerListType">  
14   <xs:sequence>  
15     <xs:element name="Peer" type="PeerType" minOccurs="0" maxOccurs=""  
     unbounded"/>  
16   </xs:sequence>  
17 </xs:complexType>  
18 <xs:complexType name="PeerType">  
19   <xs:sequence>  
20     <xs:element name="Identifier" type="PeerIdentifierType" minOccurs="1"  
     maxOccurs="unbounded"/>  
21   </xs:sequence>  
22 </xs:complexType>  
23 <xs:complexType name="PeerIdentifierType">  
24   <xs:sequence>  
25     <xs:element name="IP" type="xs:string"/>  
26     <xs:element name="Port" type="xs:integer"/>  
27   </xs:sequence>  
28   <xs:attribute name="nat" type="xs:string"/>  
29 </xs:complexType>  
30 </xs:schema>  
31
```

Listing 3.2: IDMS Session Object for MPEG-DASH.

Listing 3.2 depicts the XML Schema for an ISO. As the definition of the ISO demands that an ISO shall allow a unique number of the peers in an IDMS session the XML schema foresees a list of peers (represented by *@PeerListType*). The definition of the ISO further states that an ISO shall be a time bounded entity. Therefore, we introduce a Time-To-Live (TTL) element (represented by *@TTL*). The maximum TTL for an IDMS session is the duration of the requested multimedia content which may be in case of a live stream an estimated duration (e.g., the estimated end time of a soccer match). The *@PeerIdentifierType* contains the public IP address, port number, and the NAT type of a specific peer. Note that a peer may have more than one identifier if it has several network interfaces that are connected to different networks.

```
1 <MPD xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:iso="http://www.aau.at/DASH/Session
  " type="static" mediaPresentationDuration="PT3256S" minBufferTime="PT1.2S" profiles=
  "urn:mpeg:dash:profile:isoff-on-demand:2011" >
2 <BaseURL>http://www.example.com/</BaseURL>
3 <Period>
4   <AdaptationSet>
5     <Representation id="0" mimeType="video/mp4" codecs="avc1.42c01f,mp4a.40.02"
      startWithSAP="1" bandwidth="251674" >
6       <SegmentList timescale="1000" duration="10000" >
7         <Initialization sourceURL="init0.mp4" />
8         <SegmentURL media="seg0-1.m4s" />
9         <!-- ... further segments -->
10        </SegmentList>
11      </Representation>
12      <Representation id="1" mimeType="video/mp4" codecs="avc1.42c01f,mp4a.40.02"
        startWithSAP="1" bandwidth="380974" >
13        <SegmentList timescale="1000" duration="10000" >
14          <Initialization sourceURL="init1.mp4" />
15          <SegmentURL media="seg1-1.m4s" />
16          <!-- ... further segments -->
17        </SegmentList>
18      </Representation>
```

```
19 <Representation id="2" mimeType="video/mp4" codecs="avc1.42c01f,mp4a.40.02"  
    startWithSAP="1" bandwidth="666666">  
20 <SegmentList timescale="1000" duration="10000">  
21 <Initialization sourceURL="init2.mp4"/>  
22 <SegmentURL media="seg2-1.m4s"/>  
23 <!-- ... further segments -->  
24 </SegmentList>  
25 </Representation>  
26 <!-- ... more representations -->  
27 </AdaptationSet>  
28 </Period>  
29 <iso:IDMSSessionObject>  
30 <iso:PeerList>  
31 <iso:Peer>  
32 <iso:Identifier nat="NoNAT">  
33 <iso:IP>143.205.122.242</iso:IP>  
34 <iso:Port>8029</iso:Port>  
35 </iso:Identifier>  
36 <iso:Identifier nat="FullCone">  
37 <iso:IP>143.205.199.149</iso:IP>  
38 <iso:Port>8030</iso:Port>  
39 </iso:Identifier>  
40 </iso:Peer>  
41 <iso:Peer>  
42 <iso:Identifier nat="PortRestricted">  
43 <iso:IP>10.0.0.5</iso:IP>  
44 <iso:Port>8029</iso:Port>  
45 </iso:Identifier>  
46 </iso:Peer>  
47 <!-- ... more peers -->  
48 </iso:PeerList>  
49 <iso:TTL>2014-07-26T21:32:52</iso:TTL>  
50 </iso:IDMSSessionObject>  
51 </MPD>
```

Listing 3.3: Excerpt of an example MPD including an ISO.

Listing 3.3 shows an excerpt of a MPD comprising an ISO. Peers requesting the MPD will be added to the corresponding ISO. The process of adding peers to the

ISO induces an implicit ordering of the peers which is used by the subsequent P2P synchronization algorithms. Every peer numbers the peers in the ISO strict monotonically increasing beginning with one. Even if peers leave the p2p overlay their entry in the ISO is not deleted, otherwise removing peers that leave would violate the total order on the peers.

Please note that it is not mandatory to use MPEG-DASH with our approach. Our approach only requires that the peers have to connect only once to a central instance where the ISO is hosted for the requested multimedia content and the corresponding IDMS session. In the case of MPEG-DASH the peers have to fetch the MPD before starting to stream the multimedia content. This circumstance fits very well with our approach and allows to add the IDMS session information using the ISO using the MPD of MPEG-DASH.

Figure 3.2 depicts an example of a peer that requests a MPD with a specific session key. Before any communication to the *MPD Server* happens each peer determines its public IP, port, and NAT type (if existent) by using the STUN instance (cf. Section 3.3.1). If the *MPD Server* receives a request for a MPD with a session key it first checks whether the session key is valid. In the case that the provided session key is invalid the *MPD Server* returns an error message. If the session key is valid the *MPD Server* creates or/and loads the corresponding ISO. If the TTL of the ISO has expired, the *MPD Server* responds with an error and deletes the ISO. Otherwise, the *MPD Server* adds the requesting peer with its public IP, port and NAT type, if it is not yet contained in the ISO. After this handshake the peer is able to join or create (if it is the first peer in the IDMS session) the unstructured P2P overlay network.

### 3.3 Signalling of Timing Information and Negotiation of the Playback Timestamp

For creating the P2P overlay in order to signal timing information and to agree on a reference playback timestamp among the peers, we propose to determine the playback timestamp to which the peers shall synchronize their playback in two steps. In the first step, referred to as *coarse synchronization*, new peers request the segment



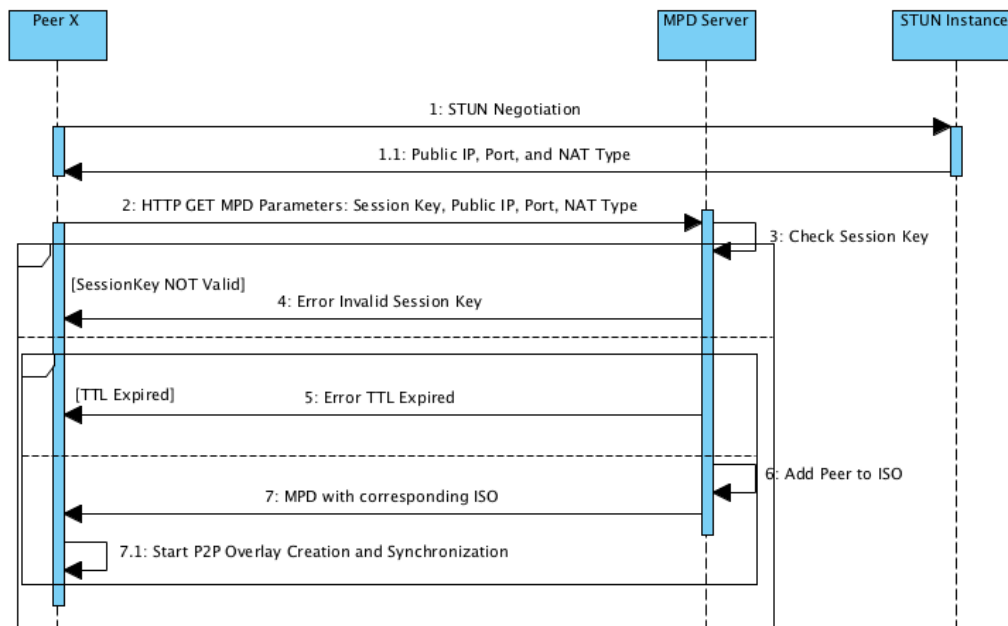


Figure 3.2: Sequence Diagram of a peer requesting a MPD with a session key.

numbers which are currently played by other peers within the IDMS session, thus reducing the synchronization effort and building the P2P overlay. In a second step, *fine synchronization*, peers agree on a reference playback timestamp in a distributed manner by using the constructed P2P overlay. In general, we assume that the clocks of the peers are synchronized (e.g. using NTP [24] or PTP [25]).

### 3.3.1 Unstructured Peer-To-Peer Overlay Construction and Coarse Synchronization

The P2P overlay is created using the information contained in the ISO. As mentioned before, peers may be behind NATs which have to be traversed in order to create a P2P overlay network and to communicate in a P2P manner. Therefore, each peer communicates with a STUN instance (could be the *MPD Server*) in order to determine whether the peer is behind a NAT and, if available, the type of NAT. We differentiate between the following types of NAT: no NAT, symmetric firewall, full cone NAT, restricted cone NAT, port restricted NAT, and symmetric NAT. In the case of a full cone NAT, where the mapping is done statically by the NAT such that

any address is allowed to send packets by using the public IP address and port number to a peer, the communication with the STUN instance has already registered the necessary ports at the peer's NAT. If the peer detects that its NAT is a restricted cone NAT or port restricted NAT, the peer's NAT allows only incoming packets from an address if the peer has already send a packet to this address. Therefore, we add a relaying function to the STUN instance which is used by peers with a restricted cone NAT or port restricted NAT. The procedure for a peer with a restricted cone NAT or port restricted NAT is as follows:

- First, it sends a UDP packet to the address with which it wants to communicate using the IP and port signaled by the ISO, this tells the NAT that incoming packets from the destination address are allowed;
- Second, if the other peer has one of the mentioned NATs it uses the relaying function of the STUN instance to signal the other peer that it shall send a packet to the given public IP and port;
- Third, the other peer uses the signaled information to open its NAT. After this *handshake* the P2P synchronization protocol can be carried out.

This allows to have more than one peer behind the same NAT. If there is no NAT but a symmetric firewall or a symmetric NAT there is no peer-to-peer communication possible. If the router which *hides* the peers supports Universal Plug and Play it is possible to add forwarding rules for the ports used by our synchronization protocols but this is out of scope of this work [69].

UDP is used as the transport protocol between the peers because reliable communication is not essential. Using TCP would imply that the peers have to maintain connections to other peers which again implies some overhead. Our algorithms do not need a reliable communication, the only requirement that the algorithms that will be introduced is a connected network such that there is no partitioning.

Figure 3.3 depicts the message structure for the *coarse synchronization* which is used for both response and request as specified by the *Type* field. For requests, the fields *IP* and *Port* are set to the public IP address and port number of the requesting

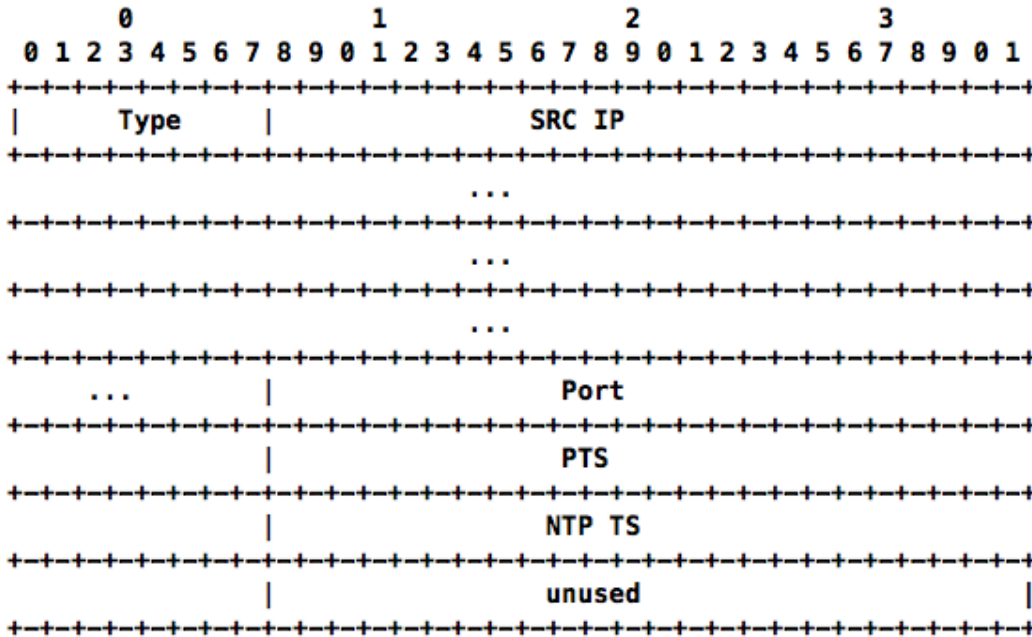


Figure 3.3: Message structure for the *coarse synchronization*.

peer – other fields are empty – which indicates that the receiving peer shall respond with its current playback timestamp. The responding peer sets its own IP address (field *IP*) and port number (field *Port*) allowing peers to track which peers responded to which requests. The field *PTS* is set to the current playback timestamp and the field *NTP TS* is the corresponding NTP timestamp. The NTP timestamp is used to align all received timestamps to the same point in time. As the list of peers in the ISO grows over time, peers joining the session early will only have a subset of available peers. Therefore, if a peer is asked for its playback timestamp by an unknown peer, it adds the associated IP address and port to the list of known peers.

Algorithm 1 implements *coarse synchronization*. Each peer that receives the ISO, requests the current playback timestamp from all peers it lists. The peer requesting (i.e., *request\_timestamps*) the playback timestamps waits until either all requests have been satisfied or until a given time period  $T_C$  has elapsed. If no timestamp arrives during  $T_C$  the peer starts over by requesting playback timestamps from the known peers. Each peer that receives a request (i.e., *receive\_request*) responds with its IP address, port number, playback timestamp, and the corresponding NTP timestamp. The timestamps from the responses may be combined to calculate the start segment

---

**Algorithm 1** Coarse Synchronization.

---

```

1: function request_timestamps
2:   for all  $p \in \text{peers}$  do
3:     sendPacket(Type.Request,  $p.IP$ ,  $p.Port$ ,  $myIP$ ,
4:                $myPort$ ,  $null$ ,  $null$ )
5:   end for
6:    $wait(T_C) \vee \text{receivedTS.size}() = \text{peers.size}()$ 
7:   if  $\text{receivedTS.size}() = 0$  then
8:     request_timestamps()
9:   else
10:    calculateSegment()
11:  end if
12: end function
13: function receive_request( $pt : \text{packet}$ )
14:   if  $isPeerKnown(pt.srcIP, pt.srcPort) \neq true$  then
15:     addPeer( $pt.srcIP$ ,  $pt.srcPort$ )
16:   end if
17:   sendPacket(Type.Response,  $pt.srcIP$ ,  $pt.srcPort$ ,
18:              $myIP$ ,  $myPort$ ,  $PTS$ ,  $NTPTS$ )
19: end function
20: function receive_timestamp( $pt : \text{packet}$ )
21:    $\text{receivedTS.add}(pt.PTS, pt.NTP)$ 
22: end function

```

---

using one of four strategies, namely the *i*) maximum, *ii*) minimum, *iii*) average, and *iv*) weighted average of the received timestamps. The selection of the strategy depends on the application and may be influenced by Quality of Service (QoS) parameters like bandwidth, delay, and maximum selectable bit-rate of the multimedia content. In our application, we determine the start segment using the average of the received playback timestamps. This procedure provides a first educated guess on the reference timestamp calculated later on and allows the newly joined peer(s) to start streaming multimedia content without waiting until a reference timestamp has been calculated in a distributed manner.

Let  $T_{ref}$  be the timestamp resulting from such a strategy. We calculate the segment to start with by  $\lceil \frac{T_{ref}}{T_s} \rceil$ , where  $T_s$  is the segment size in seconds (e.g., 1s, 2s, 3s, 4s, ...). Suppose that  $M$  is the theoretical reference timestamp to which all peers will adjust their playback. Therefore, without loss of generality the asynchronism  $\xi$  after asking the other peers for their playback timestamp and downloading  $N$  segments until the

playback starts is given by:

$$0 \leq |\xi| \leq \left| M - \left\lceil \frac{T_{ref}}{T_s} \right\rceil \cdot T_s + \sum_{i=1}^N \frac{b_c(t_i)}{b_r(t_i)} \right|, \quad (3.1)$$

where  $b_c(t)$  is the bit-rate of the transmission channel in bit/s at time instant  $t$  and  $b_r(t)$  is the bit-rate of the current representation of the multimedia content. The *coarse synchronization* ensures that if a peer joins an IDMS session it starts with a segment that is as closest as possible to the segment the other peers are currently playing. Especially, if the other peers have not yet agreed on a reference playback timestamp.

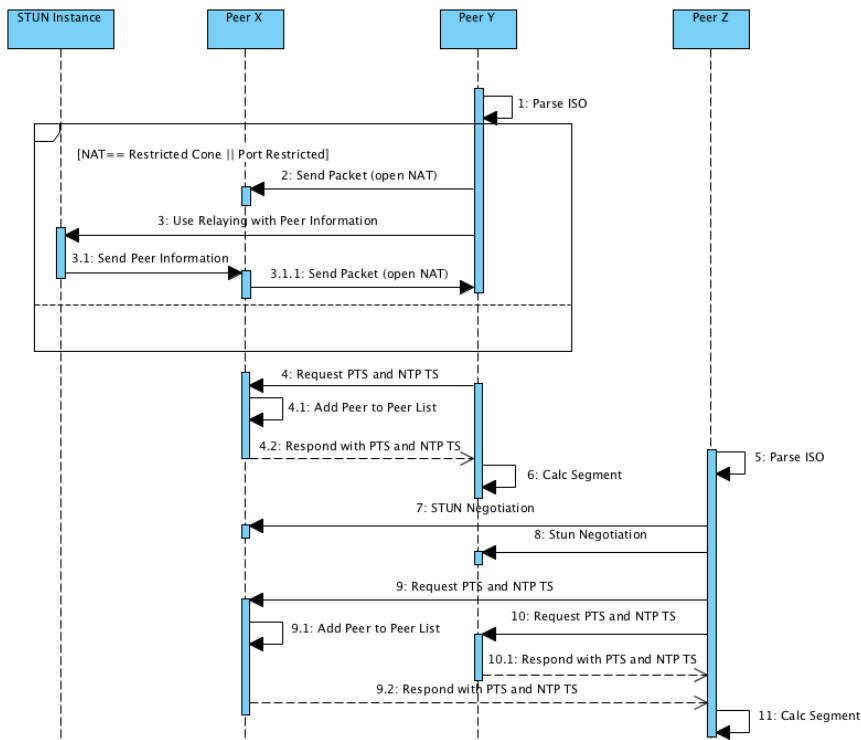


Figure 3.4: Sequence Diagram of three peers that join the unstructured P2P overlay network and carry out the coarse synchronization.

Figure 3.4 depicts a sequence diagram showing three peers that join an IDMS session. *Peer X* is the first peer that joins the IDMS session. It receives the corresponding ISO from the *MPD Server* (cf. Figure 3.2). The received ISO is parsed and the list of peers is locally stored using an appropriate data structure (i.e., a (hash) map using a structure that holds the public IP, port, and NAT type of each peer).

*Peer X* is the first peer in the unstructured P2P overlay (which is indicated by only a single entry in the ISO which is *Peer X* itself). Therefore, *Peer X* directly starts streaming the multimedia content described by the requested MPD. *Peer Y* joins the IDMS session after *Peer X* and, therefore, the ISO contains two entries, *Peer X* and *Peer Y*. Thus, *Peer Y* will try to connect to *Peer X*. Therefore, it has to check the NAT type and follows the procedure as described earlier in order to traverse its NAT and the NAT of the peer it wants to connect with. After the final STUN negotiation, *Peer Y* starts to request the PTS and NTP TS from all peers listed in the ISO (in this case from *Peer X*) and follows Algorithm 1. *Peer X* stores the new peer in its peer list for further communication and sends its current PTS and NTP TS to *Peer Y*. *Peer Y* calculates the segment to start with and begins streaming the multimedia content according to the calculated start segment. *Peer Z* follows the same procedure as *Peer Y* but will try to connect to both *Peer X* and *Peer Y*.

### 3.3.2 Aggregate

After successfully joining an IDMS session and, therefore, joining the unstructured P2P overlay and carrying out the coarse synchronization a peer starts with the fine synchronization. Literature provides a base-line algorithm using unicast [35] which we call *Aggregate*. Algorithm 2 depicts *Aggregate* and how this base-line algorithm allows the peers to agree on a reference playback timestamp. *Aggregate* follows a pure flooding approach. Every peer broadcasts periodically a list of playback timestamps that it has seen to its neighbors. Every time a peer receives a packet containing playback timestamps it merges the received ones with its own list. *Aggregate* utilizes the message structure depicted in Figure 3.5. The semantics of single fields is as follows:

- ***PeerID***: the unique identifier of a peer (determined by the the ISO).
- ***PTS***: the playback timestamp of the *j*-th peer.
- ***NTP TS***: the NTP timestamp for the corresponding PTS.

---

**Algorithm 2** Aggregate.
 

---

```

     $C_i, T_i, P_i$ 
1: function broadcastToNeighbors
2:   update( $T_i, NTP_i$ )
3:   for all  $p \in \text{peers}$  do
4:     sendPacket( $C_i, T_i$ )
5:   end for
6: end function
7: function receiveList( $C_i, T_i$ )
8:   update( $T_i, NTP_i$ )
9:   update( $T_j, NTP_j$ )
10:  for all  $t \in T_i \wedge t \in T_j$  do
11:    if  $T_j[t].seq_{nr} > T_i[t].seq_{nr}$  then
12:       $T_i[t] \leftarrow T_j[t]$ 
13:    end if
14:  end for
15:  for all  $t \in T_j \wedge t \notin T_i$  do
16:     $T_i \leftarrow T_i \cup t$ 
17:  end for
18:   $P_i \leftarrow \text{average}(T_i)$ 
19: end function

```

---

- **Sequence Number**: used to indicate that the PTS of a particular peer has changed (due to stalls or if the user used any trick modes).

The message structure indicates already a high overhead in terms of bytes that have to be sent. Each message contains  $j \geq 0$  items that are of the following form: *PeerID*, *PTS*, *NTP TS*, and the *Sequence Number*. Each peer maintains the following variables and lists:  $C_i$  denotes the number of peers in the list  $T_i$ ,  $T_i$  denotes the list of peers including their PTS and NTP TS, and  $P_i$  denotes the peer's current PTS. Each entry in the list of  $T_i$  contains the IP, Port, PTS, NTP TS, and the sequence number of a specific peer. If peer  $i$  receives a list from one of its neighbors  $j$ , it first aligns all playback timestamps using the corresponding NTP timestamps (cf. Algorithm 2 Line 8 and Line 9). Second, it checks if any updated timestamps are in the received list for which the sequence number is increased (cf. Algorithm 2 Line 10 to 14). Peers may use the sequence number to indicate that their playback timestamp has changed such that other peers can recalculate the average playback timestamp. Third, peer  $i$  merges its own list with the received one and calculates the average playback

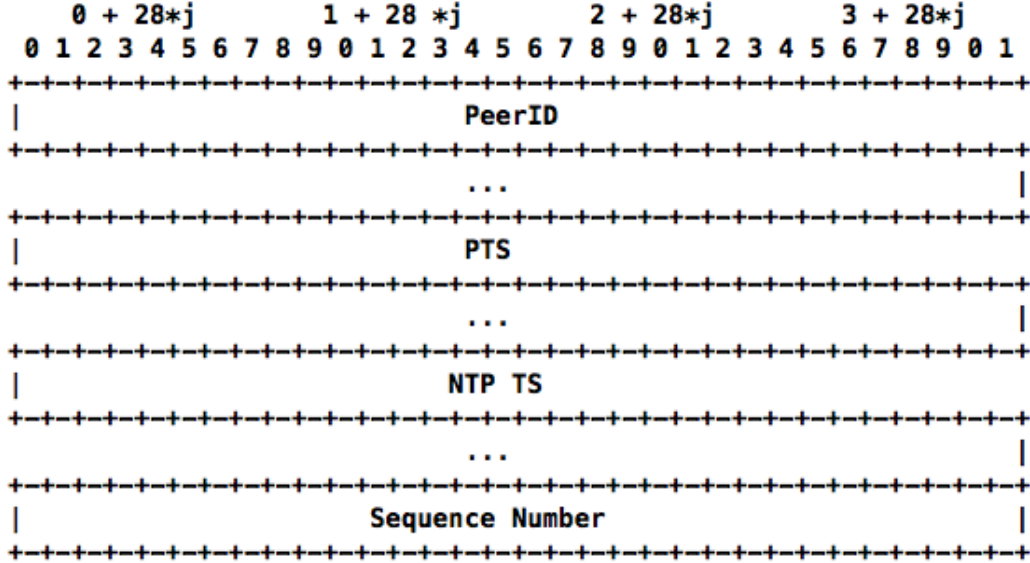


Figure 3.5: Message structure for *Aggregate*.

timestamp (cf. Algorithm 2 Line 15 to 18). Each peer sends its list periodically to its neighbors, with period  $\tau$ . We use *Aggregate* as a base-line algorithm. *Aggregate* has the property of being optimal in terms of time required until all peers have agreed on the same reference (average) playback timestamp.

In the following we are going to show that *Aggregate* has the property of being optimal in the sense of time required until all peers have agreed on the reference playback timestamp. Therefore, we are going to introduce some definitions that are needed for the following Theorem 3.3.1.

**Definition 2 (Undirected weighted network graph)** Let  $G(V, E, W)$  be an undirected weighted network graph (henceforth network graph) with  $V$  the set of vertices,  $E : V \times V$  the set of edges, and the set  $W$  with  $E \times \mathbb{R}$  that assigns an edge  $(v_i, v_j) \in E$  the weight  $w_k$  where  $w_k = \max\{\tau, \delta(v_i, v_j)\}$ ,  $\delta(v_i, v_j)$  denotes the channel delay for the channel between vertex  $v_i$  and  $v_j$  and  $\tau$  denotes the period.

**Definition 3 (Radius and diameter)** Given  $G(V, E, W)$  let the matrix  $S$  with  $|V|^2$  entries be the matrix that states the shortest path from  $v_i$  to  $v_j$ ,  $\forall v_i, v_j \in V$  and let  $s : |V| \times |V| \rightarrow \mathbb{R}$  be the function that returns the delay for a shortest path between  $v_i$  and  $v_j$ . Then we call  $\max_i \{\max_j \{s(i, j)\}\}$  the diameter  $d$  and  $\min_i \{\max_j \{s(i, j)\}\}$  the radius  $r$  of  $G$  with respect to Definition 2.



**Theorem 3.3.1** Suppose an undirected weighted network graph  $G(V, E, W)$  and we further assume that each edge that represents a connection between two peers has enough bandwidth available. The path  $\pi = \{e_j, \dots, e_m\}$  consisting of edges  $e_i \in E$  with  $k$  edges between two vertices that represents the diameter of the network graph, then the time  $T$  needed by *Aggregate* until all peers have agreed on a reference playback timestamp is given by:

$$T = \sum_{e_i \in \pi} w_i \quad (3.2)$$

**Proof of Theorem 3.3.1** Suppose an undirected network graph  $G(V, E, D)$  with diameter  $d$ . Let  $\pi = \{e_j, \dots, e_m\}$  be the path from  $v_a$  to  $v_b$  consisting of edges  $e_i \in E$  and having  $k$  edges that represent the diameter  $d$ . According to the definition of Algorithm 2 each peer *sends* its timestamp to all other peers.

- Suppose that  $T < \sum_{\forall e_i \in \pi} w_i$ . Thus, there has to be a shortest path  $\pi'$  from  $v_a$  to  $v_b$  in the network graph which consists of  $k'$  edges with  $s'(a, b) < s(a, b) = d$  and such that all peers received all playback timestamp within  $\sum_{e'_i \in \pi'} w_i$ . This is a contradiction.
- Suppose that  $T > \sum_{e_i \in \pi} w_i$ . Thus, there has to be another shortest path  $\pi'$  from  $v_{a'}$  to  $v_{b'}$  in the network graph which consists of  $k'$  edges such that  $s(a', b') > d$ . Due to the assumption that  $d = \max_i \{ \max_j \{ s(i, j) \} \}, \forall i, j \in |V| : d \geq s(i, j)$  we have:

$$s_{a', b'} > \max_i \{ \max_j \{ s(i, j) \} \} \geq s(i, j) \forall i, j \in |V|,$$

it follows that  $s(a', b') = s(a, b)$ . This concludes the proof.

□

In Section 3.4.1 we experimentally show how *Aggregate* performs in terms of overhead produced. Due to the high overhead generated by *Aggregate*, we introduce an algorithm for which *Aggregate* depicts a sharp lower bound for the time required by

the synchronization process but reduces the overhead produced during the synchronization process. If the synchronization protocol utilizes too much bandwidth, there may be too less bandwidth available in order to provide a high QoE with respect to the actual streaming of multimedia content or even stalls may be the result.

### 3.3.3 Merge and Forward

As already mentioned, in our approach, the fine synchronization phase starts once playback commences at the segment determined by *coarse synchronization* with the goal of agreeing on a reference timestamp to which all the peers in an IDMS session should synchronize. In contrast to the base-line approach we introduced in Section 3.3.2, we propose *Merge and Forward*, a flooding-based algorithm that calculates the average playback timestamp among the peers in a distributed and self-organized manner. Here, the average playback timestamp is utilized because it favors neither the peers already within the IDMS session nor those that have just joined. Selecting the minimum would privilege peers that recently joined an IDMS session and it will force all other peers to synchronize to this playback timestamp. The maximum will have the opposite effect. *Merge and Forward* focuses on reducing the overhead introduced when exchanging playback timestamps using unicast between all peers in contrast to other algorithms which use multicast [35]. Furthermore, by avoiding *pure flooding* it maintains media throughput and thus the QoE.

For tracking which and how many peers have contributed to the average playback timestamp we use a Bloom filter. Therefore, each peer uses the same set of hash functions  $h_1(x), \dots, h_k(x)$  for computing the bits to set when inserting itself into the Bloom filter [70]. We suggest to use hash functions with a low collision probability (e.g., Murmur Hash, SHA-1, SHA-2). For our purpose we used the Secure Hash Algorithm-1 (SHA-1) as hash function. This allows us to exchange a fixed length packet and contributes to the scalability of our P2P approach. Instead of a Bloom filter for indicating which peers have contributed to the current average playback timestamp, we could have used a simple bit-field and set the bits accordingly to the peer ids. However, if we have a high churn rate, where many new peers join an IDMS session and many peers leave the session, every time the peer id exceeds the length

---

**Algorithm 3** Merge and Forward.

---

```

     $B_i, L_i, BL_i, P_i, NTP_i, I_i^M, I_i^m, S_i, C_i \leftarrow 1$ 
1: function broadcastToNeighbors
2:   update( $P_i, NTP_i$ )
3:   for all  $p \in \text{peers}$  do
4:     sendPacket( $P_i, NTP_i, I_i^M, I_i^m, S_i, C_i, B_i$ )
5:   end for
6: end function
7: function receiveBloomFilter( $P_j, NTP_j, I_j^M, I_j^m, S_j, C_j, B_j$ )
8:   if  $S_j > S_i$  then
9:      $L_i \leftarrow \emptyset, S_i \leftarrow S_j, C_i \leftarrow 1$ 
10:     $B_i \leftarrow H(i)$ 
11:   end if
12:   if  $\text{bits}(B_j) > \text{bits}(B_i)$  then
13:     increaseSize( $B_i, \text{bits}(B_j)$ )
14:   end if
15:   update( $P_i, NTP_i$ )
16:   update( $P_j, NTP_j$ )
17:   if  $B_i \oplus B_j \neq \emptyset \wedge B_i \cap B_j = \emptyset \wedge B_j \notin L_i$  then
18:      $B_i \leftarrow B_i + B_j, P_i \leftarrow \frac{P_i \cdot C_i + P_j \cdot C_j}{C_i + C_j}$ 
19:      $I_i^M \leftarrow \max\{I_i^M, I_j^M\}$ 
20:      $I_i^m \leftarrow \min\{I_i^m, I_j^m\}$ 
21:      $C_i \leftarrow C_i + C_j$ 
22:   end if
23:   if  $B_i \oplus B_j \neq \emptyset \wedge B_i \cap B_j \neq \emptyset \wedge B_j \notin L_i$  then
24:     if  $C_j \geq C_i \wedge i \in B_i \cap B_j$  then
25:        $B_i \leftarrow B_j, P_i \leftarrow P_j, C_i \leftarrow C_j$ 
26:     else if  $C_j \geq C_i$  then
27:        $B_i \leftarrow B_j + H(i)$ 
28:        $P_i \leftarrow \frac{P_j \cdot C_j + P_j}{C_j + 1}$ 
29:        $C_i \leftarrow C_j + 1$ 
30:     end if
31:      $I_i^M \leftarrow \max\{I_i^M, I_j^M\}$ 
32:      $I_i^m \leftarrow \min\{I_i^m, I_j^m\}$ 
33:   end if
34:   if  $\#B_i - C_i > 0$  then
35:     increaseSizeAndTest( $B_i, BL_i, C_i, I_i^M, I_i^m$ )
36:     return
37:   end if
38:    $L_i \leftarrow \{B_j\} \cup L_i$ 
39:    $BL_i \leftarrow \{B_i, C_i, P_i, NTP_i\}$ 
40: end function

```

---

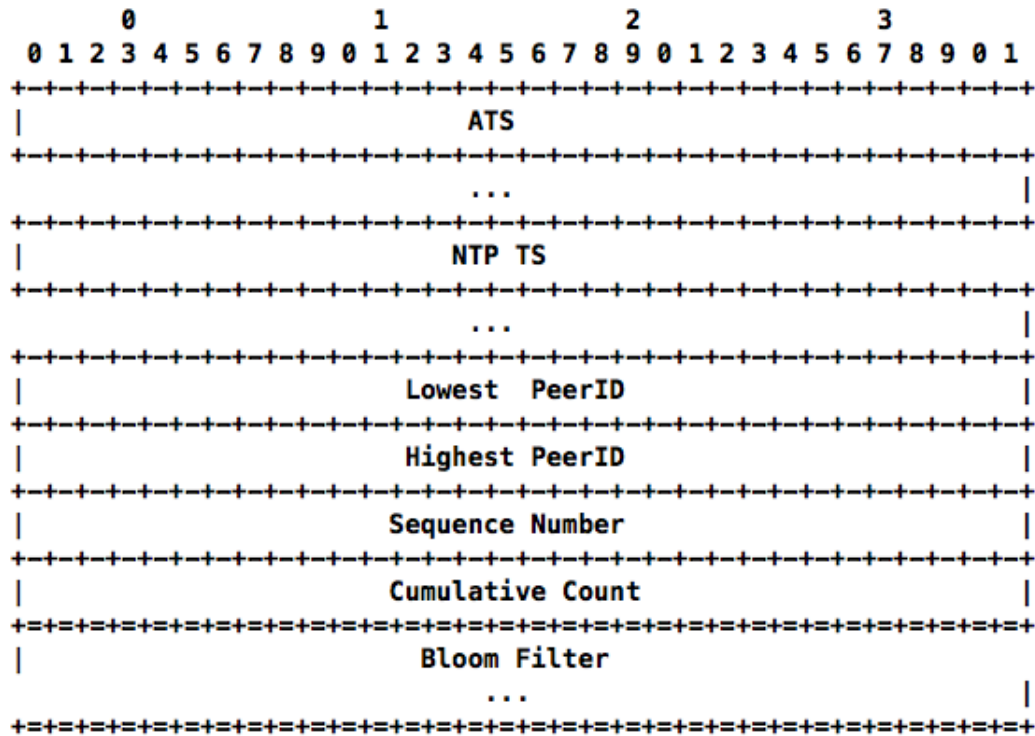


Figure 3.6: Message structure for the *fine synchronization*.

of the bit-field we would have to increase its length which will lead to the fact that parts of the bit-field are unused. Therefore, we use a Bloom filter to keep track which peers have already contributed to the reference playback timestamp.

Figure 3.6 depicts the message structure for the P2P communication once the overlay has been constructed by the *coarse synchronization* with the following semantics:

- **ATS**: average playback timestamp;
- **NTP TS**: NTP timestamp for aligning the playback timestamp;
- **Lowest PeerID**: lowest peer identification number seen by the sending peer according to the ISO;
- **Highest PeerID**: highest peer identification number seen by the sending peer according to the ISO;
- **Sequence Number**: indicates the current synchronization round;

- **Cumulative Count**: indicates the number of peers that contributed to the average playback timestamp;
- **Bloom filter**: fixed length Bloom filter with a length of  $m$  bits.

The *PeerID*, which consists of the lowest and highest peer id seen, is used later for determining how many peers are in a certain Bloom filter. The overall size of such a message is  $32 + \frac{m}{8}$  bytes. The *Sequence Number* allows the peers to trigger a re-synchronization by increasing the sequence number (e.g., due to asynchronism or MPD update). Other peers receiving a message with a higher sequence number than their own sequence number will reset to the initial condition and start over. The P2P algorithm making use of this message structure is shown in Algorithm 3 and referred to as *Merge and Forward (M&F)*. *Merge and Forward* is named after its operations because it **merges** incoming Bloom filters and **forwards** them to its neighbors.

Each peer  $i$  maintains the following variables and lists (cf. Algorithm 3):

- $B_i$  denotes the Bloom filter, each peer initially inserts itself with its own peer id according to the indices obtained by the use of a common set of  $k$  hash functions  $h_1(x), \dots, h_k(x)$ , the actual index is then determined by  $f_n(x) = h_n(x) \text{ MOD } size, \forall 1 \leq n \leq k$ , where *MOD* denotes the modulo operation and *size* denotes the actual size of the Bloom filter in bit (henceforth we use  $h_n(x)$  interchangeably to  $f_n(x)$ );
- $L_i$  denotes the list of already seen Bloom filters;
- $P_i$  denotes the PTS;
- $NTP_i$  denotes the NTP timestamps;
- $S_i$  denotes the current sequence number ( $S_i \in \mathbb{N}$ , initially set to zero);
- $C_i$  denotes the cumulative count which is initially set to zero;
- $I_i^m$  denotes the lowest peer id seen by peer  $i$ ;
- $I_i^M$  denotes the highest peer id seen by peer  $i$ ;
- $H(x)$  denotes the function that generates the bit-sequence for peer  $x$  by applying  $h_1(x), \dots, h_k(x)$ .

Each peer **forwards**  $B_i, P_i, I_i^M, I_i^m$  and  $C_i$  periodically to its neighbors depicted by the function *broadcastToNeighbors*, with period  $\tau$  (cf. Algorithm 3 Line 1 to 6). Please note, that the period must not be the same for every peer. If peer  $i$  receives a message from one of its neighbors it first check whether the sequence number  $S_j$  is greater than  $S_i$ , if this is the case peer  $i$  clears its Bloom filter and re-initializes all variables and sets its sequence number to the received one (cf. Algorithm 3 Line 8 to 11).

If peer  $i$  receives a Bloom filter from one of its neighbors  $j$ , it first compares the size of its Bloom filter and the received Bloom filter and increases the size of its Bloom filter to the size of the received Bloom filter if this one is bigger (cf. Algorithm 3 Line 12 to 14). Furthermore, peer  $i$  checks whether it can **merge** the Bloom filters (cf. Algorithm 3 Line 17 to 33). The Bloom filters can only be merged if they are disjoint to avoid introducing a bias to the resulting weighted average. If the Bloom filters  $B_i$  and  $B_j$  are distinct in terms of  $B_i \cap B_j = \emptyset$  (cf. Algorithm 3 Line 17) then we can merge the Bloom filters using the bit-wise *OR* depicted by  $+$  and we calculate the weighted average between  $P_i$  and  $P_j$  (cf. Algorithm 3 Line 18). *Merge and Forward* counts how many peers have contributed to the average playback timestamp denoted by  $C_i$  and in the case if two distinct Bloom filters are merged the cumulative count is calculated according to Algorithm 3 Line 21.

If the Bloom filters are not disjoint and if we have not seen the received Bloom filter yet we have two further cases (cf. Algorithm 3 Line 17). First, if  $C_j \geq C_i$  and if peer  $i$  is already in both of the Bloom filters  $B_i$  and  $B_j$  we keep the more recent one (cf. Algorithm 3 Line 24 and Line 25). If  $C_i \geq C_j$  but  $i \notin B_i \cap B_j$  peer  $i$  adds itself to  $B_j$  and calculates the weighted average (cf. Algorithm 3 Line 26 to Line 30).

The highest peer id  $I_i^M$  is set to the maximum of  $I_i^M$  and  $I_j^M$ , and  $I_i^m$  is set to the minimum of  $I_i^m$  and  $I_j^m$  (cf. Algorithm 3 Line 19 to 20 and Line 31 to 30). After a synchronization round has finished (all peers hold the same reference playback timestamp), a peer may trigger a new synchronization round by increasing  $S_i$ . The re-synchronization may be triggered if a peer has paused the playback of the multimedia content or if it is unable to synchronize its playback to the negotiated reference. In the latter case, the peer may increase the importance of its timestamp by introducing

a weight (e.g.,  $PTS = w_i \cdot PTS$ ,  $w_i \geq 1$ ). Finding appropriate weights and detected faulty behavior is out of scope of this work.  $C_i$  depicts the number of peers that have already contributed to  $P_i$ .  $BL_i$  denotes the backlog where the Bloom filter that have not caused any false positives are stored (cf. Algorithm 3 Line 39).

In order to calculate the overlap or the intersection of two Bloom filters ( $B_i \cap B_j$ ), *Merge and Forward* has to know **which peers** have already been inserted. Due to the nature of Bloom filters the calculation of the number of peers in a filter may identify peers that were not inserted (false positives). Consider a test function  $test(B, x)$  that returns true if peer  $x$  was inserted into the Bloom filter. Let's assume we know that peer  $x$  is not in Bloom filter  $B$  and that when we receive the Bloom filter of size  $m$ ,  $s$  bits are set by the use of  $k$  hash functions. If we test whether peer  $x$  was inserted into the Bloom filter and this test returns true we have encountered a false positive. Furthermore, we assume that the probability that a bit is set by a hash function is distributed uniformly with  $p = \frac{1}{m}$ . Furthermore, the probability that an already set bit is set by  $h_i(x)$  assuming that  $s$  bits are set is  $\sum_{t=1}^s \frac{1}{m} = \frac{s}{m}$ . If we receive a Bloom filter with  $s$  bits set and if we test the existence of a specific peer the probability to encounter a false positive is  $p_{false} := P(\bigcap_{i=1}^k h_i(x)) = \prod_{i=1}^k \frac{s}{m} = (\frac{s}{m})^k$  [70]. For reducing the probability of encountering a false positive we have introduced  $I_j^M$  and  $I_j^m$ . Especially, if a peer receives the Bloom filter containing only a single peer. Then  $I_j^M - I_j^m = 0$  and, therefore, only the peer with id  $I_j^M$  or  $I_j^m$  is in the Bloom filter.

Since we are keeping track of how many peers have already contributed to the Bloom filter using  $C_i$ , we can detect whether we encountered a false positive when testing for the ids of the contributed peers. If peer  $i$  receives the Bloom filter of peer  $j$ , it tries to determine which and how many peers ( $p_1, p_2, \dots, p_n$ ) have contributed to the average playback timestamp  $P_j$ . Assuming that peer  $i$  tests for  $N = I_j^M - I_j^m + 1$  peers and  $C_j$  peers were really inserted in the Bloom Filter ( $C_j \leq N$ ) the false positive rate is given by:  $P_{false}(p_1 \vee p_2 \vee \dots \vee p_n) = \sum_{i=1}^{N-C} (\frac{s}{m})^k = (N - C) \cdot (\frac{s}{m})^k$  (due to independence).  $\#B_i$  denotes the number of peers found in Bloom filter  $B_i$  by testing for the membership of  $I_j^M - I_j^m + 1$  peers. *Merge and Forward* is able to detect whether false positives occurred by testing if  $\#B_i > C_i$  (cf. Algorithm 3 Line 34). If  $\#B_i - C_i > 0$  we employ a mechanism that dynamically increases the size of the

Bloom filter denoted by  $increaseSizeAndTest(B_i, BL_i, I_i^M, I_i^m)$  (cf. Algorithm 3 Line 35).

**Proposition 3.3.1** *If  $h_k(x)$ ,  $x \in \mathbb{N}$  are  $k$  uniformly distributed hash functions then it suffices to test for  $C_i$  distinct peer ids whether a false positive occurs.*

**Proof of Proposition 3.3.1** *Since  $h_k(x)$  are  $k$  uniformly distributed hash functions each peer id has the same probability to set a specific bit in the Bloom filter with  $s$  bits set and an overall size of  $m$  bits. Suppose any two sets  $M = \{x_1, \dots, x_{C_i}\}, N = \{y_1, \dots, y_{C_i}\}, M, N \subset \mathbb{N}$  of peer ids with cardinality  $C_i, \forall x \in M : x \notin M \setminus x, \forall y \in N : y \notin N \setminus y$ , and  $M \cap N = \emptyset$ . Assume that the probability of encountering a false positive using set  $M$  is  $p_m$  and when using set  $N$  it is  $p_n$ . If the peer id would have an impact on the probability of encountering a false positive we would see that  $p_m \neq p_n$  but, since the  $k$  hash functions provide uniformly distributed indices (with  $p = \frac{1}{m}$ ) we have:*

$$p_m = P\left(\bigcup_{r=1}^{C_i} \bigcap_{t=1}^k h_t(x_r)\right) = \sum_{r=1}^{C_i} \prod_{t=1}^k \frac{s}{m} = C_i \cdot \left(\frac{s}{m}\right)^k = P\left(\bigcup_{r=1}^{C_i} \bigcap_{t=1}^k h_t(y_r)\right) = p_n$$

*This concludes the proof.* □

If the test for false positives shows that the newly allocated Bloom filter does not produce any false positives for the test configuration, we use the backlog  $BL_i$  and add the peers that are in the newest backlog entry to our new Bloom filter depicted by  $increaseSizeAndTest$  (cf. Algorithm 3 Line 35). This mechanism avoids that false positives have an impact on the calculated reference playback timestamp. Every peer that sees a Bloom filter bigger than its own, simply increases the size of its Bloom filter and restores the state of the old Bloom filter by inserting the peers accordingly. This mechanism prohibits the occurrence of false positives when testing the membership of peers (especially when testing whether  $i \in A \cap B$ ). Nevertheless, this may increase the time required by the synchronization process. Therefore, we suggest to select appropriate values for the size  $m$  of the Bloom filter and the number of hash functions  $k$  in the beginning. With this mechanism in place we can assume that the **resulting average playback timestamp is not biased by false positives.**



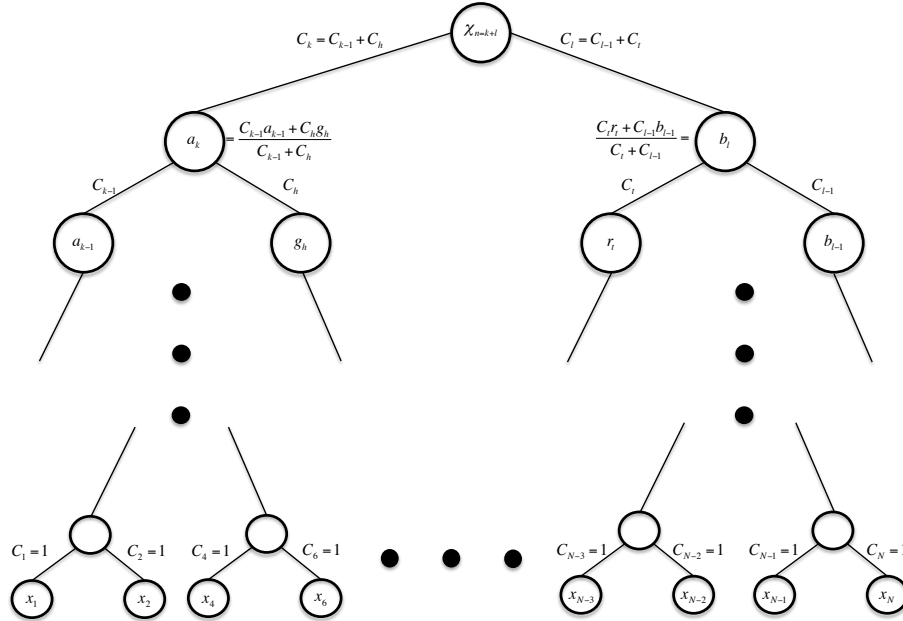


Figure 3.7: Construction of the weighted average  $\chi_n$  using *Merge and Forward*.

According to the definition of *Merge and Forward* we state the following theorem:

**Theorem 3.3.2** The weighted average  $\chi_n$  calculated by the peers that *merge* Bloom filters for the playback timestamps  $x_i \geq 0, 1 \leq i \leq N$  which is the result of  $n$  merges to which  $N$  peers contributed, converges to the average playback timestamp of the  $N$  peers and  $\chi_n$  is unique for a given set of peers.

**Proof of Theorem 3.3.2** Thanks to Proposition 3.3.1 and the previously introduced *increaseSizeAndTest*( $B_i, BL_i, I_i^M, I_i^m$ ) (cf. Algorithm 3 Line 35) there are no errors introduced during the calculation of the weighted average of two timestamps. Thus, according to *Merge and Forward* the construction of  $\chi_n$  is as follows (cf. Figure 3.7).

$$\begin{aligned} \chi_n &= \frac{1}{C_k + C_l} \cdot (C_k \cdot a_k + C_l \cdot b_l) = \\ &= \frac{1}{C_k + C_l} \cdot \left( C_k \frac{C_{k-1} \cdot a_{k-1} + C_h \cdot g_h}{C_{k-1} + C_h} + C_l \frac{C_{l-1} \cdot b_{l-1} + C_t \cdot r_t}{C_{l-1} + C_t} \right) =, \end{aligned}$$

where  $C_k$  (included in the message cf. Figure 3.6) denotes the weight for the sub tree from which  $a_k$  resulted and  $C_h$  is the corresponding weight for the sub tree from which  $b_h$  resulted. The number of merges that happened in the sub tree of  $a_k$  is  $k$  and for the sub tree of  $b_h$  we have  $l$  merges, thus,  $n = k + l = N - 1$  and the binary tree has  $\lfloor \log(N) \rfloor + 1$  planes. Note, that  $C_k = C_{k-1} + C_h$  (and so forth) is the number of peers that have contributed to  $a_k$ . Thus, the denominators and the corresponding  $C_i$  can be canceled. Following each sub tree recursively to its leaf we see the original playback timestamps of the corresponding peers for which the  $C_i$ s are 1 as initially set by each peer (cf. Figure 3.7 and Algorithm 3). Let  $\Psi = \frac{1}{N} \sum_{i=1}^N x_i$  is the average playback timestamp of the  $N$  playback timestamps.

$$\begin{aligned}
 &= \frac{1}{C_k + C_l} \cdot (C_{k-1} \cdot a_{k-1} + \dots + C_{l-v} \cdot g_{l-v} + \dots + C_{h-1} \cdot b_{h-1} + C_t \cdot r_t) = \\
 &= \frac{1}{C_k + C_l} \cdot (C_1 \cdot x_1 + \dots + C_{l-v} \cdot g_{l-v} + \dots + C_{h-d} \cdot b_{h-d}) = \\
 &= \frac{1}{C_k + C_l} \cdot (C_1 \cdot x_1 + C_2 \cdot x_2 + \dots + C_{N-1} \cdot x_N + C_N \cdot x_N) = \frac{1}{N} \sum_{i=1}^N x_i = \Psi \quad (3.3)
 \end{aligned}$$

Thus,  $\chi_k \xrightarrow{k \rightarrow n} \Psi$ . It remains to show that the resulting average timestamp is unique. Therefore, suppose that  $\theta_n$  is calculated by *Merge and Forward* for the same number  $N$  of peers with the same playback timestamps  $x_1, x_2, \dots, x_N$  with  $x_i \geq 0$ , but for a *rewired* network  $G'(V, E', W')$ . Again, we take a look at the tree depicted in Figure 3.7 for which  $\theta_n$  depicts the root with  $n = f + g = N - 1$ . Assume that this *rewiring* changes the construction of  $\theta_n$  such that it is different from  $\chi_n$ ,  $|\chi_n - \theta_n| > 0$ . Again, we look at the tree depicted in Figure 3.7, for which  $\theta_n$  depicts the root with  $n = f + g = N - 1$ . It immediately follows from Equation 3.3 that  $|\chi_n - \theta_n| > 0$  contradicts the commutativity of the summation. Therefore, the weighted average playback timestamp computed by *Merge and Forward* is always unique regardless of the order of the *merges*. This concludes the proof.  $\square$

The time that *Merge and Forward* requires until the peers have agreed on the average playback timestamp is greater or equal to the time that is required by *Aggregate* and is given by Equation 3.4. This can be easily verified because the minimum time required corresponds to the longest of the shortest paths (the diameter) in terms of

transmission time between all peers. Since, Bloom filters are employed to keep track of peers that already contributed, *Merge and Forward* is not able to merge Bloom filters that overlap (otherwise we would introduce errors). Thus, its required time is greater or equal to the time required by *Aggregate* (cf. Theorem 3.3.1).

$$T_{M\&F} \geq \sum_{e_i \in \pi} w_i \quad (3.4)$$

A possible method to detect whether a peer holds already the final playback timestamp may be done by counting how often in a row the peer has received the same Bloom filter by its neighbors. If this exceeds a certain threshold the peer triggers the adjustment of the playback rate. But, the peer keeps receiving and periodically sending messages to its neighbors such that in case it has synchronized to an intermediate weighted average playback timestamp (by spuriously triggering the adjustment of the playback rate) it still can calculate the correct reference playback timestamp.

## 3.4 Experimental Results

First, we investigate the proposed DCS approach by simulation with respect to the traffic generated during the negotiation on the reference playback timestamp and the time needed until all peers have the necessary information for calculating the average playback timestamp. Second, we conduct a subjective quality assessment for evaluating the introduced distortion metric and the dynamic AMP approach.

### 3.4.1 Overhead Analysis

We compare *Merge and Forward* to *Aggregate* under various conditions (e.g., without cross-traffic, with cross-traffic) and with different parameters. Since we put no upper limits on the number of peers in our overlay network nor does *Merge and Forward* put any restriction on the number of peers, we decided to use 40, 60 and 80 peers for the evaluations/simulations. We will further investigate the traffic generated using *Merge and Forward* by a single peer assuming a maximum false positive rate  $p_{false}^*$  and a

specific number of hash functions  $k$ . We simulated the algorithms on Erdős-Rényi random networks [71] implemented using OMNeT++ and the INET framework [72]. Erdős-Rényi random networks mimic the *random* creation of an unstructured P2P overlay network using an unreliable communication such as UDP. A period  $\tau$  of 250 milliseconds was used for both algorithms, with the round trip time set to 80 milliseconds and a maximum clock skew of 30 milliseconds randomly (uniformly) chosen from an interval of  $[-15, 15]$  milliseconds. The random network mimics the occurrence of packet loss during the *coarse synchronization* and, thus, leading to an overlay networks with different connectivities. We further assume unicast only for our simulations. For multicast we provide a discussion at the end of this section. For each of the parameter settings we conducted 30 simulation runs and taking the average of the results. The simulations shall provide insights on the worst case performance of *Merge and Forward*. Our evaluations cover the case if there are many peers in the overlay network and if they have not yet (nor partly) agreed on a reference.

First, we investigate how *Merge and Forward* performs in terms of overhead (traffic generated in kbit/s) if the size of the Bloom filter is selected big enough in the beginning such that the probability of increasing the size of the Bloom filter due to false positives is very low. Furthermore, we set the number of hash functions used to  $k = 4$ . In order to investigate the overhead we do not restrict the bandwidth of the peers.

Figures 3.8, 3.9 and 3.10 depict the traffic in kbit/s generated by *Aggregate* and *Merge and Forward* until all peers hold the same reference playback timestamp for 40, 60 and 80 peers for a specific connectivity interval, respectively. The connectivity is given by Equation 3.5.

$$c = \frac{\frac{1}{|V|} \sum_{i=1}^{|V|} d_G(v)}{|V| - 1}, \quad (3.5)$$

where  $V$  denotes the set of vertices and  $d_G(v)$  denotes the degree of node  $v \in V$  (in terms of edges). *Merge and Forward* outperforms *Aggregate* regardless of the connectivity. The overhead generated by *Aggregate* increases faster than it does with *Merge and Forward* (cf. Figures 3.8, 3.9 and 3.10). With an increase in the number of peers (80 peers) *Aggregate* starts to generate even more overhead, approximately four times the overhead *Merge and Forward* generates (cf. Figure 3.10).

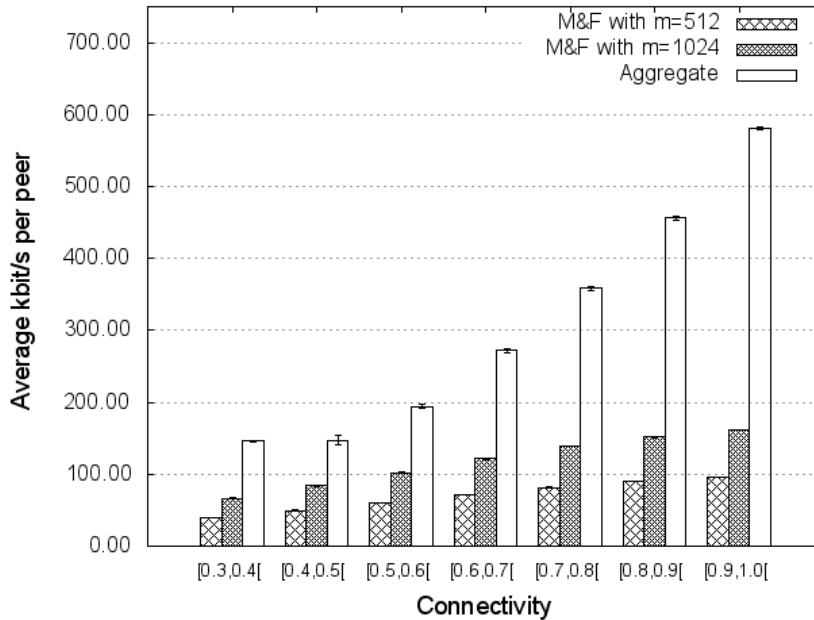


Figure 3.8: Traffic generated per second per peer until 40 peers have the same reference playback timestamp using *Aggregate* and *Merge and Forward* (M&F) (95% CI).

After agreeing on the reference playback timestamp the peers keep exchanging messages according to *Merge and Forward* and *Aggregate* in order to negotiate with peers that may join the P2P overlay network in the future or to allow for a re-synchronization. Therefore, we take a look at how big the Bloom filter can be at maximum if we consider the phase after agreeing on the reference playback timestamp. The traffic generated in bit per period (bpp) of all peers using *Aggregate*  $R_{Aggregate}$  (assuming that enough bandwidth is available between the peers) is given by:  $R_{Aggregate} = 8 \cdot (28 \cdot |V|) \cdot \sum_{v \in V} d_G(v) = 8 \cdot (28 \cdot |V|) \cdot |V| \cdot \bar{d}_G$ . The total traffic generated by *Merge and Forward*  $R_{M\&F}$  in bpp is given by  $R_{M\&F} = (32 \cdot 8 + m) \cdot \sum_{v \in V} d_G(v) = (32 \cdot 8 + m) \cdot |V| \cdot \bar{d}_G$  under the assumption that the size of the Bloom filter does not change during the synchronization process (or the selected Bloom filter is big enough to avoid false positives when inserting the peer ids using the  $k$  hash functions). If we assume that the peers keep exchanging the playback timestamps for both algorithms during the playback of the multimedia content then the maximum size of the Bloom filter such that the traffic caused by *Merge and Forward* is lower or equal compared to *Aggregate* is given by:  $m \leq 8 \cdot (28 \cdot |V| - 32)$ . This upper bound on the size for the Bloom

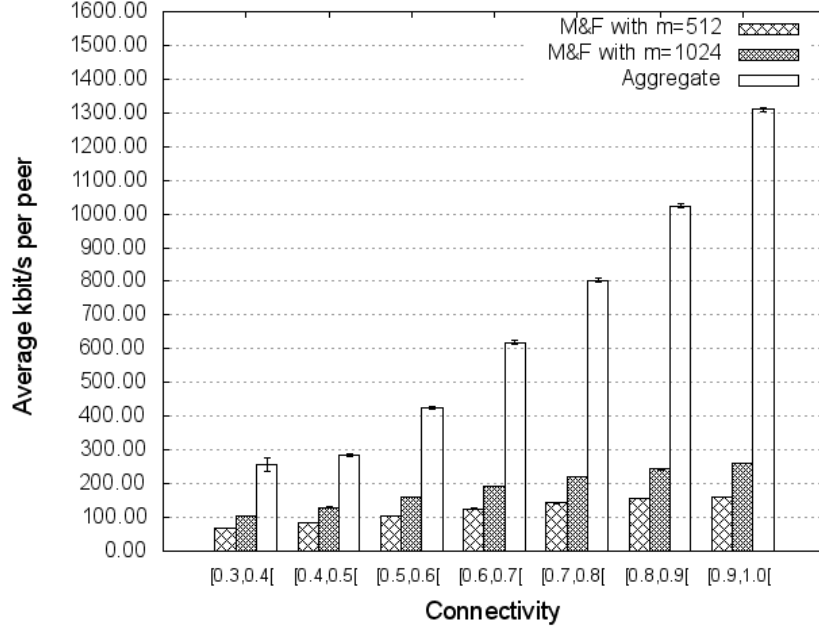


Figure 3.9: Traffic generated per second per peer until 60 peers have the same reference playback timestamp using *Aggregate* and *Merge and Forward* (M&F) (95% CI).

filter does imply a lower bound for the false positive rate. The false positive rate is thus bounded by  $p_{false} \geq (\frac{1}{m})^k$ . Let  $m$  be the maximum possible size, we have  $p_{false} \leq (\frac{1}{8 \cdot (28 \cdot |V| - 32)})^k$ . For example, if we consider an overlay network with 80 peers the maximum size of the Bloom filter could be 17888 bits wide, until *Merge and Forward* reaches the same amount of traffic as *Aggregate*, which corresponds to a lower bound for the false positive rate of  $p_{false} \geq 9.7 \cdot 10^{-18}$ . We selected a Bloom filter of 512 bits which is approximately 35 times smaller than the maximum allowed size.

$$\bar{R}_{M\&F} = \bar{d}_G \cdot (|V| \cdot k \cdot (p_{false}^*)^{-\frac{1}{k}} + 32 \cdot 8) \cdot \frac{1}{\tau} \quad (3.6)$$

Equation 3.6 denotes the average traffic in bit per second (bps) generated by a peer using *Merge and Forward*, assuming a maximum false positive rate of  $p_{false}^*$ ,  $|V|$  peers having  $|V| \geq 2$ ,  $k$  hash functions,  $\tau$  the period in seconds (equal for all peers, otherwise we set  $\tau = \min\{\tau_i\}$  and  $\bar{R}_{M\&F}$  denotes an upper bound for the average traffic), and an average edge degree of  $\bar{d}_G$ . If we assume a certain number of peers  $|V|$  and  $k$  hash functions the maximum false positive rate that will occur for a given size

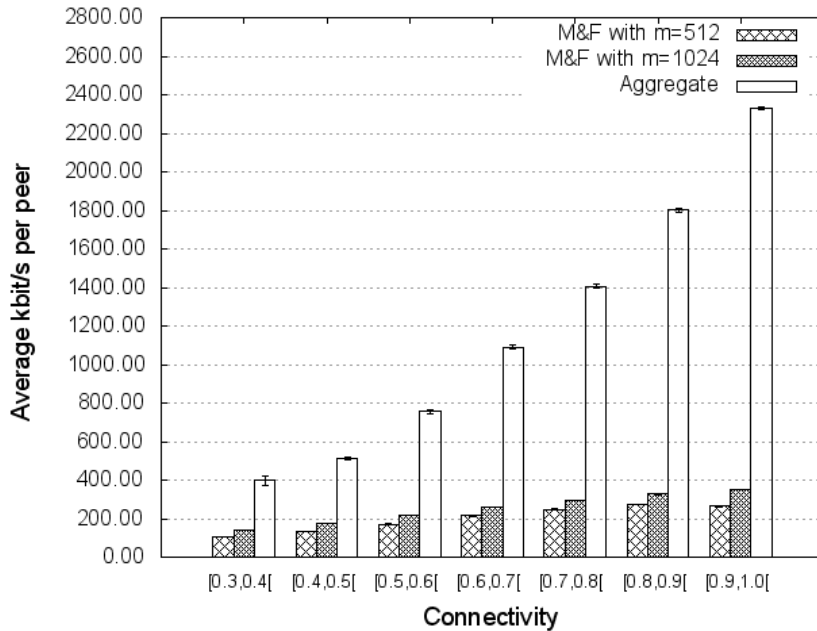


Figure 3.10: Traffic generated per second per peer until 80 peers have the same reference playback timestamp using *Aggregate* and *Merge and Forward* (M&F)(95% CI).

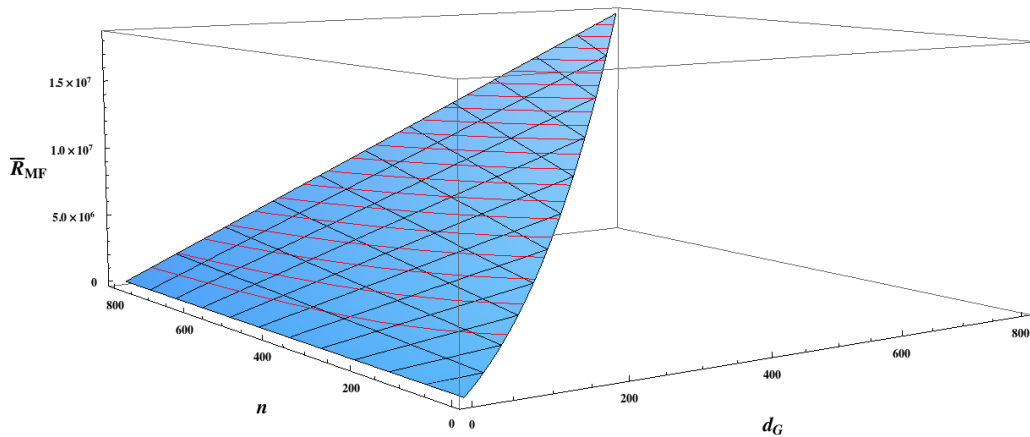


Figure 3.11: Traffic generated using *Merge and Forward* given by Equation 3.6 assuming a maximum false positive rate of  $p_{false}^* = 0.1$ .

$m$  of the Bloom filter is denoted by  $p_{false}^* = \left(\frac{k \cdot |V|}{m}\right)^k$ . Since, *Merge and Forward* will increase the size of the Bloom filter if it detects that false positives occur, we have to initially assume a small maximum false positive probability (e.g.,  $p_{false}^* \leq 0.1$ ). Figure 3.11 depicts the expected generated traffic in bps assuming that we use  $k = 4$  hash functions and a maximum false positive rate of  $p_{false}^* = 0.1$  for different values of

$n$  and  $\bar{d}_G$ . For a greater number of peers even *Merge and Forward* starts to generate a lot of additional traffic. For instance, for  $n = 800$  and an average degree of  $\bar{d}_G = 799$  we would generate approximately 18 Mbit/s assuming that  $p_{false}^* = 0.1$ , even if we use an average degree of  $\bar{d}_G = 400$  *Merge and Forward* would generate approximately 9.4 Mbit/s. In order to overcome this problem, *Merge and Forward* may employ a probabilistic sending scheme, e.g., that every peer does not send messages to all of its neighbors every period but only to a subset according to a certain distribution. Another approach would be to reduce the average degree of each peer by introducing a threshold on the number of connected peers. In the latter case it has to be ensured that the overlay network is not partitioned. At last, the period can be increased. In all cases the time until the peers agree on a reference playback timestamp will be increased. A detailed analysis on the time required by *Merge and Forward* with respect to  $\bar{d}_G$  or the connectivity  $c$  and considering the number of peers  $n$  is given in Section 3.4.2.

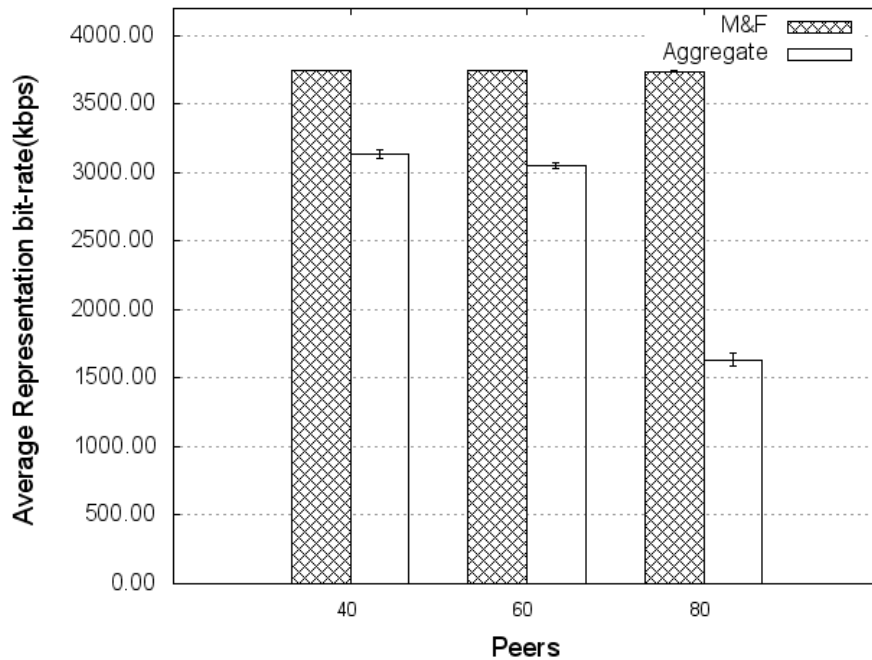


Figure 3.12: Average Representation bit-rate using an average connectivity of the peers between  $[0.3, 0.4[$  when using *Aggregate* and *Merge and Forward* (M&F) (95% CI).

We further investigate how the traffic generated by the synchronization protocols



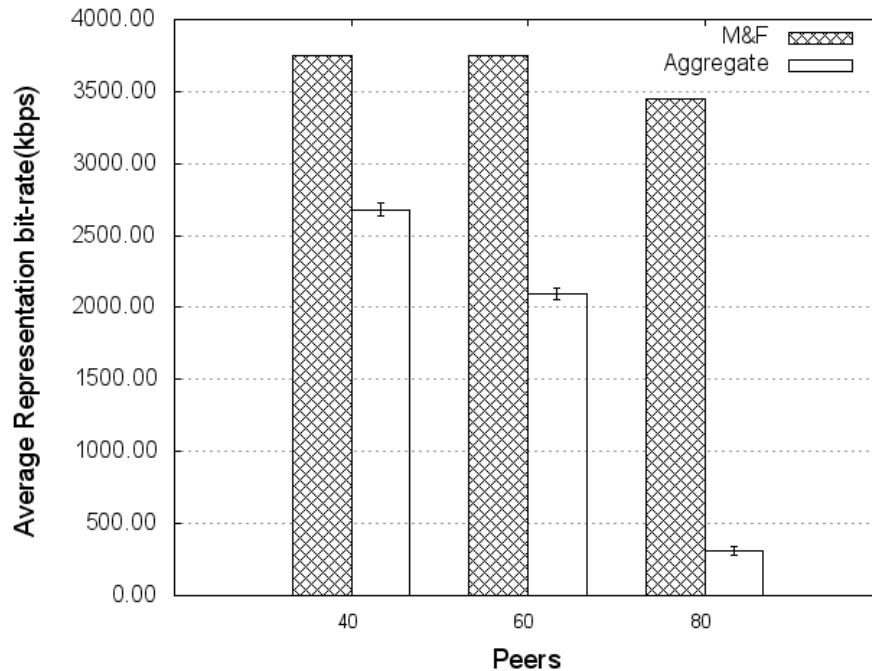


Figure 3.13: Average Representation bit-rate using an average connectivity of the peers between  $[0.6, 0.7[$  when using *Aggregate* and *Merge and Forward* (M&F) (95% CI).

influences the resulting average representation bit-rate when using MPEG-DASH. Therefore, we simulate the playback of multimedia content and the negotiation on a reference playback timestamp for 40, 60 and 80 peers using our simulation framework that is built upon OMNeT++ . We set a bandwidth of 4.5 Mbit/s (symmetrically) for each peer and a round trip time of 80 milliseconds. For creating random graphs we use connectivities within the following intervals:  $[0.3, 0.4[$ ,  $[0.6, 0.7[$ , and  $[0.9, 1.0[$ . Again, we conducted 30 simulation runs for each configuration. The size of the Bloom filter was set to 512 bit. As adaptation logic for the MPEG-DASH enabled player we used a rate-based adaptation logic that makes an adaptation decision upon estimating the available bandwidth at each peer. The bandwidth is estimated every time a segment is requested by calculating the weighted average of the measured available bandwidth. Equation 3.7 depicts the exponential average used to estimate the available bandwidth [73].

$$b_n = (1 - w) \cdot b_{n-1} + w \cdot b_m \quad (3.7)$$

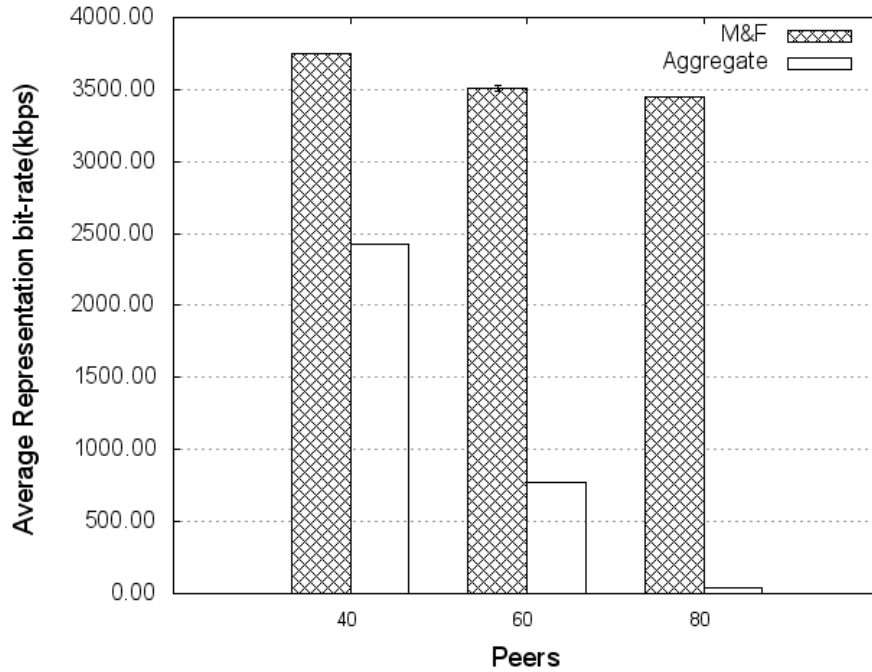


Figure 3.14: Average Representation bit-rate using an average connectivity of the peers between  $[0.9, 1.0[$  when using *Aggregate* and *Merge and Forward* (M&F) (95% CI).

, where  $b_{n-1}$  is the throughput calculated at the  $n - 1^{th}$  segment,  $b_n$  denotes the throughput measures when the  $n - 1^{th}$  segment is downloaded. For estimating  $b_0$  we measure the available bandwidth when downloading the MPD. For the weight we have selected following value  $w = 0.65$  which mimics an optimistic behavior. The multimedia content used was taken from BigBuckBunny [74]. We transcoded 19 representations with the following bit-rates (bit/s): 45652, 89283, 131087, 178351, 221600, 262537, 334349, 396126, 522286, 595491, 791182, 1244778, 1546902, 2133691, 2484135, 3078587, 3526922, 3840360 and 4219897. Furthermore, we use a segment size of two seconds. For determining the point in time  $t$  when peers join the IDMS session we assume that  $t \sim Exp(\lambda)$  with  $\lambda = \frac{1}{30}$ . Furthermore, we set the latest point at which peers enter the IDMS session to  $t = 100$ . The simulation duration was 600 seconds.

Figures 3.12, 3.13 and 3.14 depict the average representation bit-rate that each peer was able to retrieve while using *Aggregate* and *Merge and Forward*, respectively.

With lower connectivities the impact on the average representation bit-rate when using *Aggregate* is negligible (cf. 3.12 for 40 peers). But, if the number of peers increases and/or the connectivity of the overlay network increases the average representation bit-rate that can be retrieved by the peers decreases monotonically using *Aggregate*. *Merge and Forward* does always provide the same or a higher average representation bit-rate for all configurations of peers and connectivity. Especially, when the overlay network is well connected *Merge and Forward* outperforms *Aggregate* dramatically. *Aggregate* does not scale with the number of peers (cf. Figure 3.12, 3.13 and 3.14). For all the configurations using *Merge and Forward* the average representation bit-rate is above 3 Mbit/s.

$$R_{M\&F} = \frac{1}{\tau} \cdot (m + 32 \cdot 8) \quad (3.8)$$

If multicast is in place the overhead caused by the synchronization algorithms reduces dramatically at the peers because a peer sends only a single packet to many recipients. The connectivity of the network does no longer influence the generated overhead. But, still having an impact on the negotiation time. In the case of multicast the generated traffic of *Merge and Forward*  $R_{M\&F}$  at each peer is given by Equation 3.8. The point of inflection when *Merge and Forward* pays off compared to *Aggregate* using multicast and assuming that  $n$  peers are in the list that is maintained by *Aggregate* is given by  $n \geq \lceil \frac{m+32}{28} \rceil$ . The threshold of  $n$  peers depends linearly on the size of the selected Bloom filter.

### 3.4.2 Synchronization Time Analysis

For analyzing the time needed by *Merge and Forward* until all peers have agreed on the reference playback timestamp, we first compare *Merge and Forward* and *Aggregate* without any bandwidth limits set to the connection between the peers. Second, we investigate the impact of false positives on the required synchronization time for different start sizes of the Bloom filter of *Merge and Forward*.

For comparing *Merge and Forward* and *Aggregate* under optimal conditions we simulated the algorithms on Erdős-Renyi random networks [71] implemented using

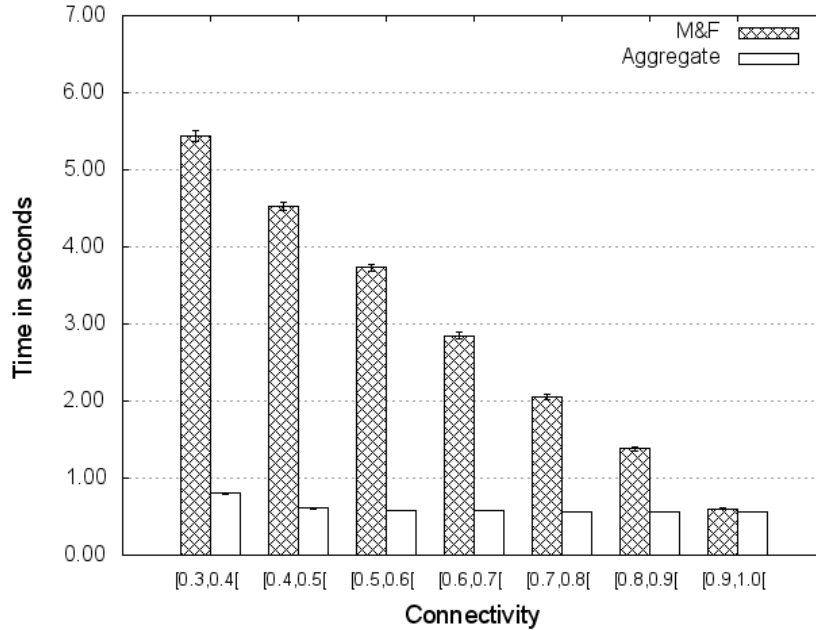


Figure 3.15: Time required for the distributed calculation, without cross-traffic, of the average playback timestamp for 40 peers using *Aggregate* and *Merge and Forward* (M&F) (95% CI).

OMNeT++ [72]. A period  $\tau$  of 250 milliseconds was used for both algorithms, with the round trip time set to 80 milliseconds and a maximum clock skew of 30 milliseconds randomly (uniformly) chosen from an interval of  $[-15, 15]$  milliseconds. The random network mimics the occurrence of packet loss during the *coarse synchronization* and, thus, leading to an overlay network with different connectivities. For each of the parameter settings we conducted 30 simulation runs taking the average of the results. Again, there was no bandwidth limit set for the peers such that the algorithms are evaluated under optimal conditions.

Figures 3.15, 3.16, and 3.17 depict the time until all peers have agreed on the reference playback timestamp averaged for different intervals of the connectivity of the overlay network. The figures show clearly that if the connectivity increases the required synchronization time decreases when using *Merge and Forward*. The Figures further provide an empirical example of Theorem 3.3.1 that states, that *Aggregate* is optimal in terms of time needed until all peers have negotiated if we assume unrestricted or enough bandwidth between the peers in the overlay network.

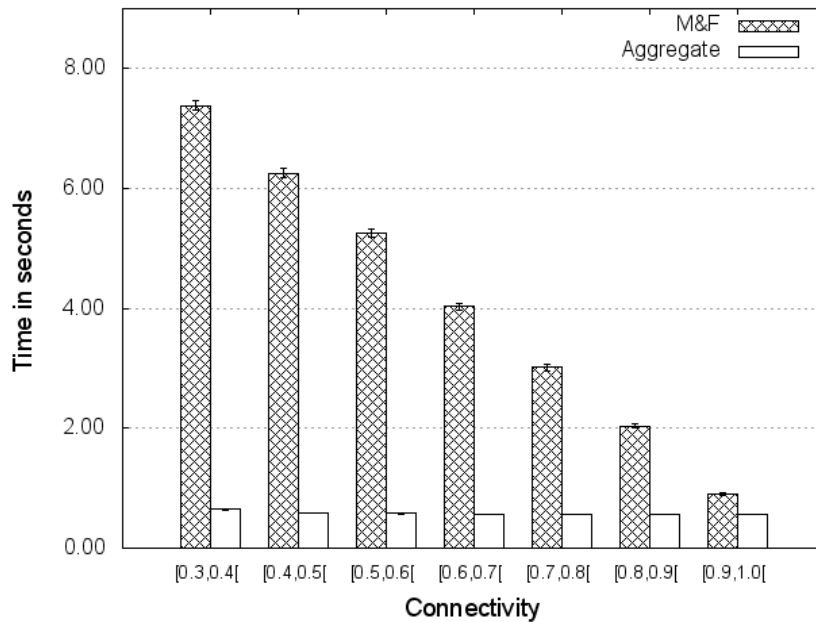


Figure 3.16: Time required for the distributed calculation, without cross-traffic, of the average playback timestamp for 60 peers using *Aggregate* and *Merge and Forward* (M&F) (95% CI).

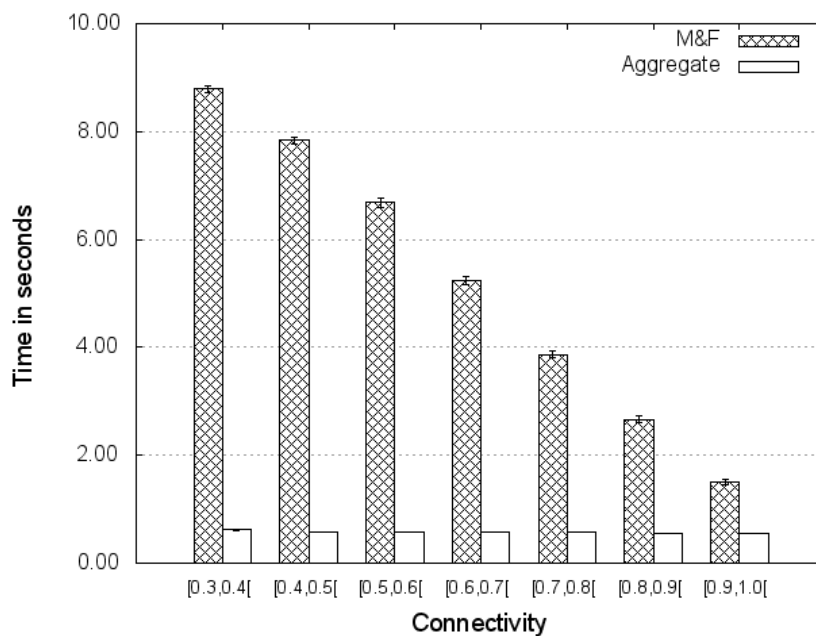


Figure 3.17: Time required for the distributed calculation, without cross-traffic, of the average playback timestamp for 80 peers using *Aggregate* and *Merge and Forward* (M&F) (95% CI).

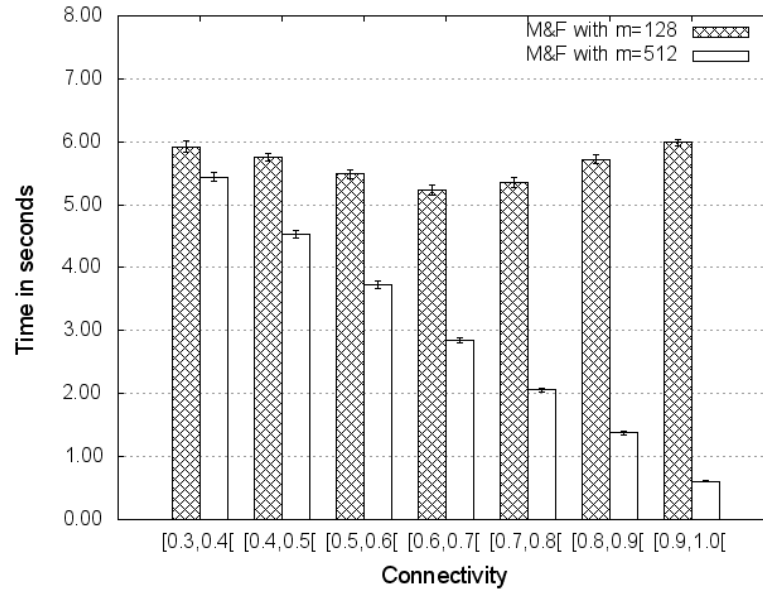


Figure 3.18: Comparison of the time required for the distributed calculation when using a Bloom filter size of 512 bit and 128 bit for 40 peers (95% CI).

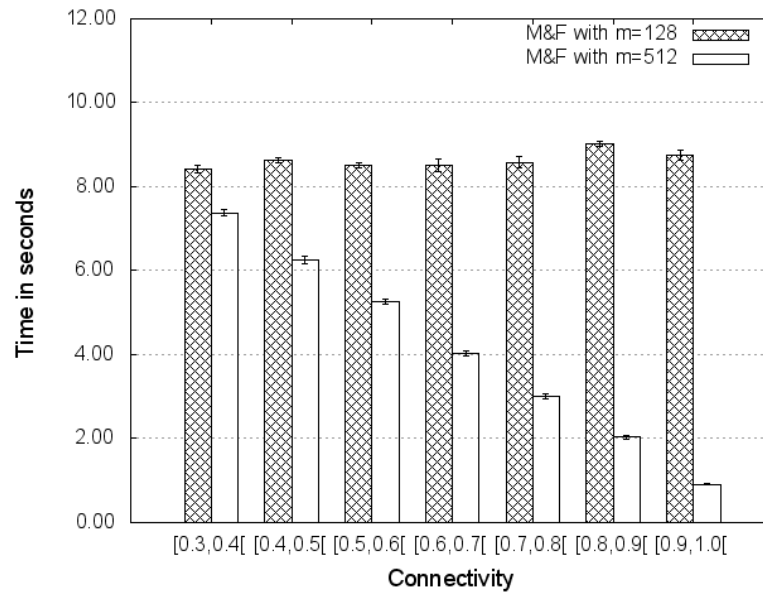


Figure 3.19: Comparison of the time required for the distributed calculation when using a Bloom filter size of 512 bit and 128 bit for 60 peers (95% CI).

If we consciously select a too small size for the Bloom filter false positives will occur and *Merge and Forward* will therefore increase the size of the Bloom filter.

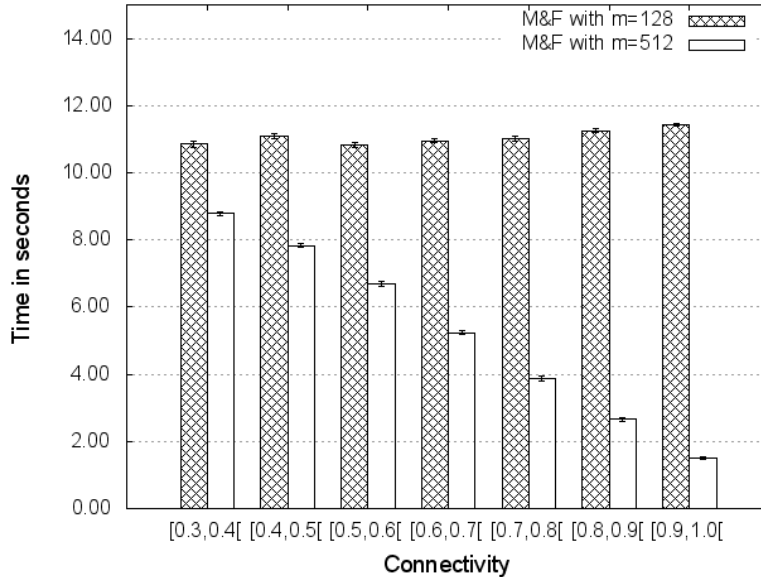


Figure 3.20: Comparison of the time required for the distributed calculation when using a Bloom filter size of 512 bit and 128 bit for 80 peers (95% CI).

Figure 3.18, 3.19, and 3.20 depict the time needed for agreeing on a reference playback timestamp for 40, 60, and 80 peers using a Bloom filter size of  $m = 128$  and  $m = 512$ , respectively. Letting *Merge and Forward* find an appropriate size for the Bloom filter increases the required time because each time a false positive is detected the algorithm increases the Bloom filter by a given amount of bits and reverts to a previous consistent state. We have increased the Bloom filter by 64 bits every time the algorithm detected a false positive. The figures show how much time increasing and restoring a consistent state costs in contrast to a case where only few or no false positives occur.

If we consider the results depicted by Figures 3.8, 3.9 and 3.10 it can be seen that *Aggregate* generates much more additional traffic than *Merge and Forward*. If we restrict the bandwidth of the peers to one or two Mbit/s *Aggregate* will not be able to maintain its optimality regarding the time required until all peers have agreed on a reference playback timestamp. For higher connectivities of the overlay network, *Merge and Forward* will outperform *Aggregate* even for the required synchronization time.

The evaluations on the required synchronization time of *Merge and Forward* depicts the worst case scenario, where many peers joins an IDMS session at once and then start to agree on a reference playback timestamp or if a re-synchronization has been triggered. For practical cases it is feasible to take into account the case where peers join one by one. Assume that a peers join an IDMS session where the other peers have already synchronized their multimedia playbacks. This new peer will receive the Bloom filter and the current average playback timestamp. Due to the definition of *Merge and Forward* (cf. Algorithm 3) this new peer will only add himself to the Bloom filter and will pass it on to its neighbors in the next period. Therefore, the required time until the new Bloom filter and its corresponding average playback timestamp propagates will be given by the diameter of the network graph. In this standard case *Merge and Forward* performs optimal (such as *Aggregate*) in terms of required time until all peers have agreed on the same reference playback timestamp.

### 3.5 Conclusion

We introduced IDMS for pull-based streaming by using a DCS to agree on a reference playback timestamp among the peers participating in a session. Furthermore, we introduced the notion of an IDMS session for pull-based streaming and showed how MPEG-DASH can be adopted to incorporate these IDMS sessions in the MPD. We introduced a DCS for negotiating on a reference playback timestamp among the peers in an IDMS Session. The following research objectives have been fulfilled by the introduced mechanism:

- (1) to define the notion of an IDMS session.
- (2) to introduce session management for IDMS in the context of MPEG-DASH.
- (3) to provide an algorithm that identifies the asynchronism between the multimedia playback of different users in an IDMS session in a distributed and self-organized manner.
- (4) to evaluate the introduced distributed and self-organized approach.



The first research objective **(1)** to define an IDMS Session is given in Definition 1. We further introduced session management for IDMS in the context of MPEG-DASH by extending the MPD with necessary information that allows a unique identification of IDMS sessions which corresponds to research objective **(2)**.

We introduced a DCS that outperforms algorithms from related work in an unicast and multicast scenario. The introduced DCS *Merge and Forward* was evaluated with respect to scalability and the time required until all peers in an IDMS session have agreed on a reference playback timestamp. The results show that *Merge and Forward* scales very well with the number of peers and the connectivity of the overlay network. Furthermore, the overhead saved allows peers to request higher quality streams which can improve the overall QoE of the IDMS system (research objectives **(3)** and **(4)**).

The selection of the average playback timestamp as the reference has the potential drawback that certain peers may be unable to synchronize to it due to a shortage of bandwidth. Therefore, we introduced a re-synchronization method that allows a peer to influence the calculation of the reference playback timestamp such that all peers are guaranteed able to synchronize their multimedia playback.

If we assume that multicast is in place the overhead produced by *Merge and Forward* decreases dramatically compared to *Aggregate* because *Merge and Forward* uses a constant (except if the false positive rate increases the size of the Bloom filter is increased) sized message structure in contrast to *Aggregate* which uses a list that carries all the necessary information of each peer.



---

# 4 Adaptive Media Playout for achieving IDMS

## 4.1 Introduction

In Chapter 3 we have provided a solution for the problem of determining a reference playback timestamp among a group of peers and, therefore, introduced a distributed algorithm. Once the peers have calculated the reference playback timestamp the question arises on how to overcome the identified asynchronism between the peer's current playback and the reference playback timestamp. A naïve approach would be to simply pause the playback or skip multimedia content in order to achieve the desired synchronization. Recent studies have shown that an increase in stalls (or pauses) of multimedia playback cause a decrease in the QoE [37] such as skipping multimedia content [22, 48]. Therefore, we propose to use AMP which adaptively increases or decreases the playback rate according to the identified asynchronism. Therefore, this chapter deals with the following mechanism of IDMS:

- **Carrying out the synchronization** to overcome the identified asynchronism by modifying the multimedia playback of each peer.

AMP has been extensively used in previous work in order to mitigate network effects on the multimedia playback such as network delay, jitter, or even packet loss. Therefore, AMP has been employed in order to maintain a constant playback buffer fill state trying to guarantee a smooth multimedia playback (cf. Section 2.3). We propose to use AMP for achieving synchronization by increasing or decreasing the multimedia playback rate with respect to the asynchronism. We focus on identifying the best *moment* for increasing or decreasing the playback rate such that the impact on the QoE is minimized. The work presented in this chapter is based on the following publications: [11], [13] and [14].

**Definition 4 (Content Section)** We define a content section as the multimedia content (represented by a collection of frames) starting at frame  $F_s$  and ending at frame  $F_e(t_{s_i}, t_{e_i}, \mu) = \lfloor F_s + (t_e - t_s) \cdot fps_{\mu_0} \cdot \mu \rfloor$ , where  $t_e - t_s$  denotes the duration of the content section with  $\forall t_s, t_e \in \mathbb{R}^+ : t_e \geq t_s$ ,  $fps_{\mu_0}$  denotes the (reference) frame rate of the audio or video domain and  $\mu$  denotes the desired (modified) playback rate.

In this chapter we report on research carried out to investigate AMP for IDMS as follows:

1. We will investigate whether selecting a content section (cf. Definition 4) using content features helps in decreasing the impact on the QoE if AMP is applied during the multimedia playback. This will be assessed by conducting a subjective quality assessment using crowdsourcing;
2. We will introduce measures that allow to quantify the distortion introduced by AMP. Using these measures we will analytically derive a utility model that provides insights on how the introduced distortion measures correlate with the QoE in the audio and video domain;
3. We will formulate and validate an optimization problem that allows to find those content sections that minimize the impact on the QoE given a starting playback timestamp, an asynchronism, and the maximum duration of the adjustment period or the maximum allowed playback rate.

## 4.2 AMP using Content Features vs. plain AMP

In order to initially investigate whether selecting content sections for increasing or decreasing the playback rate with respect to content features provides a higher QoE compared to randomly selecting a content section and then increasing or decreasing its playback rate, we conduct a subjective quality assessment using crowdsourcing. Therefore, we introduce the two following algorithms.

First, a naïve algorithm is introduced that randomly selects a content section with a given duration and then increases or decreases the playback rate for the selected

content section (cf. Section 4.2.1). Second, we introduce an algorithm that uses content features, e.g., in the audio domain we use the spectral energy and in the video domain we use the average length of motion vectors (henceforth *motion intensity*) of consecutive frames, respectively. The second algorithm is called QoE- and Context-aware Adaptive Media Playout (QoECAMP) (cf. Section 4.2.2) due to its content and context awareness. The content awareness is given by utilizing content features in order to find appropriate content section for increasing or decreasing the playback rate. Context awareness refers to the asynchronism itself stating whether to increase or decrease the playback rate. Previous work does only account for video and does not consider audio or the presence of both modalities (cf. Chapter 2.3). Therefore, the stimuli used for the subjective quality assessment contain both **audio and video**.

### 4.2.1 Random Selection of Content Sections

AMP algorithms discussed in [38, 44, 45] decrease or increase the playback rate according to the buffer fill state. The fill state of the buffer is influenced by an error prone channel which introduces transmission errors according to a series of random variables  $X_k$  that follow the Markov property. In order to mimic these AMP algorithms we determine the point in time for increasing or decreasing the playback rate randomly. This allows us to simulate transmission errors on which the above algorithms would react by increasing or decreasing the playback rate. The duration of these randomly determined content sections have the same duration as the content sections determined by our QoECAMP algorithm. This allows us to compare the random selection of content sections to our QoECAMP algorithm in terms of QoE.

### 4.2.2 QoE- and Context-aware Adaptive Media Playout

QoECAMP tries to postpone the increase or decrease of the playback rate until a *suitable content section* is identified that may reduce the impact of increasing or decreasing the playback rate on the QoE. Therefore, audio-visual features of the current buffer contents are used to determine these content section.

For the video feature we select the average length of motion vectors of each frame

$n \in \mathbb{N}$  depicted as  $f_v(n)$ . The set of motion vectors for frame  $n$  is denoted by  $V_n$ .  $|V_n|$  denotes the cardinality of  $V_n$  and  $N$  denotes the maximum number of frames. For each frame  $f_v(n)$  is calculated as defined in Equation 4.1.

$$f_v(n) = \frac{\sum_{i=1}^{|V_n|} \|\mathbf{v}_i\|_2}{|V_n|}, \quad (4.1)$$

where  $\mathbf{v}_i \in V_n$  and  $\|\mathbf{v}_i\|_2$  is the  $l^2$ -norm of vector  $\mathbf{v}_i \in \mathbb{R}^2$ .

For the audio feature we select the Root Mean Square (RMS) of the envelope of each audio frame  $n \in \mathbb{N}$  depicted as  $f_a(n)$  (cf. Equation 4.2). We use a resolution of signed 16-Bit for each audio sample ( $a_i$ ) and a sampling rate of 44.1 kHz. Furthermore, we use a hamming window of 1024 samples which corresponds to an audio frame and a half overlapping window, thus, resulting into 512 overlapped samples per window.

$$f_a(n) = \sqrt{\frac{\sum_{i=1}^{|A_n|} a_i^2}{|A_n|}}, \quad (4.2)$$

where  $a_i \in A_n$  and  $A_n$  denote the set of audio samples for an audio frame and  $|A_n|$  denotes the cardinality of  $A_n$ .

These features are measured over time and their average and standard deviation are used to approximate their future behavior. Therefore, the average of each feature  $f_i(n)$  within a time window is calculated (expressed as frames) by the use of a discrete moving average filter (or low-pass filter) with a windows size in frames given by  $\omega$  depicted by Equation 4.3.

$$M_{i\omega}(n) = \begin{cases} \frac{1}{\omega + 1} \sum_{j=n-\frac{\omega}{2}}^{n+\frac{\omega}{2}} f_i(j), & n \geq \frac{\omega}{2} \\ \frac{1}{n + 1} \sum_{j=0}^n f_i(j), & n < \frac{\omega}{2} \end{cases} \quad (4.3)$$

These averages are normalized by the current maximum of feature  $i$  which is denoted by  $\widehat{M}_{i\omega}(n) := \frac{M_{i\omega}(n)}{\max\{M_{i\omega}(n)\}}$ .  $\widetilde{M}_{i\omega}(n)$  denotes the average of the low-pass filtered feature  $f_i$  (cf. Equation 4.4) for a given window size  $\kappa$ . During media playback we do not know the overall maximum of a given feature because not all media units may

be present at the client. Thus, we cannot avoid using local maxima for normalizing  $M_{i_\omega}$ . Nevertheless, finding a new maximum does not invalidate previous decisions and calculations.

$$\widehat{M}_{i_{\kappa,\omega}}(n) = \begin{cases} \frac{1}{\kappa} \sum_{j=1}^{\kappa} \widehat{M}_{i_\omega}(n - \kappa + j), & \kappa \leq n \\ \frac{1}{n} \sum_{j=1}^n \widehat{M}_{i_\omega}(j), & \kappa > n \end{cases}, \quad (4.4)$$

$\kappa$  denotes the window size in frames that is used to take the *past* of  $\widehat{M}_{i_\omega}(n)$  starting at frame  $n$  into account. Past values of  $\widehat{M}_{i_\omega}$  reflect how feature  $f_i$  changed on average over the specified time window in frames. The current value of the features is compared to the mean minus the corresponding standard deviation for feature  $f_i$ . If the actual value of the features is below this threshold, the frame is selected for increasing or decreasing the playback rate.

For calculating the lower threshold  $l_i(n)$  and upper threshold  $u_i(n)$  for feature  $f_i$  we take the empirical standard deviation (which is an unbiased estimator for the variance)  $s_{i_{\kappa,\omega}}$  of the normalized  $M_{i_\omega}$  within a parametrized window  $\kappa$  depicted by Equation 4.5.

$$s_{i_{\kappa,\omega}}^2(n) = \begin{cases} \frac{\sum_{j=1}^{\kappa} (\widehat{M}_{i_\omega}(n - \kappa + j) - \widehat{M}_{i_{\kappa,\omega}}(n))^2}{\kappa - 1}, & \kappa \leq n \\ \frac{\sum_{j=1}^n (\widehat{M}_{i_\omega}(j) - \widehat{M}_{i_{\kappa,\omega}}(n))^2}{n - 1}, & \kappa > n \end{cases}, \quad (4.5)$$

The lower  $l_i(n)$  and upper  $u_i(n)$  thresholds are calculated as depicted by Equation 4.6 and Equation 4.7, respectively.

$$l_i(n) = \widehat{M}_{i_{\kappa,\omega}}(n) - \epsilon \cdot \sqrt{s_{i_{\kappa,\omega}}(n)^2} \quad (4.6)$$

$$u_i(n) = \widehat{M}_{i_{\kappa,\omega}}(n) + \epsilon \cdot \sqrt{s_{i_{\kappa,\omega}}(n)^2} \quad (4.7)$$

The lower and upper thresholds are learned from the past within the given window

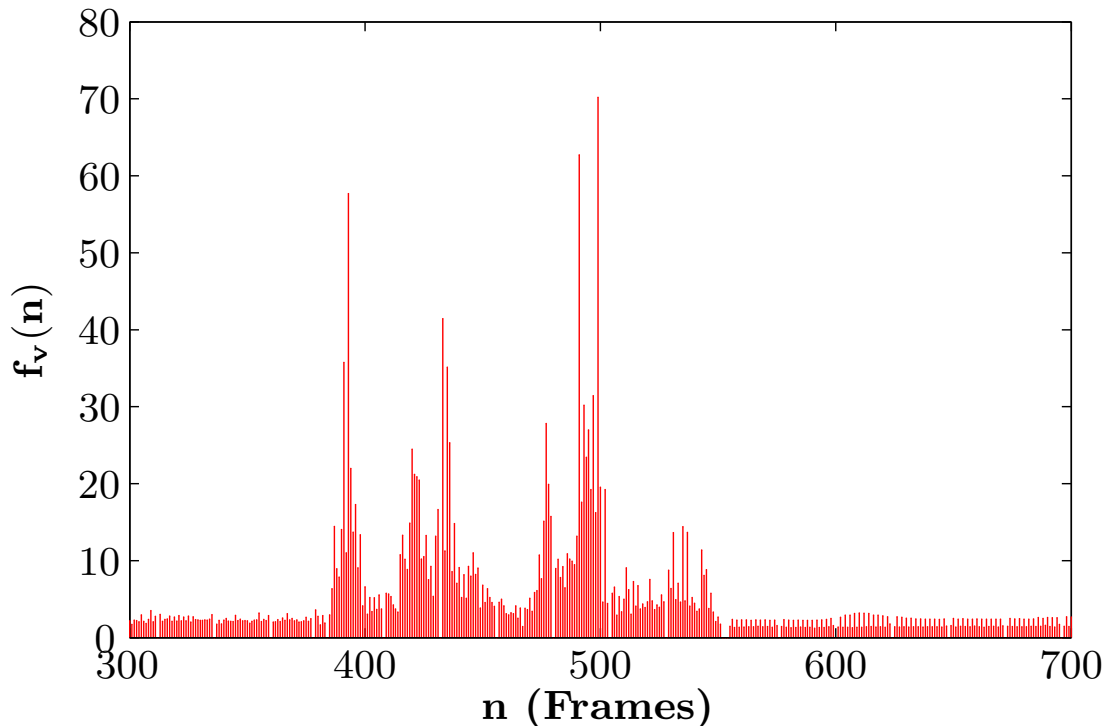


Figure 4.1: Average motion vectors, extracted from Big Buck Bunny encoded using AVC/H.264 and extracted using FFMPEG.

of  $\kappa$  frames. Increasing  $\kappa$  will increase the *learning period* and, thus, the thresholds will not *react* on immediate changes of  $M_{i_\omega}(n)$ . Decreasing  $\kappa$  will lead to a faster *reaction* because  $\lim_{\kappa \rightarrow 0} \widehat{M}_{i_{\kappa,\omega}}(n) = \widehat{M}_{i_\omega}(n)$  and, therefore,  $\lim_{\kappa \rightarrow 0} s_{i_{\kappa,\omega}}(n) = 0$ .  $\epsilon$  provides the possibility for fine-tuning and is used for constantly increasing or decreasing the standard deviation. This allows us to increase or decrease the distance from the upper and lower thresholds to  $\widehat{M}_{i_{\kappa,\omega}}$  and can be seen as sensitivity by increasing or decreasing  $\epsilon$ . For identifying the content sections for the stimuli of the subjective quality assessment we use following values for the parameters of the proposed algorithm:  $\omega = 100$ ,  $\kappa = 125$  and  $\epsilon = 1$ .

For example, Figure 4.1 illustrates the average motion vectors per frame for a sample video sequence taken from Big Buck Bunny (frames 300-700) [74]. These average motion vectors are used for calculating  $\widehat{M}_{v_\omega}(n)$  as depicted by Equation 4.3 and represent the average *motion* for a given video frame.

Figure 4.2 depicts  $\widehat{M}_{v_\omega}(n)$  for a window size of  $\omega = 100$  frames (or four seconds when using 25 fps). The frames used are taken from the average motion vectors



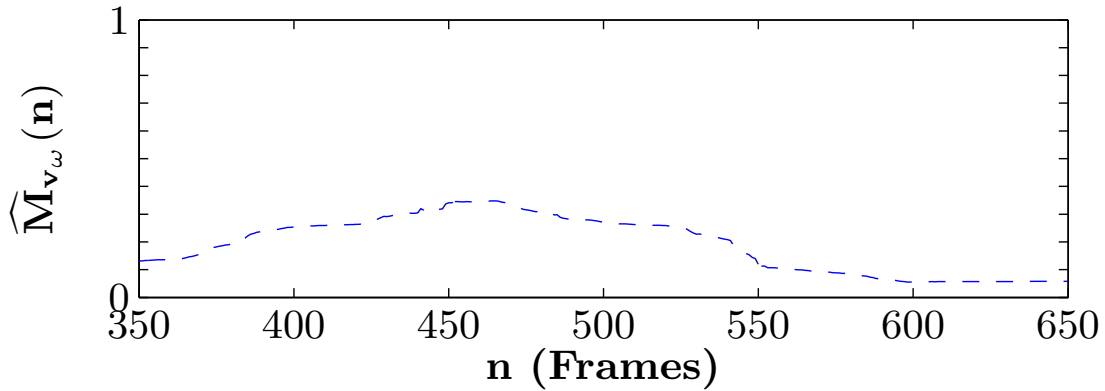


Figure 4.2:  $\widehat{M}_{v_\omega}(n)$  for a window of  $\omega = 100$  Frames.

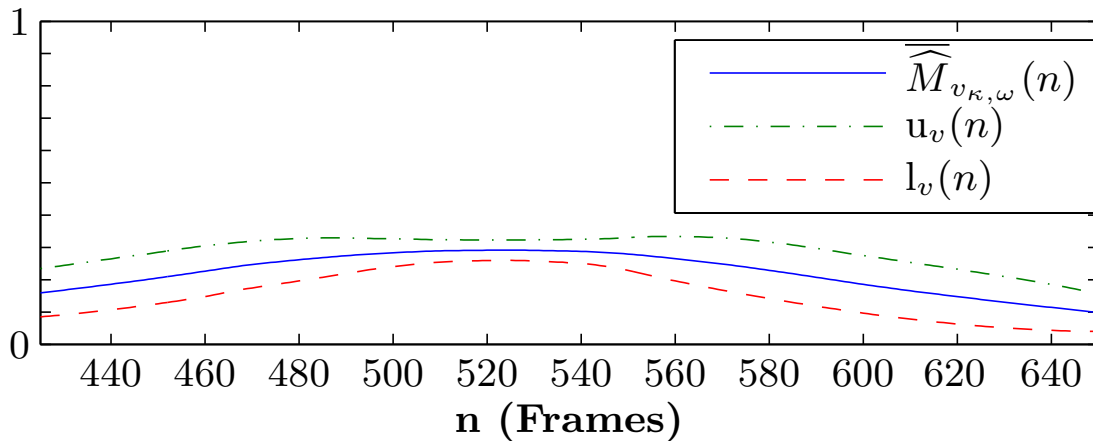


Figure 4.3:  $\widehat{M}_{v_{\kappa,\omega}}(n)$  and the lower and upper thresholds for a window of  $\kappa = 125$  frames.

depicted by Figure 4.1 starting at frame 300. Therefore,  $\widehat{M}_{v_\omega}(n)$  starts at frame 350 such that the first window comprises the frames [300,400] having its median at frame 350. Furthermore, Figure 4.2 depicts that  $\widehat{M}_{v_\omega}(n)$  is normalized according to the local maximum (cf. Figure 4.1).

Figure 4.3 illustrates  $\widehat{M}_{v_{\kappa,\omega}}(n)$  for the values of  $\widehat{M}_{v_\omega}(n)$  depicted in Figure 4.2. For calculating  $\widehat{M}_{v_{\kappa,\omega}}(n)$  we use a window of  $\kappa = 125$  and  $\omega = 100$  frames.  $\widehat{M}_{v_{\kappa,\omega}}(n)$  is based on the data shown in Figure 4.2 and, therefore, it starts at frame 425 as indicated by Equation 4.4 because we have set a window size of 125 frames. Furthermore, the upper threshold ( $u_v(n)$ ) and lower threshold ( $l_v(n)$ ) are shown in Figure 4.3 and how they converge to  $\widehat{M}_{v_\omega}(n)$  when there is no change in the low pass filtered average motion vectors given by  $\widehat{M}_{v_\omega}(n)$ . A rapid change in motion will cause  $\widehat{M}_{v_\omega}(n)$

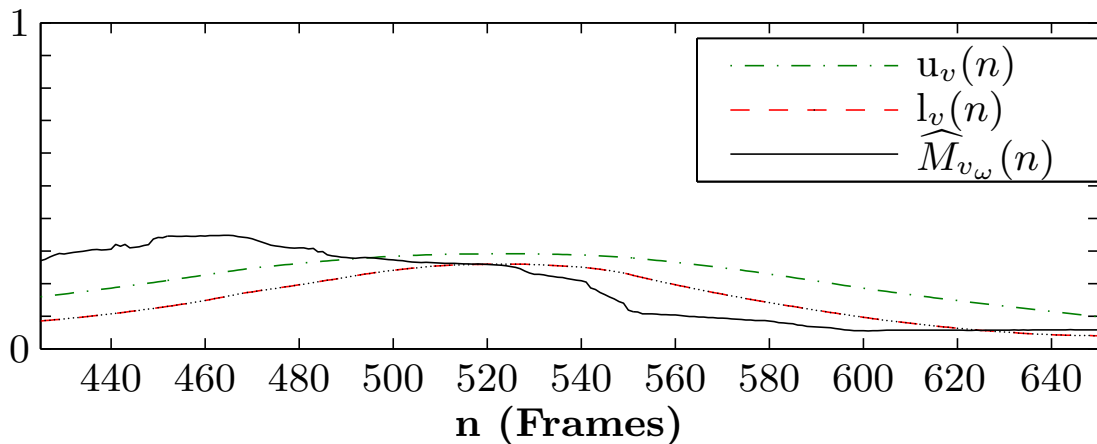


Figure 4.4:  $\widehat{M}_{v_\omega}(n)$  exceeding the lower and upper thresholds.

to change and may exceed the upper or lower thresholds.

Finally, Figure 4.4 depicts  $\widehat{M}_{v_\omega}(n)$  exceeding the lower and upper thresholds. Furthermore, it shows how  $\widehat{M}_{v_\omega}(n)$  exceeds  $u_v$  when the average motion per frame increases (cf. Figure 4.1) and under runs  $l_v$  when the short increase in motion levels out (cf. Figure 4.1).

The decision about the actual playback rate (i.e., whether to increase or decrease) is provided in Algorithm 4.

$N$  depicts the length of the video in frames.  $N_{b_{new}}$  denotes the buffer fill state in frames with respect to  $N_{b_{old}}$  which denotes the previous buffer fill state for which the playback rate adjustment has been calculated.  $TH_b$  denotes the threshold for triggering the update of the playback rate. The recalculation periods can be adjusted by modifying  $TH_b$ .  $\xi$  denotes the asynchronism (in seconds) of the media playback for a given client which is typically provided by the IDMS system whose responsibility is signaling of timing information and session management. If the asynchronism exceeds a certain threshold ( $TH_d$ ) the playback rate  $\mu$  for frame  $n$  is adjusted by using a step-wise, linear, or any other function denoted by  $adjust(n, \xi)$ . Furthermore,  $adjust(n, \xi)$  returns the remaining asynchronism if the playback rate is increased or decreased. We select a step-wise adjustment function. Investigating the impact of different adjustment functions is out of scope of this work and is devoted to future work.

**Algorithm 4** QoECAMP.

---

```

1: function  $QoECAMP(TH_b, TH_d, \xi)$ 
2:   if  $N_{b_{new}} - N_{b_{old}} > TH_b$  then
3:     for  $n \leq (N_{b_{new}} - N_{b_{old}})$  do
4:        $cp(n) = update(n, N_{b_{old}}, N_{b_{new}})$ 
5:       if  $|\xi| > TH_d \wedge cp(n) == true$  then
6:          $\mu(n), \xi = adjust(n, \xi)$ 
7:       end if
8:     end for
9:      $N_{b_{old}} = N_{b_{new}}$ 
10:  end if
11: end function
12: function  $update(n, N_{b_{old}}, N_{b_{new}})$ 
13:  if  $(l_v(n) > \widehat{M}_{v_\omega}(n) \vee u_v(n) < \widehat{M}_{v_\omega}(n)) \wedge l_a(n) > \widehat{M}_{a_\omega}(n)$  then
14:    return true
15:  else
16:    return false
17:  end if
18: end function

```

---

Updating the playback rate of frame  $n$ , is done according to the upper and lower bounds for the audio and video features and if and only if  $\widehat{M}_{i_\omega}(n)$  exceeds these thresholds. For motion vectors both thresholds are checked because we assume that rapid changes in the motion of video content allows for increasing or decreasing the playback rate according to our hypothesis introduced at the beginning of this section. Thus, the logical statement " $(l_v(n) > \widehat{M}_{v_\omega}(n) \vee u_v(n) < \widehat{M}_{v_\omega}(n)) \wedge l_a(n) > \widehat{M}_{a_\omega}(n)$ " represents our hypothesis. For audio we only test the lower thresholds because our assumption is that the playback rate can only be increased or decreased when the audio volume is low. The algorithm is designed to be lightweight such that it can be used during media playback.

### 4.2.3 Stimuli and Stimulus Presentation

For our subjective quality assessment using crowdsourcing we select excerpts from the Big Buck Bunny and Sintel (the video sequences are available under the creative commons license, thus, they can be be freely used) sequences with the absolute start

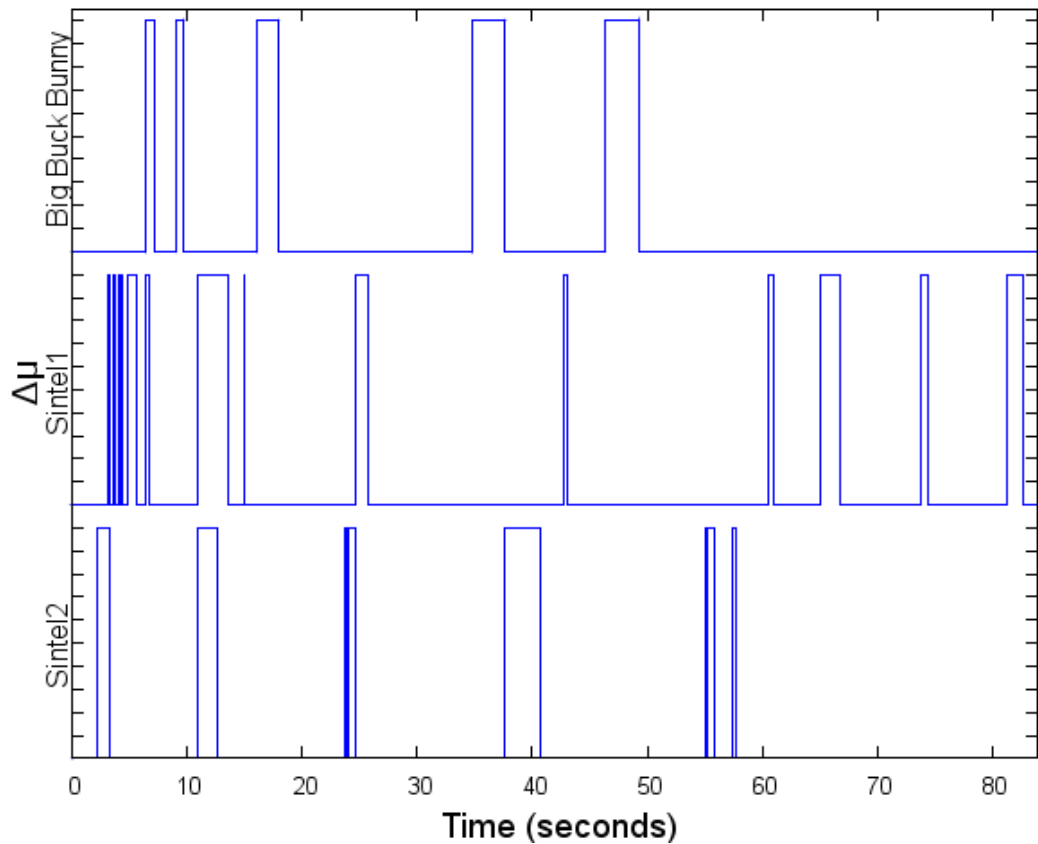


Figure 4.5: Content sections identified by QoECAMP for the selected video sequences.

time and end time of the actual sequence given in brackets (mm:ss) followed by the total length:

- Big Buck Bunny (01:10-02:00, 50s);
- Sintel1 (01:30-02:54, 84s);
- Sintel2 (02:54-03:52, 58s).

Please note that Big Buck Bunny was only presented during the training phase of the experiment. All sequences have a resolution of 720p, 25 fps, and a bit-rate of about 2.5 Mbit/s. We select two excerpts of the Sintel sequence such that the first (Sintel1) contains a fair amount of natural speech (i.e., dialogs) and the second (Sintel2) contains nearly no natural speech.

Figure 4.5 depicts the content sections identified by our QoECAMP algorithm for the selected video sequences. The x-axis depicts the length of the video sequence in

seconds and the y-axis denotes whether the playback rate can be adjusted ( $\Delta\mu = 1$ ) according to QoECAMP. For Sintel1 the total duration of the playback rate adjustments is 10.08 seconds which is 12% of the total length. The playback rate adjustments for Sintel2 have a length of 7.84 seconds representing 13.52% of the total length. For the training sequence Big Buck Bunny, the playback rate adjustments have a total length of 8.84 seconds that is 17.68% of the total length. The playback rate adjustments for the training phase should provide the possibility to become familiar with this type of temporal impairments. The content sections which have been randomly selected had the same length as the content sections selected by our QoECAMP algorithm.

For the playback rate adjustments we choose the following values for  $\mu$ : 0.5, 0.75, 1.5, and 2 times the nominal playback rate, i.e.,  $\mu = 1$ . We selected these playback rates in order to assess whether a change of 25% of the nominal playback has no significant impact on the QoE as stated in [38], [44] and [75]. Furthermore, we are interested in how the QoE is influenced when the playback rate is even higher or lower than the claimed 25%. We present each video with each of the algorithms (Random and QoECAMP) where the content sections selected by the algorithm are played with each of the given playback rates. Therefore, each algorithm is presented nine times including the reference for  $\mu = 1$  for each sequence. Thus, in total we have 18 test conditions.

#### 4.2.4 Crowdsourcing Platform and Participants

As already mentioned we use Microworkers [60] as crowdsourcing platform as it allows hiring workers outside of the USA which Amazon's Mechanical Turk does not allow [59]. As already outlined in the related work section (cf. Section 2.5) crowdsourcing provides several advantages. For instance, the instant access to a huge number of participants and the low effort and price of actually conducting the experiment. But, it is not the holy grail in the field of subjective quality assessments. It comes with a lot of disadvantages. For instance, the need for a careful and well-thought-out test methodology and design as there is no possibility to supervise the participants (in contrast to in-lab studies) [61]. Thus, a training session and a very detailed introduction

are mandatory when conducting subjective quality assessments using crowdsourcing. Even then, additional methods are needed to allow an in-depth analysis of the collected measures [61].

Figure 4.6: Creating a campaign using Microworkers.

Figure 4.6 depicts how a *campaign* or in our case a subjective quality assessment is created using Microworkers. Microworkers allows to create a campaign for different regions. Note that, only if we select *International* we have to explicitly exclude countries. If we select a specific region, Microworkers will ask to explicitly include a certain number of countries. The total costs of the campaign results from  $time \cdot \#participants \cdot 1.0839$  (Microworkers takes approximately 8.4% of the total amount

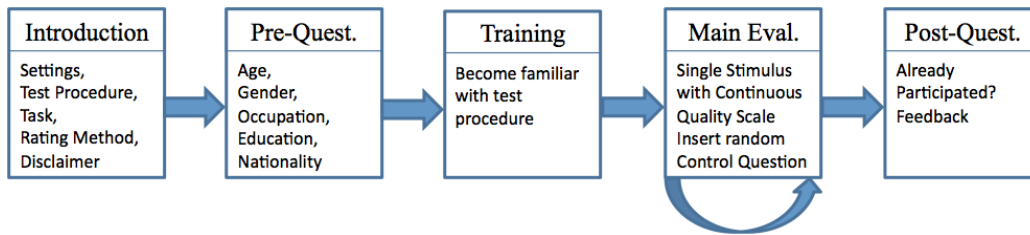


Figure 4.7: Evaluation methodology.

as a fee). We further have to provide a short description of the task that has to be fulfilled by the participants in order to get paid (cf. Figure 4.6 *What is needed to be done*). In order to validate that a micro-worker has fulfilled the task, we are allowed to ask for a dedicated proof. In our case the subjective quality assessment platform provides the participants with a unique key. This unique key should be provided by every participant in order to get the promised money (cf. Figure 4.6 *Required proof of job finished*). We found that the compensation for a task which requires approximately 20 minutes is on average about 0.7 cent (Euro) at the Microworkers platform.

### 4.2.5 Evaluation Methodology

For conducting the experiment we hired workers from Europe and the USA which results in a higher reliability of the responses [76]. The subjective quality assessment is structured as depicted in Figure 4.7.

**Introduction.** At the beginning, a short introduction is presented to the participants which explains in detail what the participants have to do and what they will have to assess. In particular, the participants' task is to evaluate the perceived quality of the viewing/hearing experience while watching the sequences. Furthermore, we explain the whole assessment such that no questions are left open. This includes a detailed explanation of what will happen during the experiment, how the rating scale and rating possibility will look like, and the different phases of the experiment. Additionally, we ask the participants to turn off mobile devices, darken the room, and set up their audio devices to a pleasant configuration. Furthermore, the introduction includes a disclaimer that persons who are visually impaired or have impairments

regarding hearing should not take part in the subjective quality assessment. Chapter 5.5.3 provides the introduction and the disclaimer.

**Pre-Questionnaire.** After the introduction, a pre-questionnaire is shown to gather demographical information about the participants, i.e., age, gender, country of residence, nationality, occupational field, and education. This will provide us with demographical data that can be used to identify influence factors for groups of participants clustered according to one of the demographic variables. Chapter 5.5.3 provides the full pre-questionnaire.

**Training.** The training phase using the Big Buck Bunny sequence is presented to allow the participants to adjust their audio volume and to become familiar with the stimulus presentation. Furthermore, it allows participants to become familiar with the rating scale. The Big Buck Bunny sequence is presented in three different configurations. The first configuration comprises the training sequence with the nominal playback rate of  $\mu = 1$  and, thus, without any temporal impairments. For the other two configurations we modified the playback rate to  $\mu = 2$  (i.e., twice the nominal playback rate) and  $\mu = 0.5$  (i.e., half the nominal playback rate) for selected content sections. We selected the Big Buck Bunny sequence as training sequence because it does not convey any natural speech because playback rate deviations are more easily perceived if natural speech is affected [77].

**Main Evaluation.** The main evaluation adopts a single stimulus with hidden reference as recommended by the ITU [56, 78]. The idea behind the selection of a single stimulus with hidden reference was that the participants should not know the reference condition (i.e.,  $\mu = 1$ ). They should only (absolutely) rate the actual sequence with or without temporal impairments like in a home TV viewing/hearing experience. The hidden reference should allow us to clarify whether there is a significant difference between the reference and the temporal impaired sequences. After each test condition the rating possibility is presented to the participants using a slider depicted by Figure 5.9 (cf. Chapter 5.5.3). We selected a continuous rating scale with an interval of  $[0, 100]$  with 0 indicating a very low QoE and 100 representing a very high QoE. Furthermore, each rating phase was limited to eight seconds. Additionally, we use a control question (i.e., "What was present in the last video sequence?") with



three possible answers using an option box to check whether the participants are paying attention. The sequence in which the answer possibilities are presented is chosen randomly. This shall even more reduce the probability that a participant selects the correct answer by random. The control question is inserted randomly following one of the 18 test conditions.

**Post-Questionnaire.** Finally, at the end of the experiment the participants are asked to fill out a post-questionnaire. The post-questionnaire provides participants the opportunity to give feedback using a free text field regarding whether they participated already in a similar experiment. After the post-questionnaire a unique token is shown which is a mandatory proof that a micro-worker had successfully participated in our subjective quality assessment.

#### 4.2.6 Filtering of Participants

We introduce a three-level scheme for filtering participants from the result set. We do not solely rely on the ratings obtained by the rating possibility. Therefore, we use the additional data that is gathered by our Web-based assessment platform (cf. Chapter 5.2.5). That is the duration of each stimulus presentation and the duration of each rating process for each participant. We first describe the methods and afterwards we give the numbers of participants that have been screened by using the following methods.

The **first level** comprises the control question and we reject participants who did not provide a correct answer to the control question. A wrong answer to the control question may indicate that the participant did not pay attention to the presented sequences. Furthermore, it may indicate that a participant did not understand the question and therefore it is likely that the participant may not have understood the introduction and, consequently, the actual task. Thus, we reject participants that provided a wrong answer to the control question.

The **second level** is about screening participants who had a significant difference in playback time in comparison to the nominal playback time for each stimulus presentation. Therefore, we use the F-test to test whether there exists a significant

difference between the variances of the nominal playback times and the playback times of each participant. This allows us to overcome some minor deviation from the nominal playback time of the stimulus presentations. Let

$$f = \frac{(n - 1) \sum_{i=0}^m (x_i - \bar{X})^2}{(m - 1) \sum_{j=0}^n (y_j - \bar{Y})^2}$$

be the ratio of the sample variance of the actual playback duration ( $s_X^2$ ) and the sample variance of the playback duration with the nominal playback rate ( $s_Y^2$ ). Thus, representing our test statistic with the hypothesis that the variances are equal ( $H_0$ ) which follows an F-distribution with  $n - 1$  and  $m - 1$  degrees of freedom. We reject  $H_0$  for values greater than  $F_{m-1, n-1}$  for a significance level of  $\alpha = 0.05$ . Thus, values within the 0.95 quartile are accepted. The accuracy of this method can be fine-tuned by selecting a higher or lower quartile.

This method reveals those participants that paused the playback or tried to skip the presentation of the stimuli. For our purpose a correct playback of the test conditions was crucial for a successful subjective quality assessment. A non-continuous or jerky playback of the test conditions (e.g., due to the participants' personal computer) may be perceived as temporal playback impairments like the ones we introduced artificially. This leads to the conclusion that participants with a non-continuous playback cannot provide viable ratings. Using this filtering scheme we reject the  $H_0$  hypothesis which states that  $\frac{s_X^2}{s_Y^2} = 1$  for participants who exceeded the significance level  $\alpha$ .

The **third level** filters those participants with *abnormal* rating behavior. Therefore, we take a closer look at the ratings of each participant and found that some participants moved the slider only a few times out of  $n$  rating possibilities (in our case  $n = 18$ ). In particular, some participants rated only a few times (i.e., three times or less) and then just let the user study pass through without moving the slider anymore (i.e., keeping it in the initial position). Interestingly, the few ratings are distributed across the entire study and not clustered, e.g., at the beginning. Please note that the actual rating time does not necessarily provide the evidence of cheating as some participants perform the rating and wait until the next sequence is shown instead of manually moving forward before the voting time expires.

Another observation is that some participants rated the minimum and/or maximum values very often. In practice, however, it was observed that only a very small fraction of participants select the extreme values provided by the rating scale [79]. In order to detect whether a participant just leaves the slider at the initial position or moves it to one of the extreme values (0 or 100), we count the cases where the slider is positioned at these values for each stimulus presentation. We assume that the probability of selecting an extreme value or not is  $p = \frac{1}{2}$ . Therefore, we model the selection of an extreme value by the use of a binomial distribution ( $X \sim B(n, p)$ ).

Let

$$f(k; n, p) = \binom{n}{k} p^k (1-p)^{n-k}$$

be the density function of the binomial distribution and

$$F(k; n, p) = \mathbb{P}(X \leq k) = \sum_{i=0}^k \binom{n}{i} p^i (1-p)^{n-i}$$

be the cumulative distribution function of the binomial distribution. We reject our null hypothesis if

$$\alpha \geq 1 - \mathbb{P}(X \leq k)$$

is less than or equal to the significance level  $\alpha$ . For our purpose we set  $\alpha = 0.05$ .

In total 119 micro-worker participated in the subjective quality assessment. The filtering using the control question revealed that four participants did not provide a correct answer. The screening according to the playback duration resulted in nine participants. Four of these nine participants did try to skip at least one stimulus presentation. Five out of the nine participants paused the playback of at least one stimulus presentation. For the ratings we rejected ten participants that did not provide viable ratings, i.e., either not moving the slider at all or selecting an extreme value very often. In total we screened 23 participants out of 119 by applying our filtering scheme. We further used the Median Absolute Deviation (MAD) to detect outliers according to the threshold of two times the standard deviation for the QoE ratings of each stimulus configuration [80]. Further statistical analysis is based on the filtered responses we received from 96 participants which is presented in the following section.

### 4.2.7 Statistical Analysis of the Responses

For the pre-questionnaire the participants provided us with the following feedback. The majority of the participants is between 20 and 25 years old with 85% of the participants 35 or younger and from the 96 participants are 20 female and 76 male. 11% stated that they are experts and work in the field of computer and mathematics. Furthermore, 26% of the participants stated that they are students.

For the post-questionnaire we got the following feedback. Approximately 80% of the participants stated that they have not participated in a similar experiment. The remaining 20% stated that they already participated in a similar subjective quality assessment hosted at Microworkers. Furthermore, approximately 4% stated that the subjective quality assessment was too long or that the number of sequences should be reduced.

Our focus is on identifying whether there exist significant differences between the two algorithms for the selected playback rates. We analyzed the responses according to significant differences between their means by using a Student's t-test. Prior to the Student's t-test we ensured that the variance between two tested samples are equal by conducting an F-test. According to the Central Limit Theorem [81] we assume that the ratings of the participants are normally distributed. Nevertheless, for testing whether there ratings are normally distributed we employed the Lilliefors-test [82] and the Shapiro-Wilk-test [83], which did not reject the null hypothesis ( $H_0$ ), stating that a normal distribution is present. If the analysis of the variances rejects the hypothesis that the variances of two cases have equal variances, we use the Welch's t-test instead of the Student's t-test which assumes a normal distribution of the samples.

Figure 4.8 depicts the Mean Opinion Score (MOS) and the 95% Confidence Interval (CI) for the first sequence Sintel1 for each test condition with the QoECAMP and Random algorithm. As already mentioned, Sintel1 contains a fair amount of natural speech (dialogs) with high audio volume. The x-axis shows the different playback rate adjustments and the hidden reference is depicted by  $\mu = 1$ , i.e., participants voted only once and, thus, the MOS for both QoECAMP and Random are equal.

At a first glance it can be observed that for playback rates close to the reference

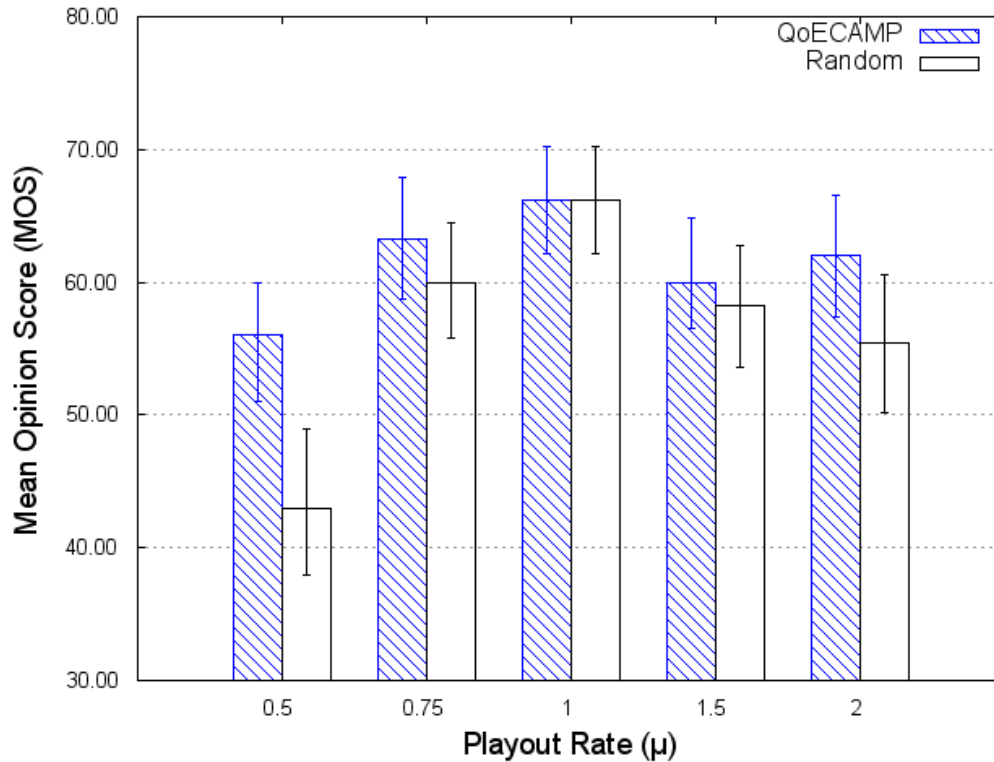


Figure 4.8: MOS and 95% CI for the Sintel1 sequence.

$\mu = 1$  the MOS does not change that much for both algorithms. This finding is supported by the results of a Student's t-test between the means of QoECAMP for  $\mu = 0.75$  and  $\mu = 1.5$  and the reference condition  $\mu = 1$ . For Random there is in fact a significant difference between the means for  $\mu = 0.75$  and  $\mu = 1.5$  and the reference with:  $\mu = 0.75$ ,  $p = 0.048$  and  $t = 1.99$ ;  $\mu = 1.5$ ,  $p = 0.01$  and  $t = 2.5387$ . Taking a look at the test conditions where the playback rate was decreased to  $\mu = 0.5$  and increased to  $\mu = 2$  it can be observed that QoECAMP starts to outperform the Random algorithm. For  $\mu = 0.5$  the difference of the means of both algorithms compared to the reference is statistically significant ( $p = 4.3 \cdot 10^{-10}$ ,  $t = 6.587$  for Random and  $p = 0.0011$  and  $t = 3.321$  for QoECAMP). According to a Student's t-test the difference of the means between both algorithms for  $\mu = 0.5$  is statistically significant too ( $p = 0.0007$  and  $t = 3.4$ ). These results state that QoECAMP performs significantly better in extreme situations where the playback rate is very low or very high.

For  $\mu = 2$  the same behavior can be observed. The QoECAMP algorithm is able

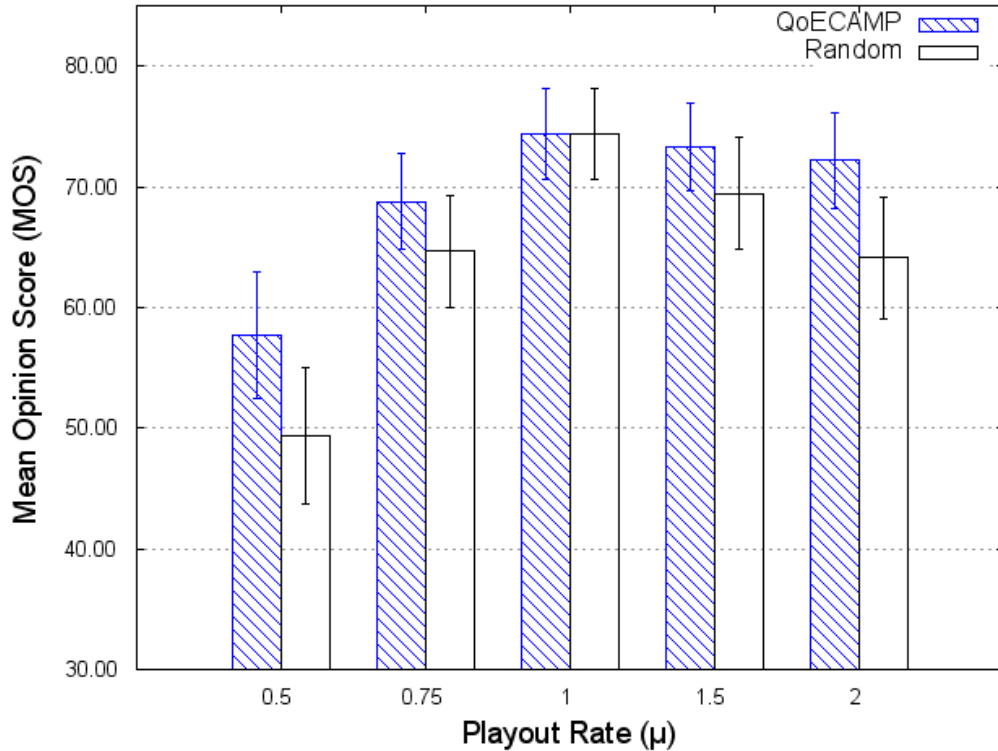


Figure 4.9: MOS and 95% CI for the Sintel2 sequence.

to maintain a QoE of above 70 MOS points. The Random algorithm scores below 65 MOS points. A Student's t-test revealed a significant difference for the mean of Random and the reference ( $p = 0.0016$ ,  $t = 3.198$ ). There is no significant difference between QoECAMP and the reference for  $\mu = 2$ . Another interesting finding is that decreasing the playback rate results into higher QoE degradations than increasing the playback rate by the reciprocal factor of the decrease.

Figure 4.9 illustrates the MOS and the 95% CI for the second sequence Sintel2 comprising nearly no natural speech and low audio volume. Interestingly, when increasing the playback rate ( $\mu > 1$ ), the MOS remains almost the same for QoECAMP while it decreases for the Random case. Furthermore, the QoECAMP algorithm scores a higher MOS than the Random algorithm for all playback rate adjustments ( $\mu \neq 0$ ). Again, it can be observed that playback rates close to the reference of  $\mu = 1$  do not show a statistically significant difference, except for Random with  $\mu = 0.75$  ( $p = 0.0017$ ,  $t = 3.18$ ). For  $\mu = 0.5$  and  $\mu = 2$ , Figure 4.9 show the same tendency as the MOS for  $\mu = 2$  in Figure 4.8. For the playback rate  $\mu = 0.5$  the means of

both algorithms are statistically significant different in comparison to the reference ( $p = 8.9 \cdot 10^{-7}$ ,  $t = 5.1$  for QoECAMP;  $p = 1.1 \cdot 10^{-11}$ ,  $t = 7.24$  for Random). For  $\mu = 0.5$  there is a significant difference of the means of both algorithms with  $p = 0.03373$  and  $t = 2.1388$ . The results state that QoECAMP performs better than Random from a QoE point of view.

If we increase the playback rate the same behavior can be observed as with  $\mu = 0.5$ . QoECAMP is able to maintain a high QoE in comparison to Random. A Student's t-test supports this finding by stating a significant difference between the means of Random for  $\mu = 2$  and the reference ( $p = 0.0016$ ,  $t = 3.198$ ) but not for QoECAMP. Furthermore, a Student's t-test revealed a statistically significant difference for both means of both algorithms for  $\mu = 2$  ( $p = 0.01476$ , and  $t = 2$ ). As before, we observe that decreasing the playback rate leads to a higher decrease in QoE than increasing the playback rate.

#### 4.2.8 Discussion on the results

The results presented in Section 4.2.7 clearly show that the impact of increasing the playback rate on the QoE is lower than the impact of decreasing the playback rate. On the one hand, the findings contradict the results of informal tests mentioned in [38], [44] and [75], where it is stated that playback variations of 25% up to 50% of the nominal playback rate may not be perceptible by users. The results state that this does not hold, especially, if the decision of increasing or decreasing the playback rate is based on a random variable. The results of the study show that a more sophisticated selection of content sections for increasing or decreasing the playback rate allows decreasing or increasing the playback rate without significantly degrading the QoE. Furthermore, the results state that the selection of the content sections gets even more important the higher or lower the playback rate is. This is a very important finding regarding the synchronization of the media playback between clients in IDMS because it states that with the selection of an appropriate content section increasing the playback rate by 25% does not have a significant impact on the QoE. This allows us to overcome asynchronism assuming that the buffer fill state is high enough.

Interestingly, increasing the playback rate does not have the same impact on the

QoE than decreasing the playback rate. Especially, for playback rates of about 25% of the nominal playback rate it seems that the selection of content sections does not matter when increasing the playback rate. Thus, increasing the playback rate should always be preferred if possible. Another very important fact that can be observed when comparing Figure 4.8 and Figure 4.9 is that with more information present in the audio domain the impact on the QoE increases. This provides us another hint. Namely, that audio plays a very important role when selecting the content sections for playback rate variations. These findings will be investigated in the following section.

### 4.3 Quantification of the Distortion caused by AMP

In this section we pick up the findings discussed in Section 4.2.8. Namely, to investigate the correlation between the distortion in the audio and video domain and the impact on the QoE when AMP is employed. Therefore, we define measures (semi-metrics) that allow to measure the distortion introduced by AMP in the temporal domain.

We further conduct a subjective quality assessment where we increase and decrease the playback rate of randomly selected content sections. We further try to find a model that explains the correlation between the average distortion in the audio and video domain for each playback rate configuration and the resulting QoE. We validate our utility model using the data of the subjective quality assessment and using the data of the subjective quality assessment described in Section 4.2.

#### 4.3.1 Quantifying the Distortion of Multimedia Content in the Temporal Domain

In the past decade many spatial quality *metrics* have been introduced, especially for the video domain [84]. It has been shown that the spatial quality *metrics* are able to represent the impact on the QoE to a certain extent. Recent quality *metrics* aim to cover the temporal domain. For example, in [85] a spatial-temporal quality metric for video has been introduced and subjectively evaluated. This metric relies on the



image data of each video frame and tries to quantify its spatial-temporal distortion. In our case we aim on quantifying the distortion caused by playback rate variations and, thus, we focus on temporal information only.

Increasing or decreasing the media playback rate – denoted as  $\mu$  – results in a perceptual distortion in audio and/or video as we have already shown in Section 4.2. This distortion depends on the actual multimedia content, especially on the temporal metrics, for which the playback rate is increased or decreased. Therefore, we propose the following measures for quantifying the distortion caused by modifying the playback rate of audio and video:

- **Audio:** the spectral energy of an audio frame for the  $c$ -th channel is denoted by  $f_{a_c}(x)$ .
- **Video:** the average length of motion vectors between two consecutive frames denoted by  $f_v(x)$ .

These measures allow us to quantify the distortion for audio and video when increasing or decreasing the playback rate for a specific content section. This is done by comparing how much of each temporal metric has been experienced by the user during the content section with and without the playback rate change. We differentiate between increasing and decreasing the playback rate when calculating our distortion metrics because our hypotheses is that increasing the playback rate *may* have a different impact on the QoE than decreasing the playback rate.

In order to calculate our metrics we calculate the last and the first frame number for the content section for which the playback rate shall be changed. Determining the first frame for the  $i$ -th content section for which the playback rate is changed is done by  $F_s(t_{s_i}) = \lfloor t_{s_i} \cdot fps_{\mu_0} \rfloor$ , where  $t_{s_i}, t_{e_i}$  denotes the duration of the  $i$ -th content section in seconds.  $fps_{\mu_0}$  represents the frames per second for the nominal playback rate  $\mu_0 = 1$ . For determining the last frame of the  $i$ -th content section for a specific playback rate  $\mu$  we use Definition 4 as depicted by Equation 4.8.

$$F_e(t_{s_i}, t_{e_i}, \mu) = \lfloor F_s(t_{s_i}) + (t_{e_i} - t_{s_i}) \cdot fps_{\mu_0} \cdot \mu \rfloor \quad (4.8)$$

In the following, we introduce the measures for the distortion in audio ( $d_{a_i}(\mu_1, \mu_2)$ ) and the distortion in video ( $d_{v_i}(\mu_1, \mu_2)$ ) for the  $i$ -th content section.

$$d_{v_i}(\mu_1, \mu_2) = \frac{1}{\sum_{j=1}^{|F|} f_v(j)} \left( \sum_{j=F_s(t_{s_i})}^{F_e(t_{s_i}, t_{e_i}, \mu_1)} f_v(j) - \sum_{j=F_s(t_{s_i})}^{F_e(t_{s_i}, t_{e_i}, \mu_2)} f_v(j) \right) \quad (4.9)$$

Equation 4.19 denotes the distortion measure for video.  $d_{v_i}(\mu_1, \mu_2)$  may be any value in the interval of  $[-1, 1]$ .  $|F|$  depicts the overall number of frames. For audio we followed the same principle as for video with the difference that we used the spectral energy of the audio frames for each audio channel by using the Fourier Transformation. The Fourier Transformation for a single audio frame and for the  $i$ -th content section is denoted by  $\hat{a}_{c_i}$  (cf. Equation 4.10), where  $c$  depicts the audio channel. Equation 4.11 defines our distortion measure for audio. Note that we take into account the audio channels.

$$\hat{a}_{c_i}(k) = \sum_{j=N_i}^{M_i} e^{-2\pi i \frac{jk}{M}} f_{a_c}(j) \quad (4.10)$$

$$s_i(\mu_1, \mu_2) = \sum_{c=1}^C \left( \sum_{u=F_s(t_{s_i})}^{F_e(t_{s_i}, t_{e_i}, \mu_1)} \sum_{k=0}^{S_f} |\hat{a}_{c_u}(k)| - \sum_{u=F_s(t_{s_i})}^{F_e(t_{s_i}, t_{e_i}, \mu_2)} \sum_{k=0}^{S_f} |\hat{a}_{c_u}(k)| \right), \quad (4.11)$$

$C$  denotes the number of audio channels available.  $S_f$  denotes the highest frequency which is bounded by half the sampling frequency (according to Shanon-Nyquist-Theorem). Finally,  $d_{a_i}$  (cf. Equation 4.12) denotes the distortion in audio for the  $i$ -th content section.

$$d_{a_i}(\mu_1, \mu_2) = \frac{1}{\sum_{c=1}^C se_c} s_i(\mu_1, \mu_2), \quad (4.12)$$

$se_c$  denotes the overall spectral energy of channel  $c$ . Again,  $d_{a_i}(\mu_1, \mu_2)$  may be any value in the interval of  $[-1, 1]$ . If there are more content sections with the same change in the playback rate we use the average of the introduced measures denoted by  $\bar{d}_v$  and  $\bar{d}_a$ , respectively. Please note, if we use the absolute values of the distortion measures  $d_{a_i}(\mu_1, \mu_2)$  and  $d_{v_i}(\mu_1, \mu_2)$  fulfill the axioms for a semi-metric with  $\mu_1, \mu_2 \geq 0$  which can be easily proven (in a semi-metric space points are not distinguishable  $d(x, y) = 0$  even if  $x \neq y$  which fits the underlying assumptions of features) .

In order to derive a utility model which correlates the introduced measures and the

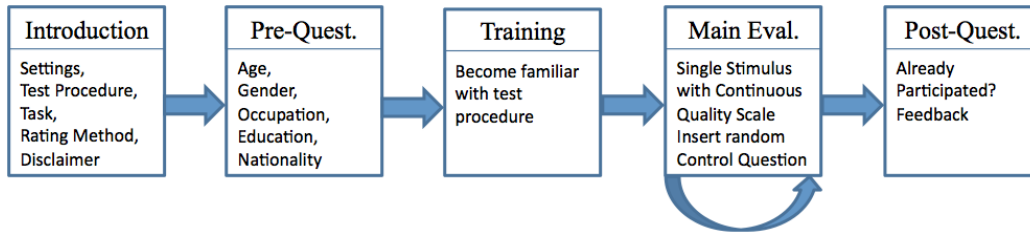


Figure 4.10: Methodology for the Study.

QoE, we conduct a subjective quality assessment using crowdsourcing. The methodology, results of the study and the derived utility model are discussed in the following sections.

### 4.3.2 Participants, Stimuli, Methodology, and Assessment Platform

For conducting our user study we selected the crowdsourcing platform Microworkers [60] (as in the previous SQA). For a more detailed explanation of Microworkers the interested reader is referred to Section 4.2.4. The duration of the study is approximately 15 minutes. We have found that the typical amount of money that is payed for a task with a duration of about 15 minutes is approximately \$0.20. Therefore, we have set a slightly higher compensation of \$0.25 as an extra motivation for each participant [86].

Figure 4.10 depicts the evaluation methodology used to conduct the study. The *introduction* explains the task and the test procedure. Furthermore, the participants are asked to agree to a disclaimer. The *pre-questionnaire* allows us to gather demographic information and helps us to check whether the participants are from the countries we asked for at Microworkers. The *training* phase should allow the participants to become familiar with the task and the rating possibilities. In order to introduce AMP to the participants and to its effects on the media playback we selected the video sequence Babylon A. D. taken from [87]. The training sequence is presented three times with three different media playback rates  $\mu \in \{1, 0.5, 2\}$ , i.e.,  $\mu = 1$  corresponds to the nominal playback rate  $\mu_0$ ,  $\mu = 0.5$  is half the nominal playback rate  $\mu_0$ , and  $\mu = 2$  denotes twice the nominal playback rate  $\mu_0$ . The training

sequence is presented for its whole duration with the mentioned media playback rates.

After the training phase the *main evaluation* starts. For the stimulus we selected a video sequence with audio and a duration of 51 seconds from the beginning of the open source movie Big Buck Bunny, which is available at [74]. We annotated the content sections for which the media playback rate is increased or decreased. The playback rate changes were initially randomly scattered throughout the whole video sequence with a cumulative duration of 8.84 seconds, thus reflecting 17.3% of the sequence's duration. As indicated in Figure 4.10 we use a single stimulus method as recommended in [56, 78]. We use a continuous rating scale  $[0, 100]$  represented by a slider. Furthermore, we randomly inserted a control question after a stimulus presentation. In particular, this control question asks the participants what they have seen in the previous video sequence with three possible answers provided. In total the sequence is presented nine times to the participants with the following configurations for the playback rate  $\mu \in \{0.5, 0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8, 2\}$ .  $\mu = 1$  denotes the case of nominal playback rate  $\mu_0$  (i.e., without any playback rate changes) and, thus, depicts a hidden reference. Please note that we only modify the playback rate of specific sections and not for the entire video sequence.

At the end, we ask the participants to fill out a short *post-questionnaire* which provides the possibility to give feedback and to state whether one had already participated in a similar study. The study is conducted by adopting an open-source Web-based QoE assessment platform presented in Chapter 5.5.3. Please note that the media player used by this platform uses the Waveform Similarity based Overlap-Add which tries to maintain the pitch of audio when increasing or decreasing the playback rate [88]. In addition to its usability, the platform provides several mechanisms to track the behavior of participants in terms of measuring the time of each stimulus presentation and the possibility to ask control questions. Screening and filtering of the participants is done according to the three steps discussed in Section 4.2.6.

### 4.3.3 Statistical Analysis of the Results

After screening the participants and their responses, the ratings for each stimulus presentation were subject to statistical significance tests. According to the Central

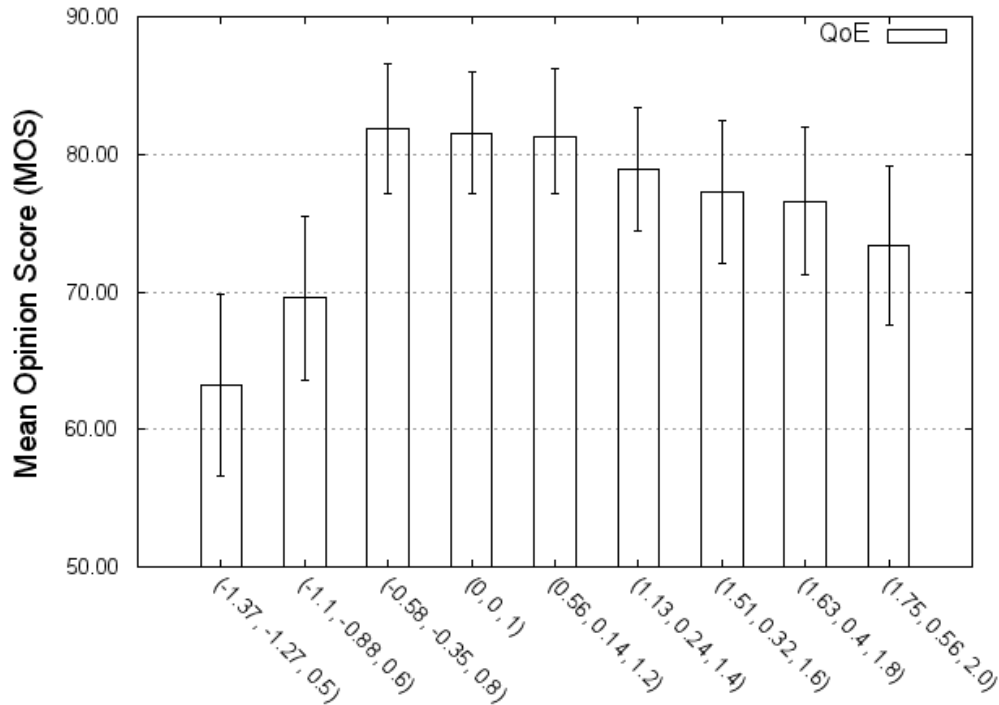


Figure 4.11: MOS and 95% CI for  $(\bar{d}_v, \bar{d}_a, \mu)$ .

Limit Theorem [81] we assume that the ratings of the participants are normally distributed. Nevertheless, we have conducted a Shapiro-Wilk-test to assess whether the ratings are normally distributed [83]. The null hypothesis ( $H_0$ ), stating that a normal distribution is present, was accepted for each configuration of playback rates of the stimulus presentations. By the use of the measures introduced in Section 4.3.1 the distortion caused by increasing or decreasing the playback rates for the selected content sections are expressed by  $\bar{d}_a$  for the average distortion in the frequency domain for the audio channels and  $\bar{d}_v$  for the average distortion of motion for video.

Figure 4.11 depicts for each triple  $(\bar{d}_v, \bar{d}_a, \mu)$  the assessed Mean Opinion Score (MOS). It can be observed that playback rates near the nominal playback rate of  $\mu = 1$  cause only a slight drop in QoE. A Student's t-test supports this finding by stating no significant difference in MOS between the reference of  $\mu = 1$  and the following playback rates:  $\mu = 0.8$  ( $p = 0.93, t = -0.083$ );  $\mu = 1.2$  ( $p = 0.92, t = 0.096$ );  $\mu = 1.4$  ( $p = 0.81, t = 0.42$ );  $\mu = 1.6$  ( $p = 0.22, t = 1.23$ );  $\mu = 1.8$  ( $p = 0.16, t = 1.41$ ). These results indicate that the users could not notice a significant difference for playback

rates  $\mu \in [0.8, 1.8]$ .

For the other playback rates it can be observed that the QoE significantly degrades. A Student's t-test revealed that there exists a significant difference in the MOS from the nominal playback rate of  $\mu = 1$  and the media playback rates with following values:  $\mu = 0.5$  ( $p = 0.00, t = 4.5217$ );  $\mu = 0.6$  ( $p = 0.002, t = 3.2$ );  $\mu = 2$  ( $p = 0.03, t = 2.19$ ). These results provide the evidence that users perceived a significant difference between the reference and the test conditions.

In the following we investigate how well our measures correlate with the MOS for the different playback rate configurations. The Pearson correlation coefficient for  $\overline{d}_v$  and the QoE ratings is  $\rho = 0.43$ . For  $\overline{d}_a$  the Pearson correlation coefficient is  $\rho = 0.679$ . If we take the absolute values of  $|\overline{d}_a|$  and  $|\overline{d}_v|$  for calculating the Pearson correlation coefficient we obtain following values for the linear correlation between the measures and the QoE scores. The Pearson correlation coefficient for  $|\overline{d}_a|$  and the QoE ratings is  $\rho = -0.5549$  and for  $|\overline{d}_v|$  and the QoE ratings  $\rho = -0.9565$ . For the semi-metric in the audio domain both  $\overline{d}_a$  and  $|\overline{d}_a|$  show a low linear correlation with the obtained MOS, respectively. For the measure in the video domain we have obtained contrary results. Taking  $\overline{d}_v$  we have a low linear correlation between the measure and the QoE ratings but if we take the semi-metric  $|\overline{d}_v|$  there exists a high negative linear correlation. Nevertheless, if we want to retain the ability of distinguishing whether the playback rate has been decreased or increased we have to use the measures. The low values between the signed measures and the QoE scores show us that assuming a linear relationship between the distortion measures and the assessed QoE may not be appropriate. Therefore, we try to find a model that explains this correlation better than a linear model which will be discussed in the next sections.

Finally, the results indicate that with an increase in  $|\overline{d}_v|$  and  $|\overline{d}_a|$  the QoE is reduced. Interestingly, the QoE does not decrease linearly. For the playback rate changes in the range of  $[0.8, 1.8]$  the QoE remains high compared to the reference. Increasing or decreasing the playback rate further causes a huge decrease in the QoE.

### 4.3.4 QoE Utility Model for AMP

In [89] several AMP algorithms were assessed on their impact on the QoE regarding QoS parameters such as the initial playback delay, loss rate, underflow time ratio, and the playback rate by introducing cross traffic. The actual impact on the playback of the content was not taken into account. Furthermore, there is the need for a model which can be easily combined with other QoE metrics in order to assess the QoE of a system that uses AMP. The presented results of the conducted subjective quality assessment using crowdsourcing gave us a first impression on how the QoE degrades with an increase or decrease in the playback rate when selecting content sections with a short time duration. With the knowledge that the Pearson correlation between the QoE scores and the audio/video measure is low, we try to find a function which allows to approximate the QoE more precisely than a linear function could do. At this point let us pick up the IQX hypothesis which describes the interdependency between the QoE and Quality of Service (QoS) parameters [90]. The hypothesis states that if the QoS parameters changed the resulting QoE depends on the QoE prior to the change in the QoS parameters. Our distortion measures the QoS in the temporal domain (lower is better). If we consider the impact in the audio and video domain separately, we have two partial differential equations.

$$\frac{\partial QoE}{\partial d_a} = \frac{\alpha(d_a)}{d} \cdot QoE(d_v, d_a)$$

$$\frac{\partial QoE}{\partial d_v} = \frac{\beta(d_v)}{d} \cdot QoE(d_v, d_a)$$

The solution to this system of partial differential equations is given in Equation 4.13.

$$QoE(d_v, d_a) = e^{\alpha(d_a)} e^{\beta(d_v)} \cdot C \quad (4.13)$$

Plugging in our marginal condition  $QoE(0, 0) = QoE_{wo}$ , where  $QoE_{wo}$  denotes the QoE without the distortions caused by AMP (our baseline where the playback rate  $\mu$  was set to one in the subjective quality assessment), and if we assume that  $\alpha(0) = 0$

and  $\beta(0) = 0$  (as the initial conditions), we have  $QoE(d_v, d_a) = e^{\alpha(d_a)}e^{\beta(d_v)} \cdot QoE_{wo}$ . We call  $\zeta(d_a, d_v)_{\theta} : \mathbb{R} \times \mathbb{R} \rightarrow [0, 1]$ ,  $(d_v, d_a) \mapsto e^{\alpha(d_a)}e^{\beta(d_v)}$  the coefficient of degradation.

The coefficient of degradation shall be *one* if and only if both  $d_a$  and  $d_v$  are approaching *zero*. Therefore, we introduce the parameter vector  $\theta \in \mathbb{R}^4$  for  $\alpha(d_a)$  and  $\beta(d_v)$  such that  $\alpha(d_a) = -\frac{(d_a - \theta_1)^2}{2\theta_2^2}$  and  $\beta(d_v) = -\frac{(d_v - \theta_3)^2}{2\theta_4^2}$ .  $\theta_1$  and  $\theta_3$  should be *zero* in order to reflect our requirement of having the highest QoE when having zero distortion, but if we fit our model to the data obtained by the subjective quality assessment we have to relax our requirement such that we allow a small deviation from the *optimal* case.  $\theta_2$  and  $\theta_4$  provide a scaling and will account for the variance of the data used to fit the model. Therefore, we have to require that  $|\theta_2|, |\theta_4| > 0$ .  $\frac{1}{2}$  is introduced for normalization purposes. Equation 4.14 depicts the final coefficient of degradation for both audio and video.

$$\zeta(d_a, d_v)_{\theta} = e^{-\frac{1}{2}\left(\frac{d_a - \theta_1}{\theta_2}\right)^2} e^{-\frac{1}{2}\left(\frac{d_v - \theta_3}{\theta_4}\right)^2} \quad (4.14)$$

In order to verify that  $\zeta$  fulfils the requirements (concavity and a global maximum) we will take a closer look at  $\ln(\zeta(d_a, d_v)_{\theta})$ . Therefore, we state the following theorem.

**Theorem 4.3.1**  $\zeta(d_a, d_v)_{\theta}$  is logarithmic concave for  $(d_a, d_v)$  and  $|\theta_2|, |\theta_4| \in \mathbb{R} : |\theta_2|, |\theta_4| > 0$ . Furthermore,  $\ln(\zeta(d_a, d_v)_{\theta})$  has a global maximum.

In order to proof Theorem 4.3.1 we use the following lemma.

**Lemma 4.3.2** The function  $f(x) = -(x - c)^2$  with  $x \in \mathbb{R}$  is concave ( $f(\alpha \cdot x + (1 - \alpha) \cdot y) \geq \alpha \cdot f(x) + (1 - \alpha) \cdot f(y)$ , cf. [91]) and  $c \in \mathbb{R}$  a constant.

**Proof of Lemma 4.3.2** We apply the theorem that states that  $f(x)$  is concave if and only if the Hessian ( $\mathcal{H}_f$ ) is negative semi-definite ( $h^T \mathcal{H}_f h \leq 0$  for any  $h \in \mathbb{R}^n$  and  $h \neq \mathbf{0}$ ) or strict concave iff the Hessian is negative definite ( $h^T \mathcal{H}_f h < 0$ ) [92]. In the case of  $f(x) = -(x - c)^2$  the Hessian has only one entry

$$\frac{d^2}{dx^2} f(x) = -2 < 0$$

Thus,  $f(x)$  is strict concave. □



**Proof of Theorem 4.3.1**

$$\ln(\zeta(x, y)_{\theta}) = -\frac{1}{2} \left( \frac{\alpha x_1 + (1 - \alpha)x_2 - \theta_1}{\theta_2} \right)^2 - \frac{1}{2} \left( \frac{\alpha y_1 + (1 - \alpha)y_2 - \theta_3}{\theta_4} \right)^2$$

Applying Lemma 4.3.2 we have:

$$\begin{aligned} & \ln(\zeta(\alpha x_1 + (1 - \alpha)x_2, \alpha y_1 + (1 - \alpha)y_2)_{\theta}) > \\ & > -\frac{1}{2\theta_2^2} (\alpha(x_1 - \theta_1)^2 + (1 - \alpha)(x_2 - \theta_1)^2) - \frac{1}{2\theta_4^2} (\alpha(y_1 - \theta_3)^2 + (1 - \alpha)(y_2 - \theta_3)^2) = \\ & \alpha \cdot \ln(\zeta(x_1, y_1)_{\theta}) + (1 - \alpha) \cdot \ln(\zeta(x_2, y_2)_{\theta}) \end{aligned}$$

The global maximum is attained for  $x = \theta_1$  and  $y = \theta_3$  (evidence provided by Lemma 4.3.2,  $(\theta_1, \theta_3)$  is the only point for which  $\nabla \ln(\zeta(x, y)_{\theta}) = 0$ ). This completes the proof.  $\square$

We have shown that  $\ln(\zeta(d_a, d_v)_{\theta})$  has a global maximum and, therefore,  $\zeta(d_a, d_v)_{\theta}$  has a global maximum (because  $e^x$  is monotonically increasing) at  $d_a = \theta_1$  and  $d_v = \theta_3$  (cf. Figure 4.12). This maximum is attained if the distortion approaches  $\theta_1$  and  $\theta_3$  in both domains and, therefore, the QoE is at its maximum. The results of the subjective quality assessment indicated that small distortions in the proposed measures, especially for audio, do already cause an impact on the QoE. Therefore, we formulate our model as follows:

$$QoE(\bar{d}_v, \bar{d}_a) = QoE_{wo} \cdot \zeta(\bar{d}_v, \bar{d}_a)_{\theta^*} + \varepsilon, \quad (4.15)$$

where  $\theta^*$  represents the optimal parameter vector such that for a cost function  $f(\theta)$  for each  $\theta$  it holds that  $f(\theta) \geq f(\theta^*)$ . Thus,  $\theta^*$  representing the optimal solution to a minimization problem.  $\varepsilon$  denotes the error/residuals of the model. We further assume that the error is normally distributed. The  $QoE_{wo}$  states the QoE without any playback rate changes. This  $QoE_{wo}$  can be derived from QoS parameters (e.g., bit-rate, resolution, delay, jitter, etc.) by the use of existing models such as those proposed in [89, 93]. The assessment of the actual  $QoE_{wo}$  is out of scope.

### 4.3.5 Instantiation and Validation of the Utility Model

In order to instantiate our proposed utility model (cf. Equation 4.15) we use the responses received by the conducted subjective quality assessment using crowdsourcing. We fit our model to the obtained data by using multiple instances of the conjugate gradient method [94]. For the cost function we used the Least-Square-Estimator or the squared  $l^2$ -norm given by  $\| \ln(\zeta_{\theta}(d_a, d_v)) - \ln(z) \|_2^2$  depicted by Equation 4.16. This function is neither concave nor convex for  $\theta$ , but it is a  $C^k$  function ( $C^k$  is the room of  $k$  times continuous differentiable functions) if  $|\theta_2|, |\theta_4| > 0$ .

$$f(\theta) = \sum_{(x,y,z) \in M} (\ln(\zeta(x,y)_{\theta}) - \ln(z))^2, \quad (4.16)$$

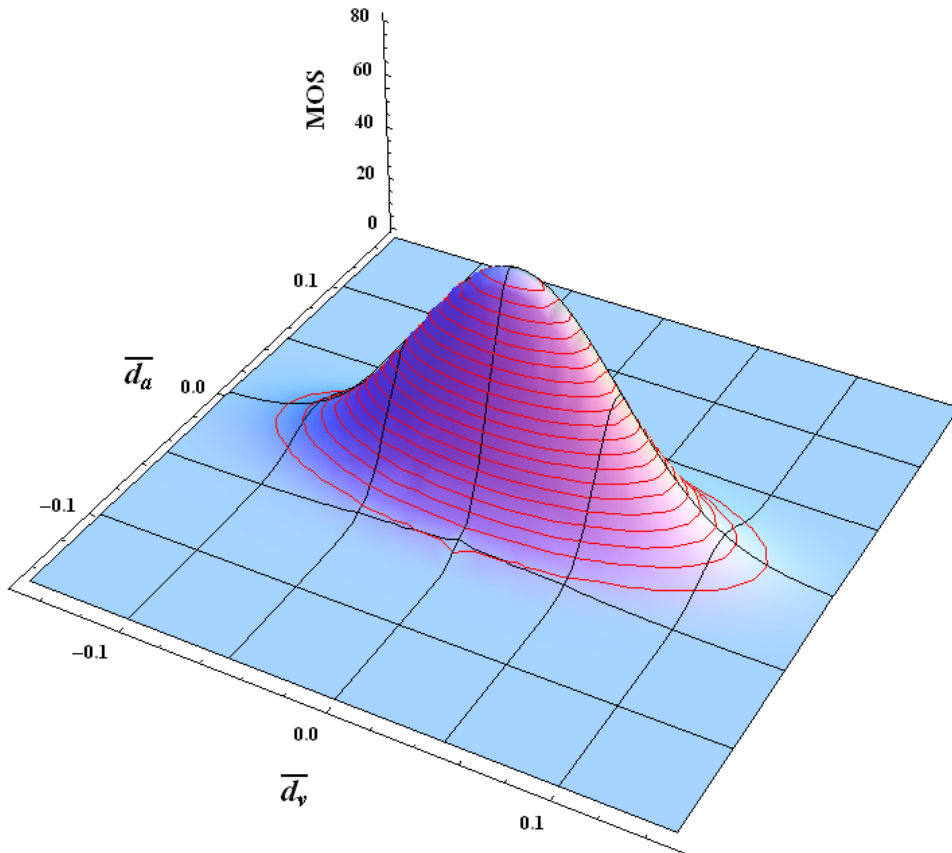
where  $M := \{(x, y, z) \in \mathbb{R}^3 | z = \frac{QoE(x,y)}{\max_{x,y}\{QoE(x,y)\}}\}$  is the set of 3-tuples with  $z$  representing the QoE assessed for  $(\bar{d}_a, \bar{d}_v)$ . Therefore, we try to find the parameter vector  $\theta^*$  that minimizes our  $f(\theta)$ . For determining the conjugated search directions we use the method proposed by Polak-Ribiere [95]. In order to find a near optimal vector  $\theta$  we use multiple instances of the conjugate gradient algorithm with starting points uniformly distributed in the interval of  $]0, 1]$  for all  $\theta_i, 1 \leq i \leq 4$ . We selected  $\theta$  that provided the lowest costs in terms of our cost function  $f(\theta)$ .

By the use of the conjugate gradient method we fitted  $\zeta(x, y)_{\theta}$  to the responses received during our study discussed in Section 4.3.2 and we found the following values for the parameter vector  $\theta^* = (0.0011, 0.0482, -0.0004, 0.0184)^T$ . Equation 4.17 depicts the fully instantiated utility model.

$$QoE(\bar{d}_v, \bar{d}_a)_{\theta^*} = QoE_{wo} \cdot e^{-\frac{1}{2}(\frac{x-0.0011}{0.0482})^2} e^{-\frac{1}{2}(\frac{y+0.0004}{0.0184})^2} \quad (4.17)$$

Figure 4.12 depicts the  $QoE(\bar{d}_v, \bar{d}_a)_{\theta^*}$  by the use of the fitted  $\zeta$ . An interesting finding is that a distortion in audio impacts the QoE more than the same amount of distortion in video. This can be observed by comparing the second and fourth component of  $\theta$  or by taking a look at Figure 4.12.

To test how well our utility model fits the actual data, we have conducted an analysis of variance (ANOVA) on how the fitted model reflects the variability of the

Figure 4.12: Fitted  $\zeta$  for the received responses.

actual data. The ratio of the sum of squares of the model and sum of squares of the actual data revealed that the instantiated model reflects 92.48% of the variability of the actual data. Furthermore, we conducted an F-test to test whether  $\theta$  is the zero vector. The test revealed that the null-hypotheses can be rejected with  $p = 2.49 \cdot 10^{-4}$  and  $F = 27.84$  for  $\alpha = 5\%$ . A Lilliefors-Kolmogorov-Smirnov test accepted the null hypothesis with  $D = 0.0465, p = 0.1232, p > 0.05$  that the residuals are normally distributed. We have shown that  $\zeta(x, y)$  fits our purpose quite well and provides the possibility to estimate the coefficient of degradation for different values of  $(\bar{d}_v, \bar{d}_a)$ .

### 4.3.6 Discussion and Conclusion

The results of the subjective quality assessment using crowdsourcing lead us to the hypotheses that the correlation between the distortion in the video/audio domain and the QoE of playback rate variations can be described by a non linear model.

The introduced model lead us to the finding that audio plays an important role when increasing or decreasing the playback rate. This is depicted by Figure 4.12 and denoted by Equation 4.17. Comparing our results to the results obtained by other subjective quality assessments which assess the QoE of playback variations for video only [22, 48, 49] we can see that altering the playback rate for the combination of audio and video has a very different impact on the QoE. Please note that for video only, the human perception is more tolerant for playback rate variations. This is not the case for audio and the combination of audio and video as shown by the results.

The contribution made here is twofold. First, we have shown that there is a significant difference between the impact of increasing and decreasing the playback rate on the QoE. An interesting finding is that increasing the playback rate for specific content sections have a lower impact on the QoE than decreasing the playback rate by the reciprocal of the increase of the playback rate. Second, we have introduced measures that quantify the distortion in audio and video caused by increasing or decreasing the playback rate. With the use of these measures (or the semi-metrics) we have derived a utility model.

## 4.4 Dynamic AMP

The previous sections have shown that using features from the audio and video domain of multimedia content can help in selecting appropriate content sections for increasing or decreasing the playback rate in order to mitigate the effects of AMP with respect to the resulting QoE. According to the insights provided by the studies presented in the previous sections we will formulate a general optimization problem that will be instantiated by using the distortion semi-metrics introduced in Section 4.3.1.

In Chapter 3 we have dealt with all the essential mechanisms for negotiating on a reference playback timestamp to which each peer will have to synchronize its playback. The push-based approach discussed in [32] uses AMP which implies that the playback rate is increased and decreased if the asynchronism at each peer has been calculated. This approach does not consider the impact on the QoE nor vary the playback rate dynamically every time a peer synchronizes its playback (cf. Section 2.2). Therefore,

we use our findings from Section 4.2 and Section 4.3 to come up with a constrained optimization problem that aims on finding content sections that have a lesser impact on the QoE than simply increasing or decreasing the playback rate without taking into account content features.

#### 4.4.1 Dynamic AMP for IDMS

We propose using the current buffer contents of a peer for determining when and for which duration we should change the playback rate and formulate the following general constrained optimization problem:

$$\arg \min_{\mathbf{x}} f(\mathbf{x}) \quad (4.18a)$$

$$x_2 \cdot (x_3^{\text{sign}(\xi)} - 1) \cdot \text{sign}(\xi) = |\xi| \quad (4.18b)$$

$$L \leq B - x_2 \cdot x_3 + x_2 \cdot \frac{b_c}{b_r} \quad (4.18c)$$

$$x_1 \leq T \quad (4.18d)$$

$$x_2 \leq t_{max} \quad (4.18e)$$

where  $\mathbf{x} \in \mathbb{R}^3$  denotes our vector with  $(x_1, x_2, x_3)^T$ ,  $x_1$  denotes the starting time of the playback rate change relative to the current buffer,  $x_2$  denotes the duration of the playback rate change, and  $x_3$  denotes the target playback rate. Our aim is to find values for  $\mathbf{x}$  such that  $\mathbf{x}^*$  is a minimizer for  $f(\mathbf{x})$  ( $\forall \mathbf{x} \in \mathbb{R}^3 : f(\mathbf{x}^*) \leq f(\mathbf{x})$ ).  $f(\mathbf{x})$  can be any function that models the impact of changing the playback rate for the given duration on the QoE. Furthermore, we define constraints that reduce the set of feasible points. First, we introduce the constraint as depicted by Equation 4.18b which states that the given asynchronism should be compensated for by selecting appropriate values for  $x_2$  and  $x_3$ , respectively.  $\xi$  denotes the asynchronism identified by comparing the current playback timestamp to the reference timestamp. If  $\xi < 0$  the playback rate is reduced and if  $\xi > 0$  the playback rate is increased in order to compensate for the asynchronism. Equation 4.18c avoids buffer underflows and, thus, stalls in the multimedia playback. This constraint only applies if the playback rate is increased. In particular,  $L$  denotes the lower buffer threshold in seconds,  $B$  the

current buffer fill state in seconds,  $b_c$  the client's bandwidth, and  $b_r$  the bit-rate of the selected representation. Furthermore, we constrain the starting time ( $T$ ) of the playback rate variation by bounding  $x_1$  (cf. Equation 4.18d). Equation 4.18e limits the duration ( $t_{max}$ ) of the playback variation.

We use the distortion semi-metrics introduced in Section 4.3.1 and modify them slightly to fit our needs as depicted in Equation 4.19 for  $d_v(\mathbf{x})$  and Equation 4.20 for  $d_a(\mathbf{x})$ .

$$d_v(\mathbf{x}) = \sum_{j=F_s(x_1)}^{F_e(x_1, x_2, \mu_0)} f_v(j) - \sum_{j=F_s(x_1)}^{F_e(x_1, x_2, x_3)} f_v(j) \quad (4.19)$$

$$d_a(\mathbf{x}) = \sum_{c=1}^C \left( \left( \sum_{u=F_s(x_1)}^{F_e(x_1, x_2, \mu_0)} \sum_{k=0}^{S_f} |\hat{a}_{c_u}(k)| - \sum_{u=F_s(x_1)}^{F_e(x_1, x_2, x_3)} \sum_{k=0}^{S_f} |\hat{a}_{c_u}(k)| \right) \right) \quad (4.20)$$

The results of the subjective quality assessment using crowdsourcing and the utility model presented in Section 4.3.3 and Section 4.3.4 has already shown that the introduced measures are suitable for estimating the QoE. Therefore, we use the utility model introduced in Section 4.3.4 in order to approximate the impact of playback rate changes on the QoE. We do not have to know the real  $QoE_{wo}$  because this is only a factor and it suffices to minimize

$$c(\mathbf{x}) = -\ln(\zeta(d_a(\mathbf{x}), d_v(\mathbf{x}))_{\theta^*}) \quad (4.21)$$

in order to maximize the QoE. We use the parameter vector  $\theta^*$  given in Section 4.3.4. This can be even more simplified by using

$$g(\mathbf{x}) = \sqrt{d_v^2 + d_a^2} = \|d(v, a)\|_2, \quad (4.22)$$

where  $d(v, a) = (d_v, d_a)^T$  because if  $\mathbf{x}$  minimizes  $c(\mathbf{x})$  then  $\mathbf{x}$  also minimizes  $g(\mathbf{x})$ .

The highest asynchronism in our IDMS system is given by Equation 3.1. The initial asynchronism of a newly joined peer depends on the time the peer requires for downloading a sufficient number of segments such that it can start the playback. If the asynchronism is greater than the current buffer fill state the peer will not be able

to compensate the asynchronism. In such cases, the synchronization process must be partitioned into a number of smaller synchronization processes. Therefore, we define the asynchronism for each synchronization process as  $\delta_{k+1} = \min(\xi - \delta_k, B - L)$ , starting with  $\delta_0 = \min(\xi, B - L)$ . This has no effect if  $\xi$  is lower than zero because reducing the playback rate does not affect the buffer fill state.

Using the sequential unrestricted minimization technique, we transform the general optimization problem given in Equation 4.18 into the following optimization problem:

$$\arg \min_{\mathbf{x}} f(\mathbf{x}) = \begin{cases} g(\mathbf{x}) + \gamma \cdot \frac{1}{2} \cdot \sum_{i=1}^3 p_i(\mathbf{x}) & \text{if } \xi \geq 0 \\ g(\mathbf{x}) + \gamma \cdot \frac{1}{2} \cdot \sum_{i=2}^3 p_i(\mathbf{x}) & \text{if } \xi < 0 \end{cases} \quad (4.23a)$$

$$p_1(\mathbf{x}) = \min\{0, B - x_2 \cdot x_3 + x_2 \cdot \frac{b_c}{b_r} - L\}^2 \quad (4.23b)$$

$$p_2(\mathbf{x}) = \min\{0, T - x_1\}^2 \quad (4.23c)$$

$$p_3(\mathbf{x}) = \min\{0, t_{max} - x_2\}^2 \quad (4.23d)$$

Equation 4.23a shows the transformed cost function. To reduce the set of feasible points we use the constraint given in Equation 4.18b and apply the implicit function theorem, reducing  $x_2$  to a function of  $x_3$  by  $u(x_3) = \text{sign}(\delta_k) \cdot \frac{|\delta_k|}{x_3^{\text{sign}(\delta_k)} - 1}$  for  $x_3 \neq 1$  for further details see Theorem 4.4.1 and its corresponding proof. Thus, we try to find values  $x_1$  and  $x_3$  that minimize  $f(\mathbf{x})$ . The penalty function  $p_1(\mathbf{x})$  (cf. Equation 4.23b) states that the buffer fill state shall not be drained below the threshold  $L$  when increasing the playback rate.  $p_2(\mathbf{x})$  (cf. Equation 4.23c) depicts the costs that depend on the starting point of the playback rate variation.  $p_3(\mathbf{x})$  (cf. Equation 4.23d) states that a peer should be synchronized within  $t_{max}$  seconds.  $\gamma$  denotes the penalty factor for the transformed constraints ( $\gamma > 0$ ). For  $\gamma \rightarrow \infty$  the solution found will be exact. Since we run into numerical problems when setting  $\gamma \rightarrow \infty$  we have to find a tradeoff between exactness and fulfilling the constraints. For solving the optimization problem during the multimedia playback we use multiple instances of the Nelder-Mead algorithm with different starting points (equally distributed) which yields a

set of local minima  $E$  [96]. There may be more than a single optimal solution that minimizes the impact on the QoE because  $f(\mathbf{x})$  is not convex since different values for  $\mathbf{x}$  may yield the same values for  $d_a(\mathbf{x})$  and  $d_v(\mathbf{x})$  depending on the multimedia content and it is not continuous, this can be easily verified because the features used are not continuous in time. We then select  $\mathbf{x}^* = \arg \min_{\mathbf{x}} \{E\}$  as a minimizer of  $f(\mathbf{x})$ . Our dynamic AMP approach overcomes asynchronism by searching for content sections where the playback rate may be increased or decreased having the least impacting on the QoE.

**Theorem 4.4.1** The function  $f(x_3, x_2) = x_2 \cdot (x_3^{\text{sign}(\delta_k)} - 1) \cdot \text{sign}(\delta_k) - |\delta_k|$  has an implicit function such that  $f(x_3, u(x_3)) = 0$  with  $u(x_3) = \text{sign}(\delta_k) \cdot \frac{|\delta_k|}{x_3^{\text{sign}(\delta_k)} - 1}$  for  $x_3 \neq 1$ .

**Proof of Theorem 4.4.1** Applying the implicit function Theorem [92] on  $f(x_3, x_2) = x_2 \cdot (x_3^{\text{sign}(\delta_k)} - 1) \cdot \text{sign}(\delta_k) - |\delta_k|$  tells us that we have to check whether  $(\frac{\partial f}{\partial x_2})^{-1}$  exists because  $D(f(x_3, u(x_3))) = \frac{\partial f}{\partial x_3}(x_3, u(x_3)) + \frac{\partial f}{\partial x_2}(x_3, u(x_3)) \cdot \frac{du}{dx_3}(x_3)$ . It follows that  $\frac{du}{dx_3}(x_3) = - \left( \frac{\partial f}{\partial x_2}(x_3, u(x_3)) \right)^{-1} \cdot \frac{\partial f}{\partial x_3}(x_3, u(x_3))$ . Therefore, we derive  $f(x_3, x_2)$  according to  $x_2$  and check whether it is invertible.

$$\frac{\partial f}{\partial x_2} = (x_3^{\text{sign}(\delta_k)} - 1) \cdot \text{sign}(\delta_k)$$

$\frac{\partial f}{\partial x_2}$  is invertible and continuous for  $x_3 \neq 1$  because  $|\det \left( \frac{\partial f}{\partial x_2} \right)| > 0$  for  $x_3 \neq 1$ . Thus, the implicit function is  $u(x_3) = \text{sign}(\delta_k) \cdot \frac{|\delta_k|}{x_3^{\text{sign}(\delta_k)} - 1}$ . This completes the proof.  $\square$

#### 4.4.2 Evaluation of Dynamic AMP for IDMS

We compare the solution of our problem formulation provided by Equations 4.23a to 4.23d against static AMP and naïve skipping and pausing of multimedia content. A first hint that the distortion semi-metrics (cf. Equation 4.19 and Equation 4.20) are suitable gives the introduced QoE utility model in Section 4.3.4 and if we take a



look at the Pearson correlation coefficients of  $g(\mathbf{x})$  (cf. Equation 4.22) for the study conducted in Section 4.3.3.

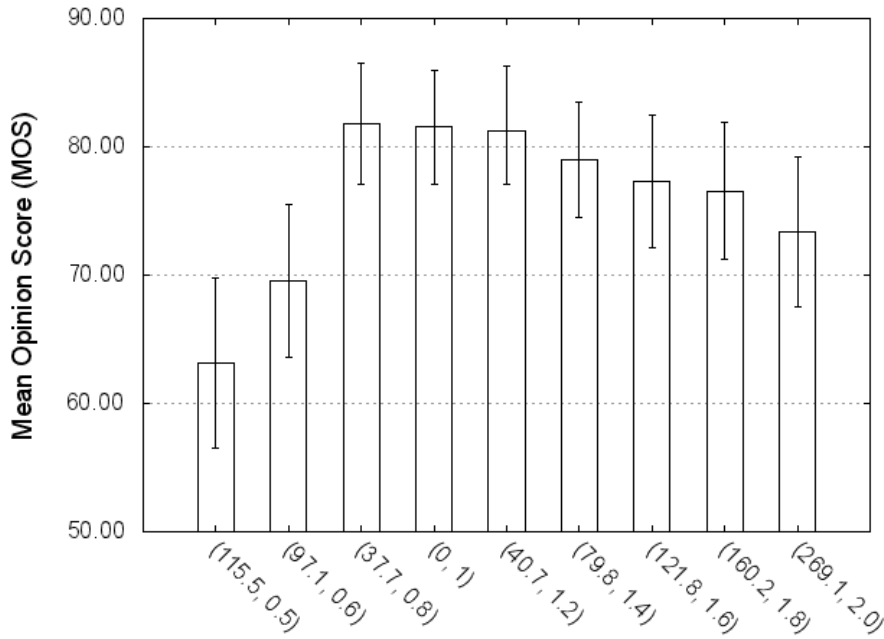


Figure 4.13: MOS and 95% CI for  $(\overline{g(X)}, \mu)$ .

Figure 4.13 depicts the MOS and the average values for  $g(\mathbf{x})$  (denoted by  $\overline{g(X)}$ ) taken from the study described in Section 4.3.2. For playback rates greater than the nominal playback rate the Pearson correlation coefficient is  $\rho = 0.975$  with the probability of encountering a false positive being  $p = 0.0009$  for playback rates higher than the reference playback rate. For playback rates lower than the nominal playback rate exhibit a high negative correlation with  $\rho = -0.995$  and  $p = 0.0047$ . These Pearson correlation coefficients are the result from the distortion values taken from Section 4.3.3. As already mentioned that minimizing Equation 4.22 is equivalent to minimizing Equation 4.21 with respect to  $\mathbf{x}$ . Therefore, if  $\mathbf{x}^*$  denotes the minimum it also maximizes the QoE. Thus, the feasibility of our dynamic AMP approach is given.

For the comparison between dynamic AMP, static AMP and pausing or skipping multimedia content, we selected the open source movie Big Buck Bunny [74] to demonstrate the difference between these three approaches. The static AMP

refers to increasing or decreasing the playback rate such that the peer approaches the synchronization reference. The playback rate is instantly altered (when the reference playback timestamp is finally calculated). We compare the three AMP approaches with different starting times, asynchronisms, and for static AMP we selected the following set of playback rates  $R = \{0.5, 0.6, 0.8, 1.2, 1.4, 1.6, 1.8, 2.0\}$ . The (absolute) starting points from which the synchronization at the peers shall be carried out and the corresponding asynchronism is given by the following set of tuples  $T = \{(6s, 2s), (9s, 3s), (16s, 4s), (34s, 3s), (6s, -2s), (9, -3s), (16s, -4s), (34s, -3s)\}$ . The first value of a tuple denotes the starting time and the second value the asynchronism. The values are arbitrarily chosen and shall demonstrate the performance of dynamic AMP using a real world example. For our dynamic AMP approach we assume that the buffer is filled and has a maximum size of 30 seconds of multimedia content. For the lower buffer bound  $L$  (cf. Equation 4.23b) we assume that the buffer level using dynamic AMP shall not drop below six seconds. We restrict dynamic AMP such that it finds a content section within a range of 30 seconds from the actual starting point (cf. Equation 4.23c). The maximum duration of a content section (cf. Equation 4.23d) is set to 30 seconds. The test cases for static AMP is the set  $T \times R$ .

Figure 4.14 depicts the relative distortion of dynamic AMP to naïve skipping and pausing of multimedia content. The x-Axis denotes the test case and the actual values found when solving the optimization problem. The results show that dynamic AMP is able to maintain a very low distortion compared to instantly skipping and pausing. Figure 4.15 depicts the relative distortion of dynamic AMP to naïve skipping and pausing of multimedia content. The x-Axis denotes the test cases. Static AMP is sometimes better than skipping or pausing but very often it exceeds it in terms of caused distortion. Comparing Figure 4.14 and Figure 4.15, dynamic AMP always finds content sections for which the process of carrying out the synchronization causes less distortion in the perceptual domain than the other two approaches.

The results for the selected test cases clearly show that dynamic AMP is able to provide a better QoE than static AMP or skipping/pausing of multimedia content to achieve a synchronous playback. Dynamic AMP requires the peer to compute content features (motion vectors for the video domain and spectral energy for the

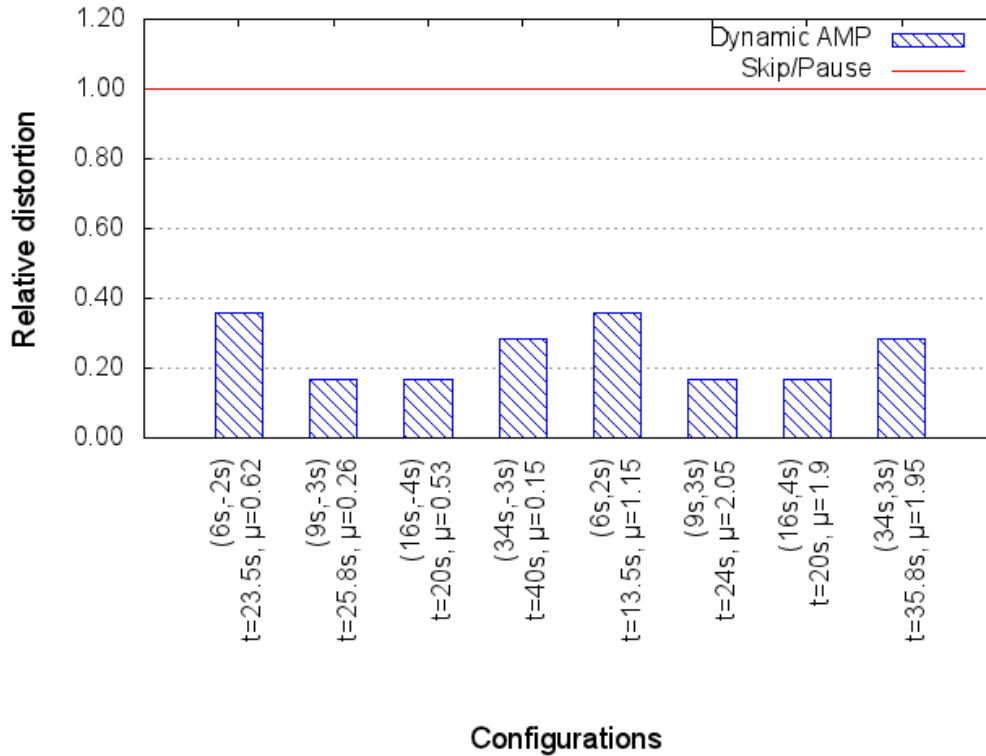


Figure 4.14: Dynamic AMP compared to skipping and pausing multimedia content for the given test cases, where  $t$  and  $\mu$  denote the resulting starting point and playback rate when searching for a solution to the optimization problem, respectively.

audio domain). Thinking of mobile peers where excessive resource consumption results in a higher power consumption, the content features can be pre-calculated and signaled in the beginning of the multimedia streaming session. This will reduce the computational effort to just solving the optimization problem.

## 4.5 Conclusion

In this chapter we have shown that selecting the content sections for applying AMP is crucial and if the selection depends on content features we are able to achieve a higher QoE than without taking into account content features. Therefore, we have introduced measures (semi-metrics) that allow a quantification of the distortion caused by AMP using content features. We have presented research towards the following research objectives (cf. Section 1.2):

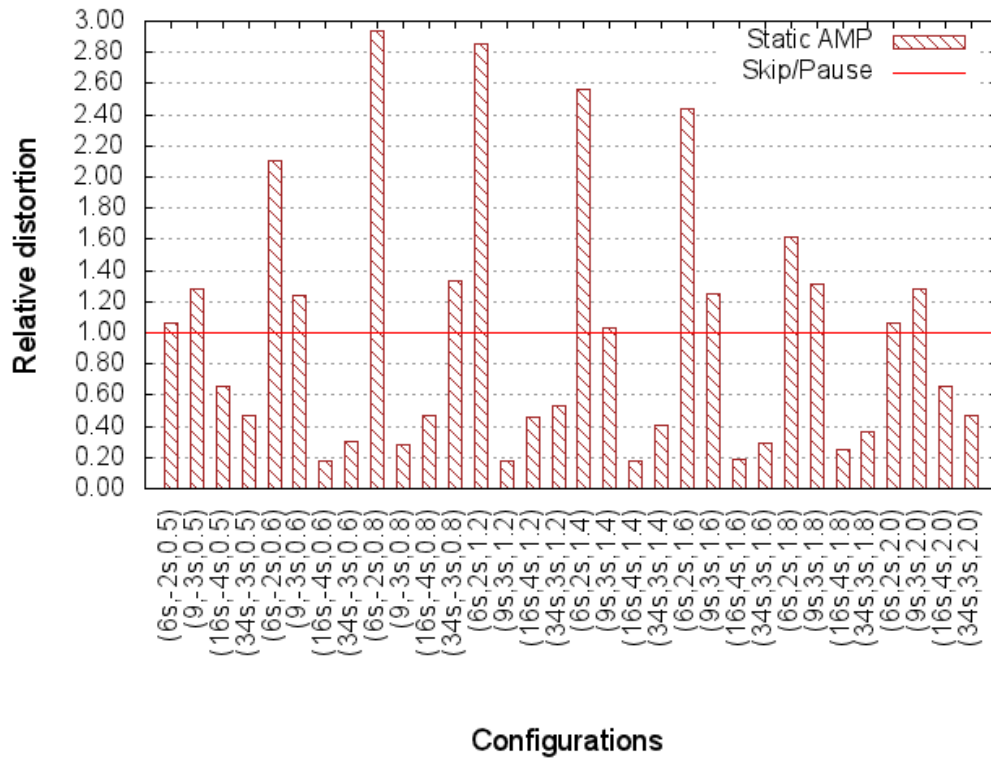


Figure 4.15: Static AMP compared to skipping and pausing multimedia content for the given test cases.

- (5) to investigate the possibility of carrying out the synchronization in terms of adaptively changing the media playback by increasing or decreasing the playback rate of the multimedia playback.
- (6) to assess whether content features allow to decrease the impact of increasing or decreasing the playback rate on the QoE.
- (7) to analyze and quantify the impact of AMP on the QoE.
- (8) to utilize content features for dynamically selecting content sections that are appropriate to overcome the identified asynchronism using AMP.

In the literature [37] it has been shown that stalls during the multimedia playback can lead to a degradation of the QoE. In order to avoid pauses or even skipping of multimedia content we proposed AMP for carrying out the synchronization at each peer individually (5).

The first subjective quality assessment using crowdsourcing shows that using content features for selecting the content section to which AMP is applied results in a better QoE than not taking content features into account **(6)**. We further provide measures that allow a quantification of the distortion introduced by AMP. Using these measures and a subjective quality assessment which provides some clues how the impact on the QoE and the introduced measures correlate, we derive a QoE utility model **(7)**.

In order to decide during the multimedia playback which content section shall be used for increasing or decreasing the playback rate we formulate a general optimization problem. For instantiating the optimization problem we use the introduced utility model as objective function. The optimization problem targets on finding the content section that minimizes the impact on the QoE with respect to the buffer fill state, the maximum duration of such a content section and the highest/lowest allowed playback rate **(8)**.



## 5.1 Introduction

This chapter discusses the applications that were developed during the course of the PhD project and the SocialSensor project [4]. The first application, the mobile DASHEncoder, was developed for generating MPEG-DASH compliant multimedia streams on mobile devices (especially on Android devices). In order to showcase the functionality of the mobile DASHEncoder we provide a live streaming application that generates on-the-fly MPEG-DASH compliant content for live streaming. We further investigate the energy consumption of generating MPEG-DASH compliant live multimedia streams on mobile devices. The second application that is discussed, is a web-based subjective quality assessment platform. This platform was extensively used to conduct the subjective quality assessments using crowdsourcing discussed in Chapter 4. This chapter is based on the following publications: [16], [17], and [18].

## 5.2 Mobile DASHEncoder

The amount of user-generated content and specifically (mobile) video is increasing significantly thanks to popular platforms like YouTube and Facebook. In most cases, users need to record the video on their mobile device and upload them using a Web interface or specific application installed on their mobile device. There are only a few tools and applications that allow an instant sharing of the video while recording [97, 98] but adopt mainly proprietary formats.

We propose an open source library that provides the transcoding and transmultiplexing of continuously recorded multimedia streams on mobile devices. The input format is flexible but the output format of the library is compliant to the MPEG-DASH standard [50]. The library enables the following features:

- output of multiple representations (bit-rate, resolution, etc.) which could utilize multiple cores (e.g., one representation per core);
- local storage and content upload to any remote server (e.g., HTTP server for further content provisioning);
- direct content sharing to other mobile devices (i.e., device-to-device communication without central server instance); and
- content upload as well instant sharing can be done adaptively following a given policy depending on the uplink bandwidth (or other context information).

We provide the possibility of transcoding and transmuxing the continuously recorded multimedia stream into MPEG-DASH compliant multimedia content [50]. The advantage of directly providing MPEG-DASH compliant multimedia content is that it can be directly offloaded from the mobile device to a HTTP Server and can be accessed without any further transcoding or transmuxing. Furthermore, our mobile DASHEncoder library is able to generate more than a single representation during live recording. This allows to react on the varying bandwidth conditions of mobile devices. The mobile DASHEncoder is developed for Linux, Windows, and Android devices (Android version 4.3). Nevertheless, it may operate using newer and older versions of Android.

Figure 5.1 depicts the components of the Semantic Middleware Suite of WP three of the SocialSensor project (cf. Section 1.1). The mobile DASHEncoder is a major part of the MyMedia component of the SocialSensor project. It closely interacts with the Semantic Service Coordination and the Semantic Data Retrieval components. It further represents an integral part of the Video Data Streaming component. The Semantic Service Coordination and Semantic Data Retrieval components provide the possibility of searching for live multimedia streams in P2P network [6].

Related open source software such as libdash [99] supports the consumption of multimedia content compliant to MPEG-DASH while MP4Box provides the possibility to generate MPEG-DASH compliant content [100]. Additionally, dash.js [101] and DASH-JS [102] enable the integration of MPEG-DASH within HTML5 environments, thanks to the Media Source Extensions (MSE).



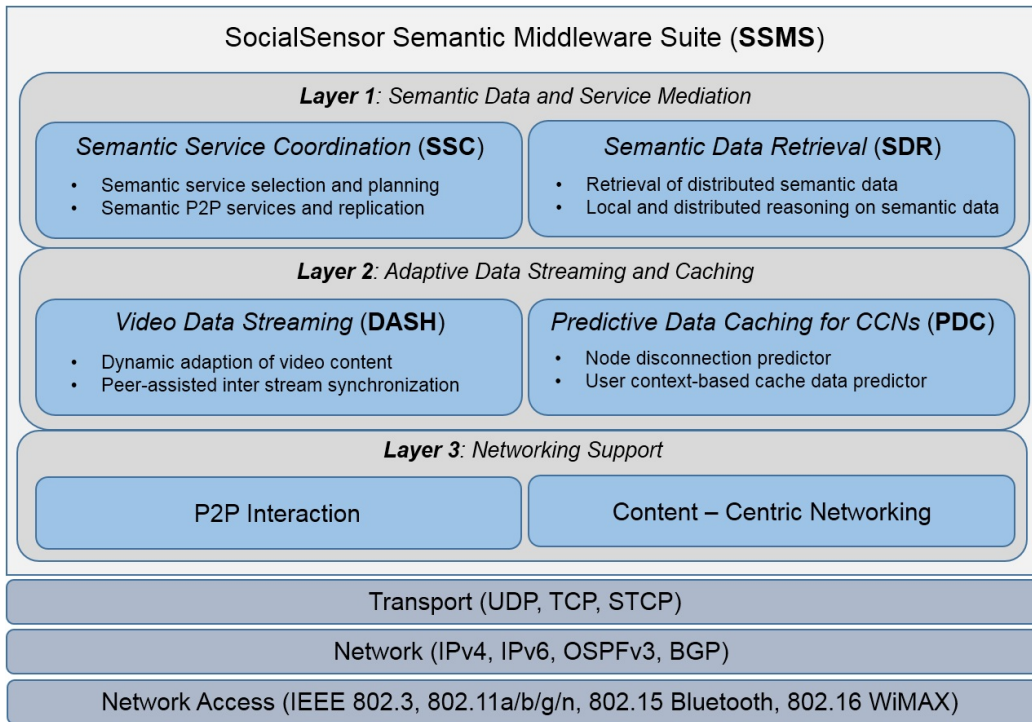


Figure 5.1: WP 3 components and modules of the SocialSensor Semantic Middleware Suite [6].

### 5.2.1 Architecture

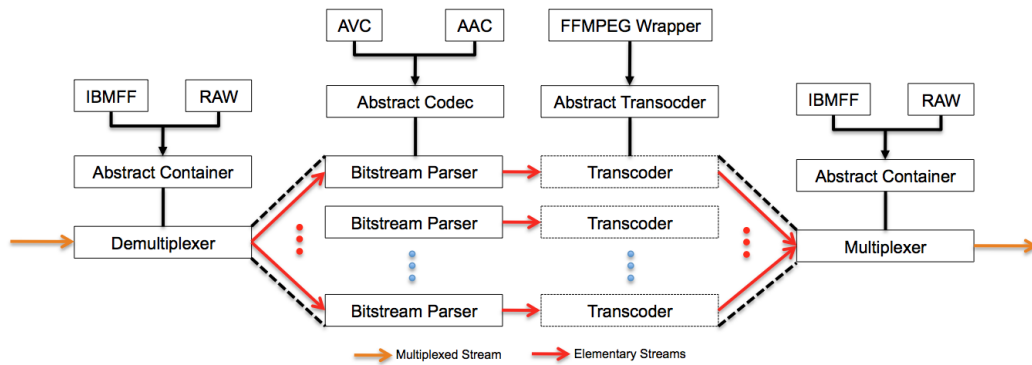


Figure 5.2: Architecture of the mobile DASHEncoder.

The architecture of our mobile DASHEncoder follows a pipe and filter architecture as depicted in Figure 5.2. It defines a filter graph comprising a *demultiplexer* and *multiplexer* at both ends and in between the elementary streams can be duplicated (to generate multiple representations) and transcoded as needed (to different bit-rates, resolutions, etc.). The mobile DASHEncoder is implemented using C/C++.

Each container has to implement the mandatory methods and functions as defined in the *abstract container* class which requires implementing the parsing, creation of the corresponding container format and whether the generated multiplexed multimedia stream is MPEG-DASH compliant. For instance, the ISO Base Media File Format (ISOBMFF) provides the possibility of fragmenting the multimedia content into equally sized time units. In the context of MPEG-DASH these units are referred to as segments which can be requested individually by the clients. Finally, the generation of the Media Presentation Description (MPD) is implicitly triggered when the *multiplexer* is instructed to generate MPEG-DASH compliant multimedia content.

The *bitstream parser* is responsible for extracting codec-specific information and for extracting the logical units (e.g., NAL units in the case of MPEG-AVC) from the demultiplexed elementary streams. Users of the library may duplicate elementary streams prior to the optional transcoding stage to enable flexibility with respect to the actual context conditions within the delivery environment. The actual *transcoder(s)* (re-)encode(s) the input elementary streams for the given configuration(s) provided by the actual application. For example, varying bandwidth conditions of the mobile device (e.g. due to user movements) can be compensated by instructing the mobile DASHEncoder library to adapt the bit-rate of the live recorded multimedia content in order to avoid stalls for peers consuming this live feed. If no transcoding is invoked the mobile DASHEncoder trans-multiplexes from the input container format to the output container format. Finally, the *demultiplexer* produces the MPEG-DASH segments according to standard supporting dynamically changing segment sizes (if needed).

Currently, we support the ISOBMFF as container format as well as MPEG-AAC and MPEG-AVC as audio/video codecs, respectively. However, the architecture is extensible for additional containers and codecs by inheriting the corresponding abstract classes. The current implementation of the library requires to manually edit the filter graph in the `transcode.cpp` file. The mobile DASHEncoder is available at GitHub [103] using the Lesser GNU Public License (LGPL) v3. The mobile DASHEncoder can be used on various platforms (e.g., Windows, Linux, Mac OS X) although it is optimized for the use with Android (version 4.3) as described in the next section.

## 5.2.2 Mobile DASHEncoder on Android

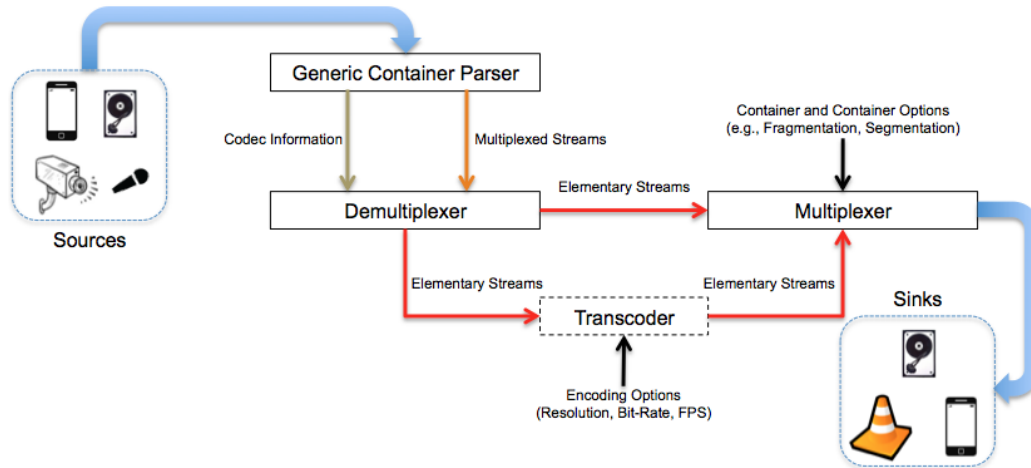


Figure 5.3: Usage scenario for the mobile DASHEncoder.

In Listing 5.1 we show how to provide a Java Native Interface (JNI) in order to use the mobile DASHEncoder with Java and, therefore, within an Android application.

```

1 #include "native_hook.h"
2 #include "mobile DASHEncoder/include/defs.h"
3 #include "mobile DASHEncoder/src/mobile DASHEncoder.h"
4 #define JNIFNDEFINE(fname) Java_itec_android_mobileDASHEncoder_NativeInterface_##fname
5 mobile DASHEncoder *mde = NULL;
6 extern "C" {
7     JNIEXPORT jint JNICALL JNIFNDEFINE(dumbStreamToFile)(JNIEnv* env, jclass cl,
8         jstring fileName, jobject inFD);
9     JNIEXPORT jint JNICALL JNIFNDEFINE(startNativeMobileDASHEncoder)(JNIEnv* env,
10        jclass cl, jstring setupFile, jstring baseName, jobject inFD, jobject bitrate, jint fps, jint
11        segmentLength);
12     JNIEXPORT jint JNICALL JNIFNDEFINE(stopNativeMobileDASHEncoder)(JNIEnv* env,
13        jclass cl);
14 };
15 int32_t grepNativeFdFromJava(JNIEnv *env, jclass cl, jobject fd)
16 {
17     jclass c;
18     jfieldID fid;
19     if(!(c = env->GetObjectClass(fd)) || !(fid = env->GetFieldID(c,"descriptor","I"))) return
20         _ERR;
21     return (int32_t)env->GetIntField(fd, fid);

```

```
17 }
18 JNIEXPORT jint JNICALL JNICALL(dumbStreamToFile)(JNIEnv *env, jclass cl, jstring
    fileName, jobject inFD)
19 {
20     int fd = grepNativeFdFromJava(env, cl, inFD);
21     if(fd == -1) return _NATIVE_FD_ERROR;
22     return 0;
23 }
24 JNIEXPORT jint JNICALL JNICALL(startNativeMobileDASHEncoder)(JNIEnv* env,
    jclass cl, jstring setupFile, jstring baseName, jobject inFD, jobject bitrate, jint fps, jint
    segmentLength)
25 {
26     char const *str;
27     const char *sbn;
28     int _fps;
29     int _segmentLength;
30     int fd = grepNativeFdFromJava(env, cl, inFD);
31     if(fd == -1) return _NATIVE_FD_ERROR;
32     str = env->GetStringUTFChars(setupFile, 0);
33     sbn = env->GetStringUTFChars(baseName, 0);
34     mde = new mobileDASHEncoder(fd, sbn, segmentLength, fps);
35     mde->start(str);
36     env->ReleaseStringUTFChars(setupFile, str);
37     env->ReleaseStringUTFChars(baseName, sbn);
38     return _NOERR;
39 }
40 JNIEXPORT jint JNICALL JNICALL(stopNativeMobileDASHEncoder)(JNIEnv* env,
    jclass cl)
41 {
42     if(mde!=NULL) delete mde;
43 }
```

Listing 5.1: JNI for the mobile DASHEncoder library.

Android's MediaRecorder API captures the video and audio by using the built-in camera and microphone, respectively. However, this API generates the necessary ISOBMFF boxes (which contain the decoding information) only at the end of the recording session. In order to overcome this shortcoming of the MediaRecorder API,

we suggest recording a short video sequence (a few milliseconds) prior to the actual recording to extract the necessary parameters (e.g., codec parameters) that will be used for the actual continuous recording. Listing 5.2 provides an example on how the mobile DASHEncoder can be used within a Java application using the MediaRecorder API.

```
1 ...
2 final Handler handler = new Handler();
3 mPreview = new RecorderWithPreview(this.getContext(), resX, resY, fps, bitrate, 200, "/
   sdcard/setup.mp4");
4 mPreview.setupCamera();
5 fmain.removeViewAt(0);
6 handler.postDelayed(new Runnable() {
7     public void run() {
8         mPreview.stop();
9         fmain.removeView(mPreview);
10        mPreview = new RecorderWithPreview(baseContext, resX, resY, fps, bitrate, 0,
        mediaRecorderLoop.getSenderFileDescriptor());
11        mPreview.setupCamera();
12        fmain.addView(mPreview,0);
13        final Handler handler = new Handler();
14        handler.postDelayed(new Runnable() {
15            public void run() {
16                mkdir("/sdcard/dash/live");
17                NativeInterface.NativeStartTranscoding("/sdcard/setup.mp4", "/sdcard/dash/live/live",
                mediaRecorderLoop.getReceiverFileDescriptor(), bitrate, fps, segmentSize);
18                mPreview.setupRecorder();
19                mPreview.start();
20            }
21        }, 500);
22    }
23 }, 1000);
24 ...
```

Listing 5.2: Example of using the mobile DASHEncoder.

Suppose that `RecorderWithPreview` is a class that deals with initializing and

setting up the MediaRecorder. The `mediaRecorderLoop` is a class that provides the redirection of file descriptors ( $fd_{in} \rightarrow fd_{out}$ ). First, we generate the file `setup.mp4` comprising a video sequence with a duration of 200 milliseconds in order to extract the necessary information for the actual continuous recoding. After the recording of this short video sequence we initialize the live recording with the same options except that we redirect the output of the MediaRecorder to the input of the mobile DASHEncoder (cf. Listing 5.2 line 10 and line 17). With the use of the `setup.mp4` the mobile DASHEncoder is able to generate MPEG-DASH compliant multimedia content using the ISOBMFF container with the provided segment size in seconds.

### 5.2.3 General Usage Scenarios

Figure 5.3 depicts a typical usage scenario of the mobile DASHEncoder. The input to the mobile DASHEncoder can be provided by devices that provides an elementary stream or a multiplexed stream, for example, built-in cameras, microphones or already stored multimedia content. In the case of the MediaRecorder which is provided by the Android API the input to the mobile DASHEncoder will be a multiplexed stream, as mentioned in the previous section. In Figure 5.3 we configured the filter graph of the mobile DASHEncoder to duplicate the elementary streams in order to provide two version of the provided multimedia content. First, we would like to provide a lower bit-rate version of the elementary streams in order to react on varying QoS parameters. Second, we always provide the highest representation (limited by the input bit-rate of each elementary stream). The high bit-rate content may be used for an on-demand scenario, where a high bit-rate representation is used to generate several representations that are spatially and/or temporally scaled.

For the transcoder a second instance of the hardware encoder or, for instance, FFMPEG [104] may be used. The hardware encoders of recent mobile devices provide the possibility of having two encoding chains at the same time with different encoding parameters. The *multiplexer* takes the elementary streams and packages them according to the selected container format with the provided options for making the output MPEG-DASH compliant. In particular, the application has to provide

the segment size in seconds and the location of the segments. We recommend using a lightweight HTTP server and the path to an external accessible Web folder which will contain the MPD according to the live profile. With the given MPD, any DASH-compliant player is able to access the multimedia content in a dynamic adaptive way.

#### 5.2.4 Energy consumption of MDE

For evaluation the power consumption of the mobile DASHEncoder when live video is recorded is measured using the Android tool PowerTutor2 [105]. The setup for measuring the energy consumption is as follows. We used a 22 inch monitor with a fixed brightness and contrast throughout all experiments. As test device we selected a Samsung Galaxy S3. The mobile device was mounted in front of the monitor such that the rear camera of the mobile device could fully capture the 22 inch monitor without its frame. Furthermore, we assured that the ambient lighting conditions were the same for every experiment because different ambient lighting may have an impact on the encoding of frames and, therefore, may have an impact on the energy consumption. As encoder we use the H.264/AVC hardware encoder, and tested resolutions of 240p, 360p, 720p and 1080p for bit-rates of 500 kbit/s and 1000 kbit/s.

In particular, we measured the energy consumption consumed by the H.264/AVC hardware encoder and by the mobile DASHEncoder during live recording.

We focus on the results of the resolution of 240p using a desired bit-rate of 500 kbit/s (cf. Figure 5.4) and on the results of the resolution of 1080p using a desired bit-rate of 1 Mbit/s (cf. Figure 5.5). The results of these two experiments depict the minimum and maximum of energy consumption in our experiments.

Figure 5.4 depicts the energy consumption for a resolution of 240p and a desired bit-rate of 500 kbit/s. With these settings the hardware encoder consumed 19.08  $J$ [Ws]. For generating a DASH compliant video stream the mobile DASHEncoder consumes 3.48 $J$  on average throughout all experiments because the effort of transmultiplexing does not increase with the resolution and desired output bit-rate (only the amount of data that is copied increases). The peak in the beginning represents the wind up of the camera and the mobile DASHEncoder. If we take into account

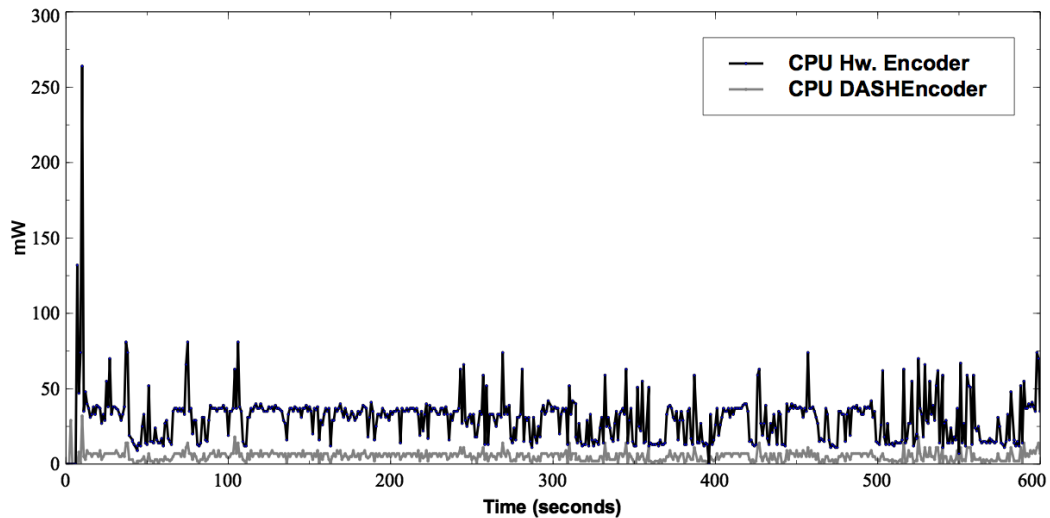


Figure 5.4: Energy consumption of live recording (240p, 500kbit/s)

that the battery of the Samsung S3 has 7.98  $Wh$ , the energy consumption of the mobileDASHEncoder results into the following maximum durations:

- 218.3h without the LCD display and WiFi and assuming that no other applications run in the background;
- 8.521h with the LCD display at full brightness;
- 4.82h with the LCD display at full brightness including that the WiFi module is at high state which corresponds to, i.e., streaming the recorded data;
- 3.388h with the LCD display at full brightness including that the WiFi module is at high state and assuming 700mW base power consumption which includes the camera and background applications [106].

Figure 5.5 depicts the energy consumption for a resolution of 1080p and a desired bit-rate of 1 Mbit/s. The energy consumed by the hardware encoder using 1080p and a bit-rate of 1 Mbit/s is 32.82  $J$ . Using 1080p and a bit-rate of 1 Mbit/s translates into a 1.7 times higher energy consumption than using a resolution of 240p and a bit-rate of 500 kbit/s. Again, the peak in the beginning represents the wind up of the camera and the mobile DASHEncoder. If we take into account that the battery of the Samsung S3 has 7.98  $Wh$ , the energy consumption of the mobileDASHEncoder



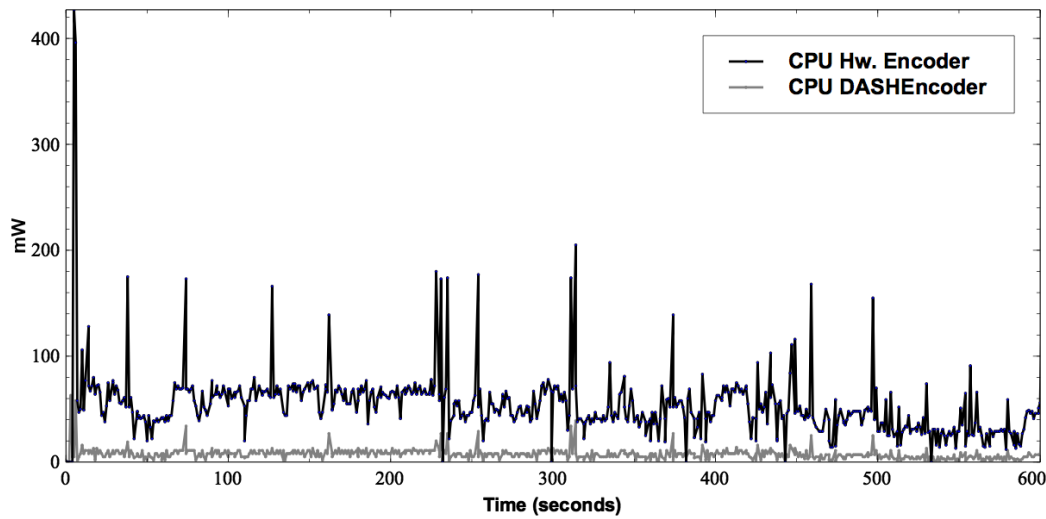


Figure 5.5: Energy consumption of live recording (1080p, 1000kbit/s)

results into the following maximum durations if we record with a resolution of 1080p and a bit-rate of 1 Mbit/s:

- 207.1h without the LCD display and WiFi and assuming that no other applications run in the background;
- 8.299h with the LCD display at full brightness;
- 4.748h with the LCD display at full brightness including that the WiFi module is at high state which corresponds to, i.e., streaming the recorded data;
- 3.354h with the LCD display at full brightness including that the WiFi module is at high state and assuming 700mW base power consumption which includes the camera and background applications [106].

### 5.2.5 Conclusion

We presented our open source mobile DASHEncoder which is easy to integrate and provides the possibility of generating MPEG-DASH compliant content. Due to its lightweight architecture it meets the strict delay constraints for live streaming. We have shown how the library can be integrated into Java applications, such as Android

Apps. The mobile DASHEncoder is freely available at [103] under the Lesser GNU Public License (LGPL).

The mobile DASHEncoder has been extensively used in the SocialSensor project, as already mentioned. It provides the possibility to stream currently recorded multimedia content. In SocialSensor it has been used to directly stream live footage to peers that are in the same P2P overlay network. In the context of this thesis the peer that provides the live multimedia content can be seen as the content provider. Therefore, the mechanisms introduced in Chapter 3 and Chapter 4 can be used to synchronize the playback of the peers streaming the live recorded multimedia content.

### 5.3 Web-based Assessment Platform for Subjective QoE Experiments

Planning, preparation, and conducting subjective quality assessments in the area of QoE is time consuming and in many cases an expensive task. The ITU provides recommendations on the methodology and design of subjective quality assessments for different use cases, e.g., multimedia applications or television pictures, but mainly for experiments in the lab [78][56] (cf. Section 2.5). In order to reduce the costs of laboratory subjective quality assessments, a new trend using crowdsourcing has evolved recently. For further details on crowdsourcing the interested readers is referred to Section 2.5 and Section 4.2.4 for details on using Microworkers, which is a crowdsourcing platform.

In [107], QualityCrowd is presented, a platform designed for Mechanical Turk. Another platform for conducting crowdsourced QoE assessments is presented in [66] which is tailored to use cases where pair comparison is the preferred choice of the evaluation method.

Our evaluation framework is based on the recommendation of the ITU for subjective quality evaluations of multimedia applications and television pictures [78][56]. The framework was mainly developed to conduct subjective quality assessments in

the domain of sensory effects and multimedia. It has been successfully used to conduct subjective quality assessments in these domains. For example, in [108], our platform was used to investigate the influence of sensory effects on the QoE and the emotional response. The platform was further used to conduct the subjective quality assessments using crowdsourcing discussed in Chapter 4.

The proposed evaluation platform requires an HTTP server with PHP support and a MySQL database. It can be integrated into any crowdsourcing platform as long as there is the possibility to embed external Web sites into a crowdsourced task. For example, both Mechanical Turk and Microworkers support this possibility. The evaluation framework is open source and available at [109].

### 5.3.1 Architecture

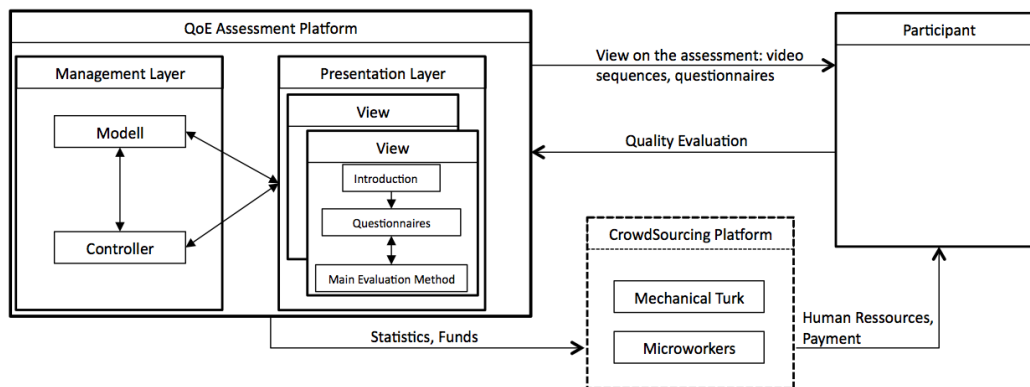


Figure 5.6: Overview of the Evaluation Framework.

The evaluation platform itself follows the Model-View-Controller (MVC) pattern. Figure 5.6 illustrates an overview of the proposed evaluation framework which was developed in HTML and PHP. The introduction and questionnaires can be configured separately from the test methodology and allow for including control questions during the main evaluation. These control questions can be randomly placed after stimulus presentation or at given points in time. The voting possibility can be configured independently from the test methodology; thus, providing more flexibility in selecting the appropriate voting mechanism and rating scale. The predefined voting mechanisms include the common HTML interface elements and some custom controls like a

slider in different variations. This offers more flexibility in the selection of the rating scale and the interface elements for voting possibilities in comparison to [107]. The platform consists of a management layer and a presentation layer. The management layer allows for maintaining the user study such as adding new questions or multimedia content and setting up the test method to be used (e.g., single stimulus, double stimulus, pair comparison, continuous quality evaluation). Therefore, this platform supports the most used test methods for conducting subjective quality assessments in comparison to [107] and [66], which support only a subset or in the case of [66], only a single test method. The presentation layer is responsible for presenting the content to the participants. This allows providing different views on the created study and, thus, one can define groups to which the participants may be randomly assigned (or in a defined way). After a participant has finished the user study, the gathered data is stored in the MySQL database. The platform provides means for an easy data analysis by developing PHP scripts, that read and analysis the data from the MySQL database.

Furthermore, the platform offers methods of tracking the participant's behavior during a subjective quality assessment by detecting if the Web browser's window has the focus and by calculating how much time each participant needed for each stimuli presentation and voting phase.

### 5.3.2 Introduction and Questionnaires

A generic version of the introduction which has been used in the subjective quality assessments discussed in Chapter 4 is provided by Description 5.1. The introduction explains the subjective quality assessment in detail and also asks the participant to agree on a disclaimer. The disclaimer is provided by Description 5.2.

We further collect some demographic data by employing a pre-questionnaire that is presented prior to the main evaluation. Figure 5.7 depicts the pre-questionnaire. The demographic data gathered is used to cross check whether the participants are really from the countries for which the subjective quality assessment was made available and for statistical analysis.

Thank you for your participation. In this experiment, we will show you two short video sequences with different temporal impairments. Each sequence is played X times with different playback adjustments. Please switch off all electrical devices (e.g., mobile phone) before reading the rest of the text and during the entire experiment. Furthermore, please put your browser into fullscreen mode (F11). Please **switch on your audio devices** and adjust the volume accordingly. Furthermore, disable all browser add-ons that may block JavaScript.

Your task is to evaluate your perceived quality of the viewing experience while watching the video sequences. The experiment comprises three parts: First, you will have to enter some general information about yourself (e.g., gender, age, country, nationality, occupational field). Second, the main evaluation will take place as described below. Third, a post experiment questionnaire will be shown to you.

The main evaluation procedure is as follows. First, a short training phase will be shown to you. The training phase is as follows: a selected video sequence will be shown to you, first without any temporal impairments and afterwards with a speed up in playback and a third time with slowing down the playback. After the training phase the main evaluation will start, where after each video sequence you will have 8 seconds for rating your perceived quality of experience. This is done by adjusting a slider on a continuous scale from 0 to 100, whereas, 0 indicates a very low quality of experience and 100 a very high quality of experience.

Your evaluation shall reflect your opinion. Please note, that the playback of the video sequences will start automatically.

With clicking the link "I agree & start the test" you agree that you are not visually impaired (glasses/contacts are okay) nor you have any impairments regarding hearing. Furthermore, you agree on the disclaimer which can be found **here**.

The overall time of the experiment will be around X minutes.

Description 5.1: Introduction presented to participants.

**Disclaimer****Epilepsy warning**

Please read before using the software provided by us. Some people are susceptible to epileptic seizures or loss of consciousness when exposed to certain flashing lights or light patterns in everyday life. Such people may have a seizure while watching certain videos with or without sensory effects. This may happen even if the person has no medical history of epilepsy or has never had any epileptic seizures. If you or anyone in your family has ever had symptoms related to epilepsy (seizures or loss of consciousness) when exposed to flashing lights, fans generating wind or vibrations, consult your doctor prior to participating in the test. If you experience any of the following symptoms: dizziness, blurred vision, eye or muscle twitches, loss of consciousness, disorientation, any involuntary movement convulsion, while watching videos, immediately discontinue use and consult your doctor.

**Content**

The author takes no responsibility for the completeness or quality of the information provided. Liability claims regarding damage caused by the use of any information or software provided, including any kind of information or software which is incomplete or incorrect, will therefore be rejected.

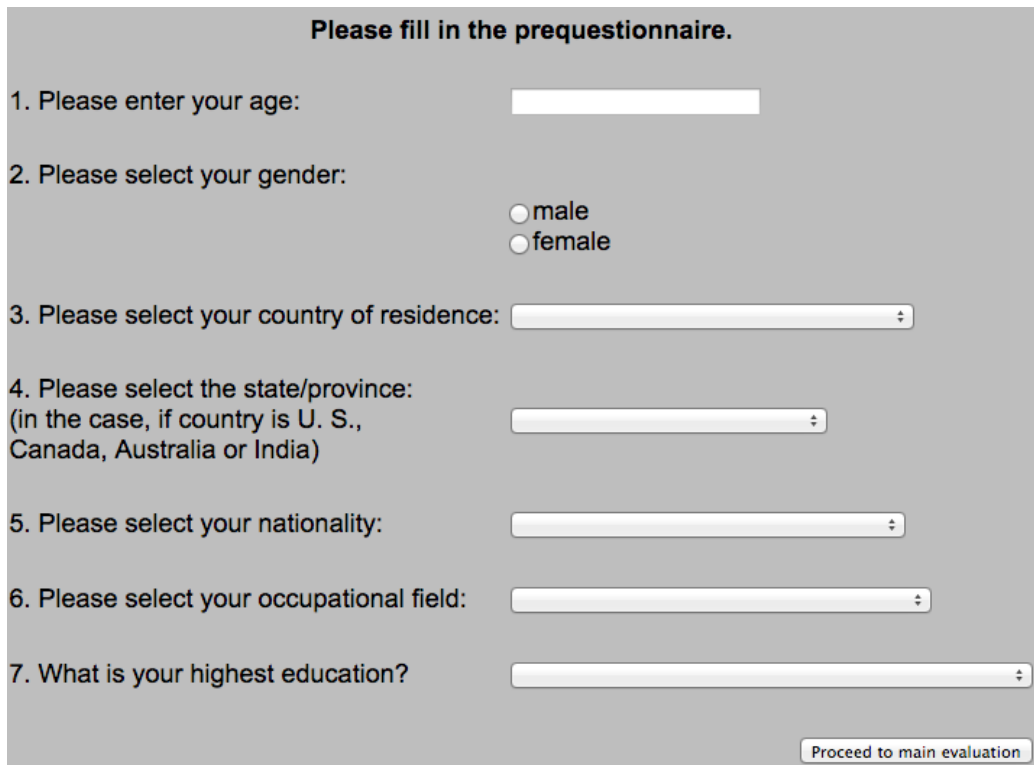
**Copyright**

The copyright for any material created by the author is reserved. Any duplication or use of objects such as software, electronic or printed publications is not permitted without the author's agreement.

**Legal validity of this disclaimer**

This disclaimer is to be regarded as part of the Internet publication which you were referred from. If any section or terms of this statement are not legal or incorrect, the validity and content of the other parts remain uninfluenced by this fact.

Description 5.2: Disclaimer to which the participants have to agree.



**Please fill in the prequestionnaire.**

1. Please enter your age:
2. Please select your gender:  
 male  
 female
3. Please select your country of residence:
4. Please select the state/province:  
(in the case, if country is U. S.,  
Canada, Australia or India)
5. Please select your nationality:
6. Please select your occupational field:
7. What is your highest education?

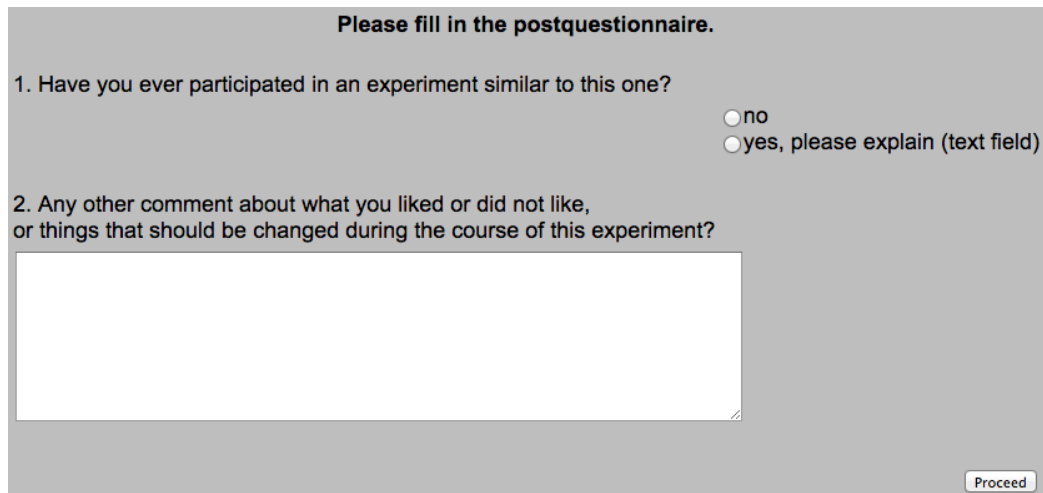
[Proceed to main evaluation](#)

Figure 5.7: Pre-Questionnaire for collecting some demographic data.

In order to assess whether participants have already participated in a similar experiment/study we provide a post questionnaire, depicted by Figure 5.8. The post-questionnaire further provides the possibility of providing free text feedback to the researchers.

### 5.3.3 Stimuli Presentation and Rating Possibility

The stimuli presentation can be configured independently from the test method. The stimuli presentation may be combined with the voting possibility to support continuous quality evaluations. Furthermore, plug-ins may be embedded for laboratory experiments where external hardware is used. Multimedia content is presented by using either the HTML5 video element or the Adobe Flash Player. The decision which technique is used to render multimedia can be set through the management layer if one of them is needed explicitly. Otherwise, the decision is based on the codecs used and which of them does the client's Web browser support. Furthermore, there



**Please fill in the postquestionnaire.**

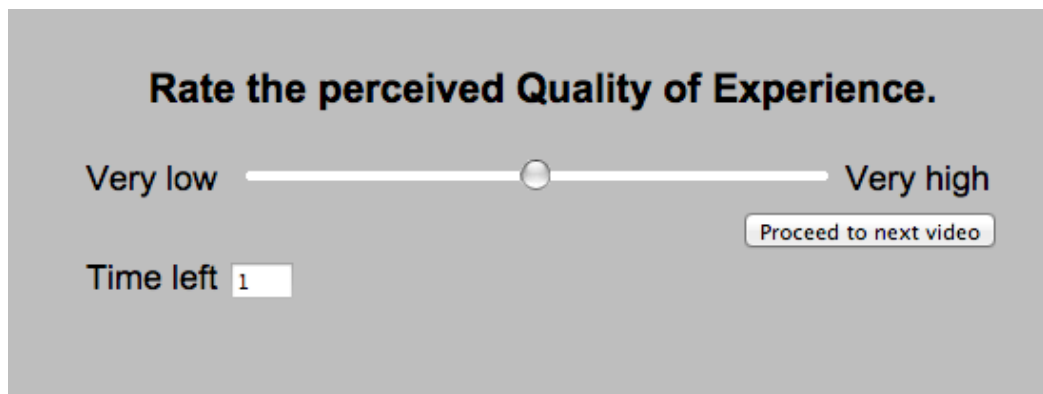
1. Have you ever participated in an experiment similar to this one?

no  
 yes, please explain (text field)

2. Any other comment about what you liked or did not like, or things that should be changed during the course of this experiment?

Proceed

Figure 5.8: Post-Questionnaire providing the possibility of giving feedback.



**Rate the perceived Quality of Experience.**

Very low  Very high

Proceed to next video

Time left

Figure 5.9: Slider for rating the QoE.

is support for adding JavaScript to alter the presentation of the stimuli or for any other purpose.

For the subjective quality assessments presented in Chapter 4 we have used a slider with a scale of  $[0, 100]$  with a stepping of *one*. Figure 5.9 depicts the representation of the slider in our web-based assessment platform. The maximum time for providing a rating can be deactivated. Nevertheless, the ITU recommends to restrict the time a participant has for providing a rating [56][78].



### 5.3.4 Conclusion

The presented subjective quality evaluation framework has been successfully used to conduct several studies in-lab and using crowdsourcing. In the context of this thesis it has been used to conduct the subjective quality assessments for AMP (cf. Chapter 4).

The platform reduces the time needed for designing and conducting subjective quality assessments. the framework can be easily extended and can be adapted to meet the requirements for a subjective quality assessment in the field of multimedia. Additionally, with the use of HTML5 and the Adobe Flash Player it supports a broad range of codecs on various browsers.

The evaluation framework has been successfully used in conjunction with research not covered by this thesis. For example, in [108] for assessing the QoE and the emotional response to video sequences enriched with sensory effects. Furthermore, it has been used in [110] where the data gathered through a subjective quality assessment has been used to derive a QoE utility model for sensory effects.



## 6.1 Summary

The ultimate goal of this thesis has been to research and develop mechanisms and technologies that provide IDMS in the context of pull-based streaming, in particular MPEG-DASH. In this final chapter we summarize the research contributions towards this goal. In the previous chapters, we have presented the mandatory parts of an IDMS system, evaluated them and have shown their advantages compared to related work. We have defined the notion of an IDMS session and introduced a session management for IDMS to MPEG-DASH. For signaling control, timing information and to agree on a reference playback timestamp among a group of peers in the same IDMS session we have introduced a two phase synchronization protocol. We further investigated how to carry out the actual synchronization and its impact on the QoE.

First, we have shown that MPEG-DASH provides the possibility to enrich the MPD with session information for achieving IDMS. For agreeing on a reference playback timestamp and, thereafter, to identify the asynchronism at each peer we introduced a two phase synchronization protocol. The first phase creates P2P a overlay and tries to make an educated guess at which segment a specific peer has to start. For the second phase, we introduced a distributed algorithm called *Merge and Forward*, that allows the peers in an IDMS session to agree on a reference playback timestamp. This algorithm was initially designed to be used in conjunction with MPEG-DASH but it can be easily decoupled because it does not rely on MPEG-DASH specific information. We have proven that *Merge and Forward* always calculates the average playback timestamp, regardless of how well the overlay P2P network is connected (unless it is not disconnected). *Merge and Forward* uses a constant size message structure which only changes if the selected size of the Bloom filter is too small in

order to map all peers in the overlay network into it without encountering false positives. We compared the introduced distributed algorithm to an existing approach from related work.

Second, we investigated how AMP can be utilized to carry out the actual synchronization. We have shown that content features can be used to find content sections that allow to hide these playback rate variations such that the QoE is not decreased significantly. Therefore, we have conducted two subjective quality assessments. The first to provide evidence that content section identified by content features provide a better QoE than randomly selecting them. The second subjective quality assessment has been conducted in order to provide a utility model for AMP and for introducing our distortion measures (and semi-metrics) that allow to quantify the impact of such playback rate variations on the QoE. The derived utility model has been further used as objective function for defining dynamic AMP. Dynamic AMP aims on finding content sections (under certain constraints) such that the synchronization can be carried out with the smallest possible impact on the QoE.

## 6.2 Findings

The research objectives presented in Section 1.2 have been covered as follows:

**(1) to define the notion of an IDMS session.**

We have defined an IDMS session in the context of MPEG-DASH by means of a XML schema. This definition is not coupled to MPEG-DASH and can be used with any streaming technology. The ISO does not solely define the timeliness of such a session but, it further describes which peers are currently participating in the session. In order to uniquely identify the peers the IP address and the corresponding port of the peer is stored in the ISO. The ISO is only retrieved once by a peer that joins an IDMS session and, therefore, the traffic between the entity that maintains the ISOs and the actual peers is reduced to a minimum.

**(2) to introduce session management for IDMS in the context of MPEG-DASH.**

Session management has been introduced by extending the MPD with the ISO and by the coarse synchronization. The coarse synchronization creates the P2P overlay network. If a peer joins an IDMS session it receives the ISO that lists peers that are already in the corresponding IDMS session. The peer enters the already existing P2P overlay by testing which of these peer can be contacted and asks them for their current PTS. This allows the peer to make an educated guess with which segment it shall start the playback of the multimedia content. Therefore, the session management in our approach is carried out by each peer itself and there is no central entity needed that separately handles peers that join and leave an IDMS sessions.

- (3) to provide an algorithm that identifies the asynchronism between the multimedia playback of different users in an IDMS session in a distributed and self-organized manner.**

We have introduced a distributed and self-organized algorithm called *Merge and Forward* which allows the peers within a P2P overlay network to agree on a reference playback timestamp. For the desired reference playback timestamp we have chosen the average playback timestamp among all peers in the P2P overlay network. We employ Bloom filters which are a probabilistic data structure for tracking which peers have already contributed to reference playback timestamp. *Merge and forward* is also able to detect false positive which may be encountered when using standard Bloom filters. We have proven that the calculation of the reference playback timestamp converges to the average playback timestamp even if the overlay network changes.

- (4) to evaluate the introduced distributed and self-organized approach.**

We compared *Merge and Forward* to an algorithm that has been introduced in related work. The simulation results stated that *Merge and Forward* provides always lesser overhead. We further show how the connectivity of the P2P overlay and the number of peers in the overlay network translate into overhead generated by the algorithms. We further investigate how long it takes the algorithms until all peer hold the final reference playback timestamp. If we do not restrict the bandwidth of the peers, the algorithm taken from related work

performs always optimal. But, with restricted bandwidths *Merge and Forward* performs better because it does not consume that much bandwidth of each peer and, therefore, reserves bandwidth for the actual multimedia streaming.

- (5) to investigate the possibility of carrying out the synchronization in terms of adaptively changing the media playback by increasing or decreasing the playback rate of the multimedia playback.**

Using AMP for carrying out the actual synchronization has been already proposed in [38]. Previous works (cf. Section 2) have shown that pausing or stalling the playback of multimedia content decreases the QoE [37]. AMP is always applied to a certain section of the multimedia content. We precisely defined a content section by Definition 4. Increasing or decreasing the multimedia playback rate at the time when a peer receives the final reference playback timestamp is equal to randomly increasing or decreasing the playback rate because the peers may have completely different PTSs. This random selection of content sections encouraged us trying to find content section for which the QoE does not decrease significantly. Therefore, we investigated if and which content features are appropriate for the selection of content sections.

- (6) to assess whether content features allow to decrease the impact of increasing or decreasing the playback rate on the QoE.**

In order to investigate if content features may help in finding content sections we introduced the spectral energy of audio frames and the motion intensity of consecutive video frames as content features for audio and video, respectively. We introduce an algorithm that tries to identify content section using these content features. We compare the introduced algorithm to randomly selecting content section for overcoming a given asynchronism by increasing or decreasing the playback rate by conducting a subjective quality assessment using crowdsourcing. The results clearly state that those content sections selected using content features provide a significantly better QoE than randomly selecting content sections. This difference in the QoE becomes even greater if we take a look at the extreme cases where we reduced the playback rate to half the nominal playback rate and increased the playback rate to twice the nominal playback rate.

**(7) to analyze and quantify the impact of AMP on the QoE.**

From the first subjective study we have learned that the selected content features can be used to select content sections that provide a higher QoE than just randomly selecting them. We then introduced for each of the features a distortion measures that shall allow to quantify the impact of AMP on the QoE. Therefore, we conducted a subjective quality assessment using crowdsourcing for which select specific content sections and presented them with different playback rates. We further computed the distortion measures for each of these content features and derived a utility model that allows to estimate the QoE by using these distortion measures.

**(8) to utilize content features for dynamically selecting content sections that are appropriate to overcome the identified asynchronism using AMP.**

With the defined distortion semi-metrics and measures for audio and video, and the introduced utility model we propose a constrained optimization problem that aims on finding content section that minimize the impact on the QoE when increasing or decreasing the playback rate. The utility model is used as objective function. The proposed constraints allow to restrict the selection of the content section such that a certain buffer fill state is not under-run. The duration, and the playback rate used to overcome a given asynchronism depend on each other. Therefore, we reduce the problem from a three dimensional space to a two dimensional space. We further show how the constrained optimization problem can be solved. The resulting content section provide the highest QoE in terms of the defined distortion semi-metrics.

### 6.3 Future Work

Identifying the asynchronism between a group of peers using *Merge and Forward* provides a very low overhead compared to state of the art algorithms. Nevertheless, *Merge and Forward* increases its Bloom filter if it encounters false positives. This increases the time until all peers have agreed on the reference playback timestamp

(cf. Section 3.4.2). A possible future work item is to investigate the trade-off of allowing false positives which will result in a deviation of the reference playback timestamp to the real average playback timestamp.

In this thesis we have covered the case that the consumed multimedia content is the same for all participants. This assumption allows us to use the PTS only for agreeing on the reference playback timestamp. If we assume that the multimedia content is present in different encodings agreeing on a reference playback timestamp becomes even more complex. Different encodings may lead to different PTS for the same audio or video frame and the injection of commercials may introduce another additive factor to the PTS. Therefore, the assumption that peer  $a$  displays the same frame at second 15 as peer  $b$  does not hold anymore. A possible solution to this would be to exchange the fingerprint of the audio and video frame that is displayed at the corresponding PTS. This implies that there exists a metric that allows to calculate the distance between the fingerprints for audio and video in order to calculate the offset to the corresponding PTS at each peer. We declare this problem to future work.

For carrying out the synchronization using AMP we have introduced a constrained optimization problem which utilizes the defined distortion metrics and the utility model for estimating the impact on the QoE. We did not look into how the actual playback rate curve looks like. We have assumed an instant change (step-wise function) to the increased or decreased playback rate. This rapid change may have an impact on the QoE and, therefore, for future work it would be interesting to evaluate how different adjustment curves of the playback rate effect the QoE. For instance the playback rate may be altered using a polynomial (quadratic, cubic) in order to overcome the identified asynchronism. This could even further increase the performance of AMP for IDMS.



# Bibliography

- [1] D. Geerts, I. Vaishnavi, R. Mekuria, O. van Deventer, and P. Cesar, “Are we in Sync?: Synchronization Requirements for Watching Online Video together,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 311–314, ACM, 2011.
- [2] B. Ngamwongwattana and R. Thompson, “Measuring One-way Delay of VoIP Packets without Clock Synchronization,” in *Instrumentation and Measurement Technology Conference*, pp. 532–535, IEEE, 2009.
- [3] V. Rodoplu, S. Vadvalkar, A. Gohari, and J. Shynk, “Empirical Modeling and Estimation of End-to-End VoIP Delay over Mobile Multi-Hop Wireless Networks,” in *Global Telecommunications Conference*, pp. 1–6, IEEE, 2010.
- [4] SocialSensor European Project, “<http://www.socialsensor.eu>,” Last Accessed October 2014.
- [5] ISO/IEC 32009-1:2014, “Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats,” 2014.
- [6] M. Klusch, P. Kapahnke, B. Rainer, J. Guerts, and C. Timmerer, “D3.3 - Semantic Middleware Suite,” in *FP7 SocialSensor EU*, 2014.
- [7] Google Hangouts, “<https://plus.google.com/hangouts>,” Last Accessed October 2014.
- [8] WebRTC, “<http://www.webrtc.org/>,” Last Accessed October 2014.
- [9] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RFC 3550: RTP: A Transport Protocol for Real-Time Applications,” tech. rep., IETF, 2003.

- 
- [10] F. Boronat, J. Lloret, and M. García, “Multimedia Group and Inter-stream Synchronization Techniques: A Comparative Study,” *Information Systems*, vol. 34, no. 1, pp. 108–131, 2009.
- [11] B. Rainer and C. Timmerer, “Self-Organized Inter-Destination Multimedia Synchronization for Adaptive Media Streaming,” in *Proceedings of the 22st ACM International Conference on Multimedia*, pp. 327–336, ACM, 2014.
- [12] B. Rainer and C. Timmerer, “Merge And Forward - Self-organized Inter-Destination Multimedia Synchronization,” in *Proceedings of the 6th International Conference on Multimedia Systems*, pp. 77–80, ACM, March 2015.
- [13] B. Rainer and C. Timmerer, “Adaptive Media Playout for Inter-Destination Media Synchronization,” in *Proceedings of the 5th International Workshop on Quality of Multimedia Experience*, pp. 44–45, IEEE, 2013.
- [14] B. Rainer and C. Timmerer, “A Subjective Evaluation using Crowdsourcing of Adaptive Media Playout utilizing Audio-Visual Content Features,” in *IEEE/I-FIP 2nd International Workshop on Quality of Experience Centric Management*, pp. 1–7, IEEE, 2014.
- [15] B. Rainer and C. Timmerer, “A Quality of Experience Model for Adaptive Media Playout,” in *Sixth International Workshop on Quality of Multimedia Experience*, pp. 177–182, IEEE, 2014.
- [16] M. Klusch, P. Kapahnke, X. Cao, B. Rainer, C. Timmerer, and S. Mangold, “MyMedia: Mobile Semantic Peer-to-Peer Video Search and Live Streaming,” in *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pp. 277–286, ACM, 2014.
- [17] B. Rainer, M. Waltl, and C. Timmerer, “A Web based Subjective Evaluation Platform,” in *Proceedings of the 5th International Workshop on Quality of Multimedia Experience*, pp. 24–25, IEEE, 2013.
- [18] B. Rainer, C. Timmerer, and M. Waltl, “Recommendations for the Subjective Evaluation of Sensory Experience,” in *4th International Workshop on Perceptual Quality of Systems 2013*, p. 4, IEEE, 2013.

- 
- [19] G. Blakowski and R. Steinmetz, "A Media Synchronization Survey: Reference Model, Specification, and Case Studies," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 1, pp. 5–35, 1996.
- [20] M. Claypool and J. Tanner, "The Effects of Jitter on the Peceptual Quality of Video," in *Proceedings of the Seventh ACM International Conference on Multimedia*, pp. 115–118, ACM, 1999.
- [21] R. Steinmetz, "Human Perception of Jitter and Media Synchronization," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 61–72, Jan 1996.
- [22] Y.-F. Ou, Y. Zhou, and Y. Wang, "Perceptual Quality of Video with Frame Rate Variation: A subjective study," in *International Conference on Acoustics Speech and Signal Processing*, pp. 2446–2449, IEEE, 2010.
- [23] M. Montagud, F. Boronat, H. Stokking, and R. Brandenburg, "Inter-destination Multimedia Synchronization: Schemes, Use Cases and Standardization," *Multimedia Systems*, vol. 18, pp. 459–482, 2012.
- [24] D. L. Mills, "Internet Time Synchronization: the Network Time Protocol," *IEEE Transactions on Communications*, vol. 39, pp. 1482–1493, 1991.
- [25] *IEEE 1588 Precision Time Protocol on Wireless LAN Software and Hardware Prototypes*, 2005.
- [26] Y. Ishibashi and S. Tasaka, "A Distributed Control Scheme for Causality and Media Synchronization in Networked Multimedia Games," in *Proceedings of the 11th International Conference on Computer Communications and Networks*, pp. 144–149, IEEE, 2002.
- [27] Y. Ishibashi, T. Hasegawa, and S. Tasaka, "Group Synchronization Control for Haptic Media in Networked Virtual Environments," in *12th International Symposium HAPTICS.*, pp. 106–113, 2004.
- [28] Y. Ishibashi, A. Tsuji, and S. Tasaka, "A group synchronization mechanism for stored media in multicast communications," in *in Proceedings of 16th Annual*

- Joint Conference of the Computer and Communications Societies. Driving the Information Revolution (INFOCOM).*, vol. 2, pp. 692–700, IEEE, 1997.
- [29] H. Stokking, M. Van Deventer, O. Niamut, F. Walraven, and R. Mekuria, “IPTV Inter-destination Synchronization: A Network-based Approach,” in *in Proceedings of the 14th Intelligence in Next Generation Networks (ICIN)*, pp. 1–6, IEEE, 2010.
- [30] M. Mauve, J. Vogel, V. Hilt, and W. Effelsberg, “Local-lag and Timewarp: Providing Consistency for Replicated Continuous Applications,” 2002.
- [31] F. Boronat Seguí, J. Guerri Cebollada, and J. Lloret Mauri, “An RTP/RTCP based Approach for Multimedia Group and Inter-stream Synchronization,” *Multimedia Tools and Applications*, vol. 40, pp. 285–319, 2008.
- [32] M. Montagud, “Design, Development and Evaluation of an Adaptive and Standardized RTP/RTCP-based IDMS Solution,” in *Proceedings of the 21st International Conference on Multimedia*, pp. 1071–1074, ACM, 2013.
- [33] R. Brandenburg, H. Stokking, O. Deventer, F. Boronat, M. Montagud, and K. Gross, “RFC 7272: Inter-Destination Media Synchronization (IDMS) Using the RTP Control Protocol (RTCP),” tech. rep., IETF, 2014.
- [34] C. Hesselman, D. Abbadessa, W. Van Der Beek, D. Gorgen, K. Shepherd, S. Smit, M. Gulbahar, I. Vaishnavi, J. Zoric, D. Lowet, R. De Groote, J. O’Connell, and O. Friedrich, “Sharing Enriched Multimedia Experiences Across Heterogeneous Network Infrastructures,” *IEEE Communications Magazine*, vol. 48, no. 6, pp. 54–65, 2010.
- [35] M. Montagud, F. Boronat, and H. Stokking, “Design and Simulation of a Distributed Control Scheme for Inter-destination Media Synchronization,” in *Proceedings of 27th International Conference on Advanced Information Networking and Applications*, pp. 937–944, IEEE, 2013.
- [36] F. Boronat, M. Montagud, and V. Vidal, “Master Selection Policies for Inter-destination Multimedia Synchronization in Distributed Applications,” in *19th*

- International Symposium on Modeling Analysis and Simulation of Computer and Telecommunication Systems*, pp. 269–277, IEEE, 2011.
- [37] T. Hossfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, “Quantification of YouTube QoE via Crowdsourcing,” in *International Symposium on Multimedia*, pp. 494–499, IEEE, 2011.
- [38] M. Montagud and F. Boronat, “On the Use of Adaptive Media Payout for Inter-Destination Synchronization,” *IEEE Communications Letters*, vol. 15, no. 8, pp. 863–865, 2011.
- [39] M. Yuang, S. T. Liang, Y. Chen, and C. Shen, “Dynamic Video Payout Smoothing Method for Multimedia Applications,” in *Proceedings of the International Conference on Converging Technologies for Tomorrow’s Applications*, vol. 3, pp. 1365–1369, IEEE, Jun 1996.
- [40] S. Meyn, R. Tweedie, and P. Glynn, *Markov Chains and Stochastic Stability*. Cambridge Mathematical Library, Cambridge University Press, 2009.
- [41] M. K. Eckehard, M. Kalman, E. Steinbach, and B. Girod, “Adaptive Payout For Real-Time Media Streaming,” in *Proceedings of the International Conference on Image Processing*, pp. 45–48, IEEE, 2002.
- [42] ITU-T Recommendation J.340, “Series J: Cable Networks and Transmission of Television, Sound Programme and other Multimedia Signals,” tech. rep., International Telecommunication Union, 2010.
- [43] M. Kalman, E. Steinbach, and B. Girod, “Rate-distortion Optimized Video Streaming with Adaptive Payout,” in *Proceedings of the International Conference on Image Processing*, vol. 3, pp. 189–192, IEEE, 2002.
- [44] Y.-F. Su, Y.-H. Yang, M.-T. Lu, and H. H. Chen, “Smooth Control of Adaptive Media payout for Video Streaming,” *Transactions on Multimedia*, vol. 11, pp. 1331–1339, Nov. 2009.

- 
- [45] M. Kalman, E. Steinbach, and B. Girod, "Adaptive Media Playout for Low-delay Video Streaming over Error-prone Channels," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 841–851, 2004.
- [46] H.-C. Chuang, C. Huang, and T. Chiang, "Content-Aware Adaptive Media Playout Controls for Wireless Video Streaming," *IEEE Transactions on Multimedia*, vol. 9, no. 6, pp. 1273–1283, 2007.
- [47] C.-H. Liang and C.-L. Huang, "Content-based Adaptive Media Player for Networked Video," in *Proceedings of the 2004 International Symposium on Circuits and Systems*, vol. 3, pp. 749–752, 2004.
- [48] Z. Lu, W. Lin, B. C. Seng, S. Kato, E. Ong, and S. Yao, "Perceptual Quality Evaluation on Periodic Frame-Dropping Video," in *Proceedings of the International Conference on Image Processing*, vol. 3, pp. 433–436, IEEE, 2007.
- [49] Q. Huynh-Thu and M. Ghanbari, "Perceived quality of the variation of the video temporal resolution for low bit rate coding," in *Picture Coding Symposium*, IEEE, 2007.
- [50] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, 2011.
- [51] R. Pantos and W. May, "HTTP Live Streaming, draft-pantos-http-live-streaming-07," tech. rep., IETF, 2011.
- [52] Microsoft Smooth Streaming, "<http://www.iis.net/downloads>," Last Accessed October 2014.
- [53] Adobe HTTP Dynamic Streaming, "<http://www.adobe.com/products/hds-dynamic-streaming.html>," Last Accessed October 2014.
- [54] ISO/IEC 14496-12:2005, "Information technology – Coding of Audio-Visual Objects – Part 12: ISO Base Media File Format," 2005.
- [55] 13818-1:2013, "Information technology – Generic Coding of Moving Pictures and Associated Audio Information – Part 1: Systems," 2013.

- 
- [56] ITU-T Recommendation P.910, “Subjective Video Quality Assessment Methods for Multimedia Applications,” tech. rep., International Telecommunication Union, 2008.
- [57] ITU-R Recommendation BT.500-13, “Methodology for the Subjective Assessment of the Quality of Television Pictures,” tech. rep., International Telecommunication Union, 2012.
- [58] D. R. Hunter, “Mm algorithms for generalized bradley-terry models,” *The Annals of Statistics*, vol. 32, no. 1, pp. 384–406, 2004.
- [59] M. Turk, “<https://www.mturk.com/mturk/>,” Last Accessed January 2015.
- [60] Microworkers, “<http://www.microworkers.com>,” Last Accessed January 2015.
- [61] T. Hossfeld, C. Keimel, M. Hirth, B. Gardlo, J. Habigt, K. Diepold, and P. Tran-Gia, “Best Practices for QoE Crowdstesting: QoE Assessment with Crowdsourcing,” *IEEE Transactions on Multimedia*, no. 99, pp. 541–558, 2013.
- [62] C.-C. Wu, K.-T. Chen, Y.-C. Chang, and C.-L. Lei, “Crowdsourcing Multimedia QoE Evaluation: A Trusted Framework,” *IEEE Transactions on Multimedia*, vol. 15, no. 5, pp. 1121–1137, 2013.
- [63] F. Ribeiro, D. Florencio, V. Nascimento, “Crowdsourcing Subjective Image Quality Evaluation,” in *Proceedings of the International Conference on Image Processing*, pp. 3097–3100, IEEE, 2011.
- [64] F. Ribeiro, D. Florencio, C. Zhang, M. Seltzer, “CROWDMOS: An approach for Crowdsourcing Mean Opinion Score Studies,” in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 2416–2419, IEEE, 2011.
- [65] C. Keimel, J. Habigt, and K. Diepold, “Challenges in Crowd-based Video Quality Assessment,” in *Proceedings of the Fourth International Workshop on Quality of Multimedia Experience*, pp. 13–18, IEEE, 2012.

- [66] K.-T. Chen, C.-J. Chang, C.-C. Wu, Y.-C. Chang, and C.-L. Lei, “Quadrant of Euphoria: a Crowdsourcing Platform for QoE Assessment,” *IEEE Network*, vol. 24, no. 2, pp. 28–35, 2010.
- [67] Network Working Group, “RFC 5389 - Session Traversal Utilities for NAT (STUN),” tech. rep., IETF, 2008.
- [68] Network Working Group, “RFC 5766 - Traversal Using Relays around NAT (TURN),” tech. rep., IETF, 2010.
- [69] K. Kuramochi, T. Kawamura, and K. Sugahara, “NAT Traversal for Pure P2P e-Learning System,” in *Proceedings of the 3rd International Conference on Internet and Web Applications and Services*, pp. 358–363, IARIA, 2008.
- [70] K. Christensen, A. Roginsky, and M. Jimeno, “A New Analysis of the False Positive Rate of a Bloom Filter,” *Information Processing Letters, Elsevier North-Holland*, vol. 110, no. 21, pp. 944–949, 2010.
- [71] P. Erdős and A. Rényi, “On Random Graphs,” *Publicationes Mathematicae (Debrecen)*, vol. 6, pp. 290–297, 1959.
- [72] OMNET++ 4.3.1, “<http://www.omnetpp.org/>,” Last Accessed December 2013.
- [73] B. Rainer, S. Lederer, C. Mueller, and C. Timmerer, “A Seamless Web Integration of Adaptive HTTP streaming,” in *Proceedings of the 20th European Signal Processing Conference*, pp. 1519–1523, European Signal Processing (EURASIP) Society, 2012.
- [74] Big Buck Bunny, “<http://www.bigbuckbunny.org/>,” Last Accessed March 2014.
- [75] Y. Li, A. Markopoulou, J. Apostolopoulos, and N. Bambos, “Content-Aware Payout and Packet Scheduling for Video Streaming Over Wireless Links,” *IEEE Transactions on Multimedia*, vol. 10, no. 5, pp. 885–895, 2008.
- [76] J. Redi, T. Hossfeld, P. Korshunov, F. Mazza, I. Pova, and C. Keimel, “Crowdsourcing-based Multimedia Subjective Evaluations: A Case Study on



- Image Recognizability and Aesthetic Appeal,” in *Proceedings of the 1st International Workshop on Crowdsourcing for Multimedia*, pp. 29–34, ACM, 2013.
- [77] J. You, U. Reiter, M. M. Hannuksela, M. Gabbouj, and A. Perkis, “Perceptual-based quality assessment for audio–visual services: A survey,” *Signal Processing: Image Communication*, vol. 25, no. 7, pp. 482–501, 2010.
- [78] “Rec. ITU-R BT.500-11,” tech. rep.
- [79] M. P. Couper, R. Tourangeau, F. G. Conrad, and E. Singer, “Evaluating the Effectiveness of Visual Analog Scales,” *Social Science Computer Review*, vol. 24, no. 2, pp. 227–245, 2006.
- [80] V. Barnett and T. Lewis, *Outliers in Statistical Data*. Wiley series in probability and mathematical statistics: Applied probability and statistics, Wiley & Sons, 1994.
- [81] D. Meintrup and S. Schäffler, *Stochastik. Statistik und ihre Anwendungen*, Springer, 2004.
- [82] H. W. Lilliefors, “On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown,” *Journal of the American Statistical Association*, vol. 62, no. 318, pp. 399–402, 1967.
- [83] S. S. Shapiro; M. B. Wilk, “An Analysis of Variance Test for Normality,” *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.
- [84] W. Lin and C.-C. J. Kuo, “Perceptual Visual Quality Metrics: A survey,” *Journal of Visual Communication and Image Representation*, vol. 22, no. 4, pp. 297–312, 2011.
- [85] Y. Wang, T. Jiang, S. Ma, and W. Gao, “Novel spatio-temporal structural information based video quality metric,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 7, pp. 989–998, 2012.
- [86] M. Hirth, T. Hossfeld, and P. Tran-Gia, “Anatomy of a Crowdsourcing Platform - Using the Example of Microworkers.com,” in *Proceedings of the 5th Innovative*

- Mobile and Internet Services in Ubiquitous Computing*, pp. 322–329, IEEE, 2011.
- [87] M. Waltl, C. Timmerer, B. Rainer, and H. Hellwagner, “Sensory Effect Dataset and Test Setups,” in *Proceedings of the 4th Workshop on Quality of Multimedia Experience*, pp. 115–120, IEEE, 2012.
- [88] W. Verhelst and M. Roelands, “An Overlap-add Technique based on Waveform Similarity (WSOLA) for High Quality Time-scale Modification of Speech,” in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 554–557, IEEE, 1993.
- [89] M. Li, “Qoe-based performance evaluation for adaptive media playout systems,” *Advances in Multimedia*, vol. 2013, p. 7, 2013.
- [90] M. Fiedler, T. Hossfeld, and P. Tran-Gia, “A Generic Quantitative Relationship between Quality of Experience and Quality of Service,” *IEEE Network*, vol. 24, no. 2, pp. 36–41, 2010.
- [91] K. Königsberger, *Analysis 1*. Analysis, Springer Berlin Heidelberg, 2013.
- [92] K. Königsberger, *Analysis 2*. No. 2 in Springer-Lehrbuch, Physica-Verlag, 2013.
- [93] F. Pereira, “A triple user characterization model for video adaptation and quality of experience evaluation,” in *Proceedings of the 7th Workshop on Multimedia Signal Processing*, pp. 1–4, IEEE, 2005.
- [94] C. T. Kelley, *Iterative Methods for Linear and Non-linear Equations*. No. 16 in Frontiers in Applied Mathematics, SIAM, 1995.
- [95] E. Polak, *Computational Methods in Optimization; a Unified Approach*. Academic Press New York, 1971.
- [96] J. A. Nelder and R. Mead, “A Simplex Method for Function Minimization,” *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.

- 
- [97] R. Roverso, S. El-Ansary, and S. Haridi, “Peer2View: A Peer-to-Peer HTTP-Live Streaming Platform,” in *Proceedings of the 12th International Conference on Peer-to-Peer Computing*, pp. 65–66, IEEE, 2012.
- [98] P. Eittenberger, M. Herbst, and U. Krieger, “RapidStream: P2P Streaming on Android,” in *Proceedings of the 19th International Packet Video Workshop*, pp. 125–130, IEEE, 2012.
- [99] C. Mueller, S. Lederer, J. Pöcher, and C. Timmerer, “libdash – An Open Source Software Library for the MPEG-DASH Standard,” in *Proceedings of the International Conference on Multimedia and Expo*, pp. 1–2, IEEE, 2013.
- [100] J. Le Feuvre, C. Concolato, and J.-C. Moissinac, “GPAC: Open Source Multimedia Framework,” in *Proceedings of the 15th International Conference on Multimedia*, pp. 1009–1012, ACM, 2007.
- [101] dash.js - DASH Industry Forum, “<https://github.com/dash-industry-forum/dash.js>,” Last Accessed November 2014.
- [102] B. Rainer, S. Lederer, C. Mueller, and C. Timmerer, “A Seamless Web Integration of Adaptive HTTP streaming,” in *Proceedings of the 20th European Signal Processing Conference*, pp. 1519–1523, European Signal Processing (EURASIP) Society, 2012.
- [103] mobile DASHEncoder, “<https://github.com/dazedsheep/dashtranscoder/>,” Last Accessed February 2015.
- [104] FFMPEG, “[www.ffmpeg.org](http://www.ffmpeg.org),” Last Accessed October 2014.
- [105] PowerTutor 2, “[http://ziyang.eecs.umich.edu/projects/power\\_tutor/](http://ziyang.eecs.umich.edu/projects/power_tutor/),” Last Accessed October 2014.
- [106] X. Chen, Y. Chen, Z. Ma, and F. C. A. Fernandes, “How is Energy Consumed in Smartphone Display Applications?,” in *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, pp. 1–6, ACM, 2013.

- 
- [107] C. Keimel, J. Habigt, C. Horch, and K. Diepold, “QualityCrowd: A Framework for Crowd-based Quality Evaluation,” in *Proceedings of the Picture Coding Symposium*, pp. 245–248, IEEE, 2012.
- [108] B. Rainer, M. Walzl, E. Cheng, M. Shujau, C. Timmerer, S. Davis, I. Burnett, and H. Hellwagner, “Investigating the impact of sensory effects on the quality of experience and emotional response in web videos,” in *Proceedings of the 4th International Workshop on Quality of Multimedia Experience*, pp. 278–283, IEEE, 2012.
- [109] S. E. Lab, “<http://selab.itec.aau.at>,” Last Accessed February 2015.
- [110] C. Timmerer, B. Rainer, and W. Markus, “A Utility Model for Sensory Experience,” in *Proceedings of the 5th International Workshop on Quality of Multimedia Experience*, pp. 224–229, IEEE, 2013.