Self-Organized Inter-Destination Multimedia Synchronization for Adaptive Media Streaming

Benjamin Rainer
Institute of Information Technology
Alpen-Adria-Universität Klagenfurt
9020 Klagenfurt, Austria
benjamin.rainer@itec.aau.at

Christian Timmerer
Institute of Information Technology
Alpen-Adria-Universität Klagenfurt
9020 Klagenfurt, Austria
christian.timmerer@itec.aau.at

ABSTRACT

As social networks have become more pervasive, they have changed how we interact socially. The traditional TV experience has drifted from an event at a fixed location with family or friends to a location-independent and distributed social experience. In addition, more and more Video On-Demand services have adopted pull-based streaming. In order to provide a synchronized and immersive distributed TV experience we introduce self-organized Inter-Destination Multimedia Synchronization (IDMS) for adaptive media streaming. In particular, we adapt the principles of IDMS to MPEG-DASH to synchronize multimedia playback among geographically distributed peers. We introduce session management to MPEG-DASH and propose a Distributed Control Scheme (DCS) to negotiate a reference playback timestamp among the peers participating in an IDMS session. We evaluate our DCS with respect to scalability and the time required to negotiate the reference playback timestamp. Furthermore, we investigate how to compensate for asynchronism using Adaptive Media Playout (AMP) and define a temporal distortion metric for audio and video which allows the impact of playback rate variations to be modeled with respect to QoE. This metric is evaluated based on a subjective quality assessment using crowdsourcing.

Categories and Subject Descriptors: C.2.4 Computer-Communication Networks: Distributed Systems; H.5.1 Information Interfaces and Presentation:Multimedia Information Systems

General Terms: Algorithms, Design, Measurement, Experimentation

Keywords: Inter-Destination Multimedia Synchronization; Adaptive Media Streaming; Self-Organization; Quality of Experience; Dynamic Adaptive Streaming over HTTP

MM'14, November 03 - 07 2014, Orlando, FL, USA.

http://dx.doi.org/10.1145/2647868.2654938.

1. INTRODUCTION

Over 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 place new requirements on the underlying technologies that help us create, distribute and view 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 presence of a real-time communication channel requires a synchronized playback of the multimedia 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 [8]. Consider for example, if, out of a group of friends 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. Thus, playback synchronization is a key feature of such a system. In general, the synchronization of the playback among geographically distributed users is termed Inter-Destination Multimedia Synchronization (IDMS) [19]. 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.

Our approach differs from existing push-based IDMS approaches that utilize RTP/RTCP receiver reports to signal timing and control information and instead extends IDMS to pull-based over-the-top streaming by adopting MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [23]. Specifically, it relies on the following mechanisms: *i*) **Session management** which is responsible for managing the session to which peers belong; *ii*) **Signalling of timing and control information** allows the exchange of timing information and, if necessary, control information between peers; *iii*) **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; *iv*) **Carrying out the synchronization** overcome the identified asynchronism by modifying the multimedia playback of each peer.

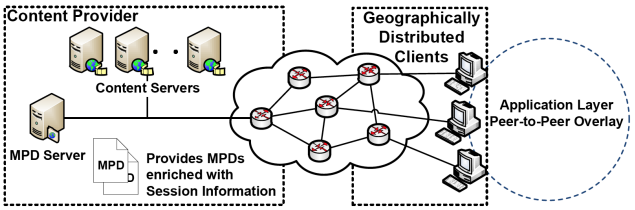

Figure 1: Architecture of IDMS for DASH.

In an 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 an MPEG-DASH compliant format (e.g., segmented ISO Base Media File Format or MPEG-2 Transport Stream). Furthermore, MPEG-DASH provides a Media Presentation Description (MPD) which describes the various spatial, temporal and quality dimensions of the content as well as different encodings. The MPD also contains information on the location of the content with support for multiple content servers [23].

Figure 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 contribution of this paper is as follows: *1*) Section 3.1 discusses our MPD extensions by including ISOs. Our actual IDMS approach adopts a Distributed Control Scheme (DCS) where peers within a session build a peer-to-peer (P2P) overlay for **signalling timing information** and **negotiating on a reference playback timestamp**. *2*) Section 3.2 describes the creation of the P2P overlay and how the playback timestamp is negotiated among the peers within a session. Furthermore, we address the scalability and reachability of peers in the P2P overlay. When peers in a session have agreed on a reference playback timestamp, the question arises how to overcome the identified asynchronism at each peer (i.e., **carrying out the synchronization**). *3*) Section 3.3 proposes a new approach that adopts Adaptive Media Playout (AMP) and aims on minimizing the impact on the Quality of Experience (QoE) when synchronizing the playback. In Section 4 we provide an evaluation of our IDMS approach and Section 5 concludes the paper.

2. RELATED WORK

2.1 Inter-Destination Multimedia Synchronization

The common assumption of most IDMS solutions is that clocks are already synchronized using existing time protocols (e.g., the Network Time Protocol (NTP) or the Precision Time Protocol (PTP)). Thus, most schemes only deal with the signalling of timing and control information to achieve IDMS among the participating clients [2, 19]. Current IDMS solutions tend to be tailored to very specific use cases, for example, the synchronization of multiplayer online games [12] or synchronization in collaborative work [11]. In general, the existing solutions can be grouped into three different schemes [2, 19]:

Master-/Slave (MS) scheme uses a dedicated master for signalling timing information which is elected from the participating peers or determined by the media source and thus implies a single point of failure. If the selected master leaves the sessions a new master must be selected or elected from the peers. Furthermore, the peers have to *trust* that the control and timing information received from the master is correct. The advantage of this scheme is that the content provider is not required to provide a central instance for signalling control and timing information. This allows the re-election of a master if the previously elected master leaves the IDMS session.

Synchronization Master Scheme (SMS) is a centralized approach where the synchronization is controlled by a synchronization master which is either the media source or a separate node (not a peer). The synchronization master collects timing information and sends timing instruction 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 instance is used, dedicated communication must be implemented between them. However, the approach has the advantage that the peers can trust the synchronization instance because it is in control of the content provider.

Distributed Control Scheme (DCS) uses distributed protocols to determine a common playback timestamp to which the peers may synchronize. Here, timing information is exchanged in a P2P manner among the peers. This scheme has the highest robustness in terms of overall failure probability as the content provider is only required to host the multimedia content. Nevertheless, the peers have to trust each other. Furthermore, Network Address Translators (NATs) may cause problems, especially if the peers are behind symmetric NATs.

In [24] the authors present a *Master-/Slave scheme* to achieve IDMS, which adopts the local lag and time warp algorithm to compensate for media playback inconsistencies [14]. Therefore, a peer or the source is selected as the master to which the media playback of all clients is synchronized. In [4] and [16] a SMS approach using RTP and RTCP for achieving IDMS is presented. Therefore, the receiver report message and the application-specific message of RTCP were extended. In a *Distributed Control Scheme* all the clients signal timing information by either unicasting or multicasting messages to each other. [9] presents the *iNEM4U* approach where a DCS is used to achieve IDMS among heterogeneous network infrastructures, thus, providing IDMS as a service using the synchronized multimedia integration language. Furthermore, it introduces *iSession* for the session management, which provides an XML description of each session including the users or clients, content source, and other service-specific data.

A very recent DCS for achieving IDMS which uses an extended version of RTCP messages is presented by [18]. The proposed DCS is designed on top of RTP/RTCP and assigns peers 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 between the peers. Another DCS which uses multicast

for intra-stream synchronization is presented in [12]. It appears that in contrast to our approach, most existing DCSs use multicast. We believe that it is risky to presume that multicast is supported by the open Internet. Furthermore, our DCS approach does not require a fully connected network where all peers can communicate with each other and our DCS approach is not directly coupled with the streaming technology employed. Pull-based streaming, typically MPEG-DASH, is used as an enabler and to store the session information that is needed to build the application layer P2P overlay (cf. Figure 1). As a result, our DCS approach may also be used in conjunction with other streaming technologies.

A crucial aspect of IDMS systems, specifically in distributed systems, is the selection of a reference or master to which other peers synchronize their media playback. In [3] three different reference selection policies are discussed: *i*) synchronization to the slowest peer, i.e., the client that is displaying the lowest frame number among the group of clients; *ii*) synchronization to the fastest peer, i.e., the client that is displaying the highest frame number; and *iii*) synchronization to the average frame number among a group of clients.

These policies assume that the playback is paused or audio/video frames are skipped to compensate for asynchronism which is in contrast to our approach where the playback rate is dynamically increased or decreased.

2.2 Adaptive Media Playout

Adaptive media playout (AMP) deals with overcoming the asynchronism and a common approach is adopting the skipping or pausing of media units. In [10] 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 [17] deals with increasing or decreasing the playback rate of media units without considering the influence on the QoE of the user. Furthermore, we assume that the playback rate of both audio and the video domain are altered simultaneously and 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. The authors of [25] modelled the adaptation of the media playback rate depending on the buffer variance. In [13] the buffer fill state was used to decide whether the playback rate should be increased/decreased. All these schemes try to avoid buffer under-flows or buffer over-flows in an error prone environment. In [6] the authors attempt to combine the decision with content features and use the spatial resolution of the video frames to influence the adaptive playback in wireless video streaming. We apply AMP to overcome identified asynchronisms and try to find time windows in the content where a playback rate increase or decrease has the least impact on the QoE.

The synchronization thresholds for IDMS using different communication methods were investigated by [8]. The authors found that the upper bound on acceptable asynchronism varies depending on the communication tool. For example, when users are provided with a voice communication tool, asynchronism is only subjectively perceived for delays greater than two seconds.

3. SELF-ORGANIZED IDMS FOR ADAPTIVE MEDIA STREAMING

3.1 Session Management

We adopt MPEG-DASH [23] 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 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. 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.

Definition: In the context of IDMS we define an ISO as 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.

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 signalled by adding it to the HTTP GET message that requests a MPD from the *MPD Server*. As peers may use Network Address Translation (NAT), Session Traversal Utilities for NAT (STUN, RFC 5389) is applied to determine the public IP address and port number to be used during the synchronization. We use the same ports for STUN negotiation and our synchronization protocols. Each peer communicates with a STUN server (in our case the *MPD Server*) in order to determine whether the peer is behind a NAT and, if available, the type of NAT. Appendix A outlines how the type of the NAT is identified and how it can be traversed.

The resulting public IP (IPv4 or IPv6) address, port number, and NAT type is added along with the session key to the initial HTTP GET message that requests the MPD from the *MPD Server* as URL parameters. The initiation of an IDMS session and the provision of the session key is out of the scope of this paper. 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.

```
<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">
  <xs:import namespace="http://www.w3.org/1999/xlink"
      schemaLocation="xlink.xsd"/>
  <xs:element name="IDMSSessionObject">
    <xs:complexType>
      <xs:sequence>
```

```
            <xs:element name="PeerList" type="PeerListType"
                 minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="TTL" type="xs:dateTime"
                 minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute ref="xlink:href"/>
        <xs:attribute ref="xlink:actuate" default="onLoad"/>
    </xs:complexType>
</xs:element>
<xs:complexType name="PeerListType">
    <xs:sequence>
        <xs:element name="Peer" type="PeerType" minOccurs="0"
             maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="PeerType">
    <xs:sequence>
        <xs:element name="Identifier" type="
             PeerIdentifierType" minOccurs="1"
            maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="PeerIdentifierType">
    <xs:sequence>
        <xs:element name="IP" type="xs:string"/>
        <xs:element name="Port" type="xs:integer"/>
    </xs:sequence>
    <xs:attribute name="nat" type="xs:string"/>
</xs:complexType>
</xs:schema>
```

Listing 1: IDMS Session Object for MPEG-DASH.

Listing 1 depicts the XML Schema for an ISO. The ISO includes a list of peers (represented by *@PeerListType*) in the IDMS session and a Time-To-Live (TTL). The maximum TTL for an IDMS session is the duration of the requested multimedia content. 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.

```
<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">
<BaseURL>http://www.example.com/</BaseURL>
<Period>
    <AdaptationSet>
        <Representation id="0" mimeType="video/mp4" codecs="avc1,
             mp4a" startWithSAP="1" bandwidth="1713804">
            <!-- ... representation info -->
        </Representation> <!-- ... more representations -->
    </AdaptationSet>
</Period>
<iso:IDMSSessionObject>
    <iso:PeerList>
        <iso:Peer>
            <iso:Identifier nat="NoNAT">
                <iso:IP>143.205.122.242</iso:IP>
                <iso:Port>8029</iso:Port>
            </iso:Identifier>
            <iso:Identifier nat="FullCone">
                <iso:IP>143.205.199.149</iso:IP>
                <iso:Port>8030</iso:Port>
            </iso:Identifier>
        </iso:Peer> <iso:Peer>
            <iso:Identifier nat="PortRestricted">
                <iso:IP>10.0.0.5</iso:IP>
                <iso:Port>8029</iso:Port>
            </iso:Identifier>
        </iso:Peer> <!-- ... more peers -->
    </iso:PeerList> <iso:TTL>2014-07-26T21:32:52</iso:TTL>
</iso:IDMSSessionObject>
</MPD>
```

Listing 2: Excerpt of an example MPD with an ISO.

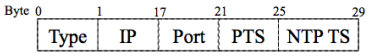

Figure 2: Message structure for the *coarse synchronization*.

Listing 2 shows an excerpt of an MPD comprising an ISO. Peers requesting the MPD will be added to the 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.

3.2 Signalling of Timing Information and Negotiation of the Playback Timestamp

For creating the P2P overlay in order to signal timing information and negotiating 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 segments 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 self-organized manner by using the constructed P2P overlay. In general, we assume that the clocks of the peers are synchronized (e.g. using NTP or PTP).

3.2.1 Peer-To-Peer Overlay Construction and Coarse Synchronization

The P2P overlay is created using the information contained in the ISO. Prior to any communication among the peers, each peer uses STUN to detect the NAT type and, in the case of a restricted NAT or port restricted NAT, it follows the procedure described in Appendix A. UDP is used as the transport protocol between the peers because reliable communication is not essential. Each peer that receives the ISO requests the current playback timestamp from all peers it lists. Figure 2 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 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*. The peer requesting the playback timestamps waits until either all requests have been satisfied or 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 (i.e., *request_timestamps*). Each peer that receives a request (i.e., *receive_request*) responds with its IP address, port number, playback timestamp, and the corresponding

Algorithm 1 Coarse Synchronization.

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

NTP timestamp. The timestamps from the responses may be combined to calculate the start segment using one of four strategies, namely the *i*) maximum, *ii*) minimum, *iii*) average, and *iv*) weighted average of the received timestamps. Which strategy should be selected 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 playback timestamps.

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. Suppose that M is the theoretical reference timestamp to which all peers will adjust their playback. Therefore, without loss of generality the asynchronism ξ after asking the other peers for their playback timestamp and downloading N segments until the playback starts is given by:

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

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.

3.2.2 Self-organized Fine Synchronization

The second 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. 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 or those that have just joined. Selecting the minimum would privilege

Algorithm 2 Merge and Forward.

```
B_i, L_i, P_i, NTP_i, I_i^M, I_i^m, S_i, C_i ← 1
1: function broadcastToNeighbors
2:     update(P_i, NTP_i)
3:     for all p ∈ 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, S_j, C_j, B_j)
8:     if S_j > S_i then
9:         B_i ← H(i), L_i ← ∅, S_i ← S_j, C_i ← 1
10:    end if
11:    update(P_i, NTP_i), update(P_j, NTP_j)
12:    if B_i ⊕ B_j ≠ 0 ∧ B_i ∩ B_j = ∅ then
13:        B_i ← B_i + B_j, P_i ← (P_i·C_i + P_j·C_j) / (C_i + C_j)
14:        I_i^M ← max{I_i^M, I_j^M}, I_i^m ← min{I_i^m, I_j^m}
15:        C_i ← C_i + C_j
16:    end if
17:    if B_i ⊕ B_j ≠ 0 ∧ B_i ∩ B_j ≠ 0 ∧ B_j ∉ L_i then
18:        if C_j ≥ C_i ∧ i ∈ B_i ∩ B_j then
19:            B_i ← B_j, P_i ← P_j, C_i ← C_j
20:        else if C_j ≥ C_i then
21:            B_i ← B_j + H(i), P_i ← (P_j·C_j + P_i) / (C_j + 1)
22:            C_i ← C_j + 1
23:        end if
24:        I_i^M ← max{I_i^M, I_j^M}, I_i^m ← min{I_i^m, I_j^m}
25:    end if
26:    L_i ← {B_j} ∪ L_i
27: end function
```

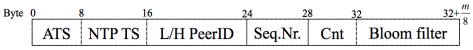

Figure 3: Message structure for the *fine synchronization*.

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 [18]. 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 $h_1(x)$, ..., $h_k(x)$ for computing the bits to set when inserting itself into the Bloom filter [5]. This allows us to exchange a fixed length packet and contributes to the scalability of our P2P approach. Figure 3 depicts the message structure for the P2P communication once the overlay has been constructed by the *coarse synchronization* with the following semantics: *a*) ***ATS***: average playback timestamp. *b*) ***NTP TS***: NTP timestamp for aligning the playback timestamp. *c*) ***L/H PeerID***: the lowest and highest peer identification number seen by the sending peer according to the ISO (i.e., number of *Peer* elements within the ISO). *d*) ***Seq.Nr.***: indicates the current synchronization round. *e*) ***Cnt.***: the number of peers contributed to current average playback timestamp. *f*) ***Bloom filter***: fixed length Bloom filter with a length of m bits. The *PeerID* 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 *Seq.Nr.* 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 will reset to the initial condition and start over. The P2P algorithm making use of this message structure is shown in Algorithm 2 and referred to as *Merge and Forward (M&F)*.

Each peer i maintains a Bloom filter B_i and inserts itself with its own peer id according to the indices obtained by the use of a common set of k hash functions $h_1(x), ..., h_k(x)$, a list of already seen Bloom filters L_i, a playback timestamp P_i and its corresponding NTP timestamp depicted by NTP_i, a sequence number S_i which is initially set to zero, and the highest peer id seen I_i. $H(x)$ depicts the function that generates the bit-sequence for peer x by applying $h_1(x), ..., h_k(x)$. Each peer **forwards** B_i, P_i, C_i, I_i^M and I_i^m periodically to its neighbors depicted by the function $broadcastToNeighbors$, with period τ. If peer i receives a Bloom filter from one of its neighbors j, it checks whether it can **merge** the Bloom filters. The Bloom filters can only be merged if they are disjoint to avoid introducing a bias to the weighted average. If the Bloom filters are merged by using the bit-wise OR depicted by $+$, then we calculate the weighted average between P_i and P_j. C_i depicts the number of peers that contributed to the average playback timestamp. If the Bloom filters are not disjoint and if we have not seen the received Bloom filter yet, we store the received Bloom filter and add it to the list of already seen Bloom filters. Finally, the 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. 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. If a peer receives an sequence number that is higher than its own, it resets B_i such that it only includes itself, empties the list L_i and sets $S_i \leftarrow S_j$. 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., $PTS = w_i \cdot PTS$, $w_i \geq 1$).

In order to calculate the overlap or the intersection of two Bloom filters ($B_i \cap B_j$), the algorithm must know which peers have already been inserted. Due to the nature of Bloom filters testing if peers are 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. The probability of encountering a false positive for the received Bloom filter is given by $(\frac{s}{m})^k$, s the number of bits set [5]. Therefore, if we try to determine which and how many peers $(p_1, p_2, ..., p_n)$ have contributed to the timestamp represented by the Bloom filter, assuming that we test for n peers and h peers are really in the Bloom Filter ($h \leq n$) the false positive rate is given by: $P_{false}(p_1 \vee p_2 \vee ... \vee p_n) = \sum_{i=1}^{n-h} (\frac{s}{m})^k = (n-h) \cdot (\frac{s}{m})^k$ (due to independence). *Merge and Forward* uses the *L/H PeerID* as the upper limit and lower limit to test for peers in a Bloom filter. Nevertheless, this has no impact on the calculated average playback timestamp because the false positives have only an impact on which Bloom filters are merged and, thus, only increase the required time. Therefore, we assume that the size of the Bloom filter is chosen sufficiently large. It can be proven that the resulting average playback timestamp equals to real average playback timestamp of the peers [*Sketch of proof*: solving the recursive construction of the weighted average playback timestamp (binary tree) leads to the real average playback timestamp. Furthermore, through contradiction it is shown that the calculated average playback timestamp is unique for the same number of peers and playback timestamps among different network graphs].

3.3 Dynamic AMP for IDMS

In this section we describe our dynamic AMP approach for IDMS which dynamically increases/decreases the playback rate to overcome the asynchronism without impacting the QoE. Other approaches neither consider the impact on the QoE nor vary the playback rate dynamically every time a peer synchronizes its playback (cf. Section 2). 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:

$$\arg\min_X f(X) \tag{2a}$$

$$x_2 \cdot (x_3^{sign(\xi)} - 1) \cdot sign(\xi) = |\xi| \tag{2b}$$

$$L \leq B - x_2 \cdot x_3 + x_2 \cdot \frac{b_c}{b_r} \tag{2c}$$

$$x_1 \leq T \tag{2d}$$

$$x_2 \leq t_{max} \tag{2e}$$

where $X \in \mathbb{R}^3$ is 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 X such that X^* is a minimizer for $f(X)$ ($\forall X \in \mathbb{R}^3 : f(X^*) \leq f(X)$). $f(X)$ 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 2b which states that the given asynchronism should be compensated for by selecting appropriate values for x_2 and x_3, respectively. ξ 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 2c 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 2d). Equation 2e limits the duration (t_{max}) of the playback variation.

By measuring distortion in the audio and video domain we attempt to approximate the impact of playback rate changes on the QoE. Therefore, we define a metric that allows measuring the playback rate variations with respect to the (vi-

sual) motion intensity and the (audio) spectral energy. We derive the visual feature (motion intensity) from the average length of the motion vectors between two consecutive frames whereas the audio featrue is extracted from the spectral energy of the audio frames (for each channel). We compare these audio-visual (AV) features from a given content section with a given duration (e.g., 2s) for the increased/decreased playback rate and the nominal playback rate. As the length of the asynchronism is known, both the target playback rate and the duration of the playback rate change can be calculated. Importantly, introducing an increase/decrease in the playback rate will alter the duration of the respective content section which may or may not be perceived by the user.

The video metric is defined in Equation 3.

$$d_v(X) = \sum_{j=s_v}^{\widetilde{e_v}} f_v(j) - \sum_{j=s_v}^{e_v} f_v(j) \tag{3}$$

where s_v denotes the number of the first frame of the content section for which the playback rate should be changed. $\widetilde{e_v} = s_v + \lfloor (t_e - t_b) \cdot fps_{\Delta\mu} \rfloor$ is the index of the last frame when using the increased/decreased playback rate, t_e the end timestamp of the content section, t_a the start timestamp of the content section, $fps_{\Delta\mu}$ the frame rate of the changed playback rate $\Delta\mu$, and e_v is the number of the last frame using the nominal playback rate for a given duration. f_v denotes the average motion intensity.

The audio metric is defined in Equation 4.

$$d_a(X) = \sum_{c=1}^{C}((\sum_{u=s_a}^{\widetilde{e_a}} \sum_{k=0}^{S_f} |\widehat{a}_{c_u}(k)| - \sum_{u=s_a}^{e_a} \sum_{k=0}^{S_f} |\widehat{a}_{c_u}(k)|)) \tag{4}$$

where C denotes the number of available audio channels and S_f denotes the highest frequency. s_a, $\widetilde{e_a}$, and e_a are defined in the same way as for video. The Fourier transformed audio frames are denoted by $\widehat{a}_c$ (we use a half overlapping Hamming window) for a given audio channel c.

The combined AV metric is defined in Equation 5.

$$g(X) = \sqrt{d_v(X)^2 + d_a(X)^2} \tag{5}$$

$g(X)$ is neither convex nor continuous because it depends on the features which are derived from the actual content.

The highest asynchronism in our IDMS system is given by Equation 1. The initial asynchronism of a newly joined peer depends on the time the peer requires for downloading sufficient 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 ξ 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 2 into the following optimization problem:

$$\arg\min_X f(X) = \begin{cases} g(X) + \gamma \cdot \sum_{i=1}^{3} p_i(X) & \text{if } \xi \geq 0 \\ g(X) + \gamma \cdot \sum_{i=2}^{3} p_i(X) & \text{if } \xi < 0 \end{cases} \tag{6a}$$

$$p_1(X) = \min\{0, B - x_2 \cdot x_3 + x_2 \cdot \frac{b_c}{b_r} - L\}^2 \tag{6b}$$

$$p_2(X) = \min\{0, T - x_1\}^2 \tag{6c}$$

$$p_3(X) = \min\{0, t_{max} - x_2\}^2 \tag{6d}$$

Equation 6a shows the transformed cost function. To reduce the set of feasible points we use the constraint given in Equation 2b and apply the implicit function theorem, reducing x_2 to a function of x_3 by $u(x_3) = sign(\delta_k) \cdot \frac{|\delta_k|}{x_3^{sign(\delta_k)} - 1}$ for $x_3 \neq 1$ [*Proof*: by application of the implicit function theorem on $f(x_3, x_2) = x_2 \cdot (x_3^{sign(\delta_k)} - 1) \cdot sign(\delta_k) - |\delta_k|$]. Thus, we try to find values x_1 and x_3 that minimize $f(X)$. The penalty function $p_1(X)$ (cf. Equation 6b) states that the buffer fill state shall not be drained below the threshold L when increasing the playback rate. $p_2(X)$ (cf. Equation 6c) depicts the costs that depend on the starting point of the playback rate variation. $p_3(X)$ (cf. Equation 6d) states that a peer should be synchronized within t_{max} seconds. γ sets the penalty factor for the transformed constraints ($\gamma > 0$). For solving the optimization problem during the multimedia playback we use multiple instances of the Nelder-Mead algorithm with different starting points [20].

To conclude, our dynamic AMP approach overcomes asynchronism by searching for content sections where the playback rate may be increased/decreased having the least impacting on the QoE.

4. EXPERIMENTAL RESULTS

The evaluation of our self-organized IDMS for DASH approach comprises two aspects. First, we investigate the proposed DCS 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 A/V metric and the dynamic AMP approach.

4.1 Evaluation of Merge and Forward

To evaluate the performance of the *Merge and Forward* algorithm, we compare our algorithm to an approach that is similar to the one described in [18]. As this approach uses multicast, we modify it such that each peer aggregates all received playback timestamps into a single message and sends them periodically to its neighbors using unicast because we do not assume that multicast is in place. We call this approach *Aggregate*. We compare the two algorithms based on the overhead produced by peers agreeing on the average playback timestamp and the duration of the process. The evaluation scenario foresees that a certain number of peers will join the P2P overlay over the lifetime of the session. If the peers join one by one after the existing peers have already synchronized, the time required by *Merge and Forward* to synchronize is the same as *Aggregate*. Furthermore, peers may leave the P2P overlay during the negotiation process as any reference timestamp they have contributed persists until a re-synchronization is triggered.

We simulated the algorithms on Erdös-Renyi random networks [7] implemented using OMNeT++ [21]. A period τ of 250 milliseconds was used for both algorithms, with the round trip time set to 300 milliseconds and a maximum clock skew of 30 milliseconds randomly (uniformly) chosen from an interval of $[-15, 15]$. For *Merge and Forward* we set the size of the Bloom filter to 512 bit. We generate random net-

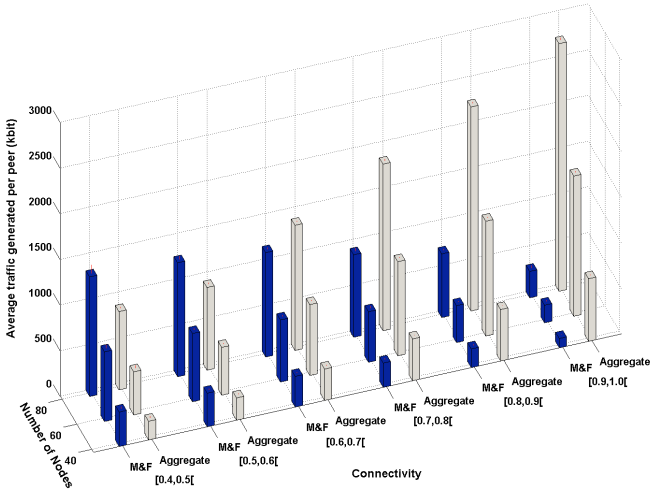

Figure 4: Traffic overhead per peer until all peers have the same reference playback timestamp using *Merge and Forward* (M&F) and *Aggregate* (95% CI).

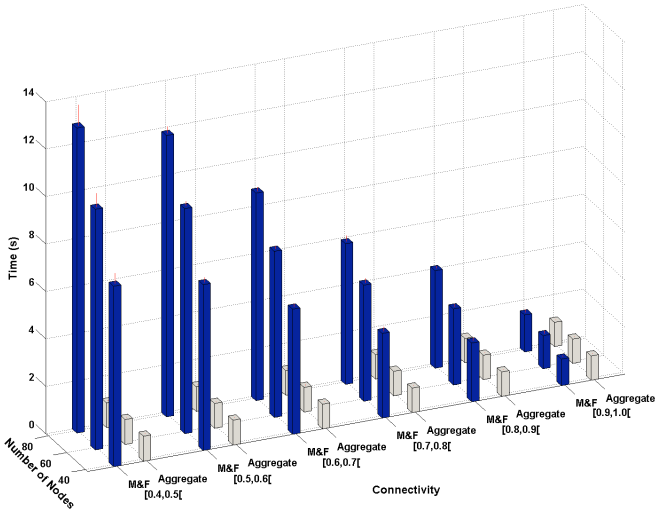

Figure 5: Time for the distributed calculation of the average playback timestamp using *Merge and Forward* (M&F) and *Aggregate* (95% CI).

works for the following number of peers: 40, 60, and 80 with following probabilities for creating a connection between two peers: 0.1 to 0.9 increased in 0.1 steps. We use these numbers of peers and connectivities in order to show how the algorithms scale when increasing the number of peers and their connectivity. For each of the parameter settings we conducted 30 simulation runs and taking the average of the results.

Figure 4 illustrates the average network traffic (in kbit) generated at each peer by both algorithms during the agreement phase with respect to the overall connectivity of the network. The connectivity is given by the ratio of the average node degree and the number of peers in the network ($c = \frac{\frac{1}{|V|}\sum_{i=1}^{|V|} d_G(v)}{|V|-1}$, V is the set of vertices and $d_G(v)$ is the degree of node $v \in V$). The x-axis represents connectivity in intervals which increase in 0.1 steps. With low connectivity, *Merge and Forward* generates more traffic than *Aggregate* but this is due to the time required to compute the average playback timestamp among all peers. If the connectivity increases the fixed length messages of *Merge and Forward* start to pay off and it subsequently outperforms the *Aggregate* algorithm.

Figure 5 depicts the time for the distributed calculation of the average playback timestamp. Here, *Aggregate* represents the optimum case because lists of timestamps can be merged even if they overlap because the contribution of each peer can be clearly identified. Using Bloom filters and only a single field for the average playback timestamp, the contribution of a single peer cannot be uniquely identified anymore. If a peer receives a Bloom filters where a subset of the peers contributed to the received one and to one that the peer holds, calculating the weighted average would yield a skewed weighted averaged compared to the real weighted average without the overlap. However, with an increase in the connectivity, the time needed by *Merge and Forward* converges to the time needed by *Aggregate*. This shows that the negotiation time required by *Merge and Forward* does not solely depend on the number of peers in the overlay.

Combining Figure 4 and Figure 5 yields the average traffic in kbit/s generated at each peer by both algorithms during the agreement phase, *Merge and Forward* always outperforms *Aggregate*. If a peer does join an IDMS session, the whole process of calculating the reference playback timestamp is restarted. Compared to *Merge and Forward*, using *Aggregate* or any other pure flooding algorithm will generate more overhead as the number of peers or connectivity increases. The assumption that *Merge and Forward* has to calculate the average among all playback timestamps and the low overhead result in an increase in the time until all peers hold the same timestamp.

4.2 Evaluation of the dynamic AMP

We evaluated our dynamic AMP approach by conducting a subjective quality assessment using crowdsourcing [22]. The aim is to determine how the AV metrics introduced in Equation 3, Equation 4, and Equation 5 reflect the actual impact on the QoE when increasing or decreasing the playback rate.

To carry out our user study we used the Microworkers [15] crowdsourcing platform which hosts so-called campaigns to which users (microworkers) can subscribe. These campaigns include a detailed description of the task and ask subjects to submit proof that they participated in the campaign. Our user study was designed with a duration of 15 minutes and offered $0.25 as a reward, slightly more that the typical $0.20 paid for such a campaign. The design of our study is as follows: *1.*) We present an introduction where we explain the task and the test procedure in detail. *2.*) A pre-questionnaire is presented to the subject which asks for demographic data. *3.*) A short training phase is conducted in order to mitigate the surprise effect. *4.*) The main evaluation takes place using a Single Stimulus method as defined in [1]. *5.*) At the end a post-questionnaire is presented which allows for comments. *6.*) A unique token is provided as proof for a successful participation.

For the training phase we selected a short video sequence from Babylon A. D. The training sequence is presented with three different media playback rates $\mu \in \{1, 0.5, 2\}$ in order to introduce AMP to the subject and and the corresponding effect on the playback. A playback rate of one depicts the

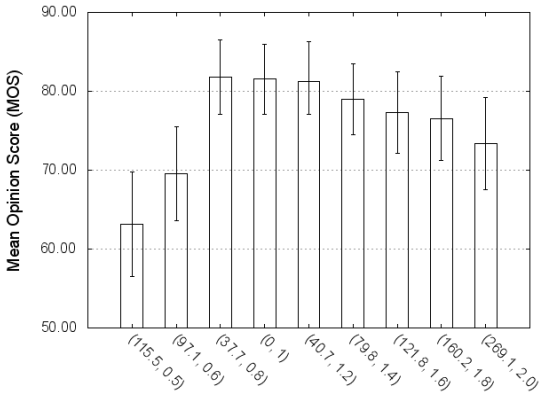

Figure 6: MOS and 95% CI for $(\overline{g(X)}, \mu)$.

nominal playback rate of the video sequence (e.g., 25 frames per second). A playback rate of 2 is twice as fast as the nominal playback rate and 0.5 is half the nominal playback rate. The stimulus for the main evaluation is the first 51 seconds of Big Buck Bunny (http://www.bigbuckbunny.org). This content provides high and low motion scenes as well as scenes with high and low audio volume. We introduce four content sections for which the playback rates are increase and decreased. The playback timestamps for the selected content sections in seconds are as follows (start-end): 6.4–7.2, 9–10, 16–18, 35–38, and 46–49. These content sections are presented with following playback rates: 0.5, 0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8, 2. The multimedia player that is employed uses the Waveform Similarity based Overlap-Add algorithm [26] that tries to preserve pitch for the audio domain while increasing or decreasing the playback rate. For rating the QoE we use a continuous rating scale $[0, 100]$ displayed as a slider. Furthermore, we randomly insert a control question after a stimulus presentation which asks the participants what they have seen in the previous video sequence.

For the statistical analysis of the results, we screened the subjects according to [1] and specifically removed participants who failed to correctly answer to the control question. This yielded 55 (48 male and 7 female) subjects for the statistical analysis out of a total of 80 participants.

The QoE ratings for each stimulus presentation were subject to a Shapiro-Wilk-test to assess whether the ratings were normally distributed. The null hypothesis ($H0$), stating that no normal distribution is present, was rejected for each configuration of playback rates. Furthermore, we calculate the average distortion $\overline{g(X)}$ for each stimulus presentation.

Figure 6 depicts the assessed Mean Opinion Score (MOS) for each tuple $(\overline{g(X)}, \mu)$. The hidden reference condition is given by $\mu = 1$. The results state that playback rates below (i.e, $\mu = 0.8$) and above (i.e., $\mu \in \{1.2, 1.4, 1.6, 1.8\}$) the nominal playback rate have no significant impact on the QoE. This confirmed a Student's t-test for equal sample variances which showed no significant difference in MOS between the reference condition and the playback rates as follows: $\mu = 0.8 : p = 0.93, t = -0.083; \mu = 1.2 : p = 0.92, t = 0.096; \mu = 1.4 : p = 0.81, t = 0.42; \mu = 1.6 : p = 0.22, t = 1.23; \mu = 1.8 : p = 0.16, t = 1.41$ for $\alpha = 5\%$. These results indicate that the subjects could not notice a significant difference for playback rates within the range of $[0.8, 1.8]$. For the other playback rates, the QoE significantly degrades. A Student's t-test revealed that there is a statistical significant difference between the MOS of the reference condition and the playback rates $\mu = 0.5 : p = 0.00, t = 4.5217; \mu = 0.6 : p = 0.002, t = 3.2; \mu = 2 : p = 0.03, t = 2.19$ for $\alpha = 5\%$.

Finally we investigated how well our distortion metric defined in Equation 5 correlates with the assessed MOS. 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 $p = 0.0009$. Playback rates lower than the nominal playback rate exhibit a high negative correlation with $\rho = -0.995$ and $p = 0.0047$. This indicates a strong linear correlation between our distortion metric and the MOS assessed by the subjective quality assessment, supporting our approach to the optimization problem as stated in Section 3.3. Furthermore, such an optimizer will always perform equal or better than skipping or pausing the multimedia playback in terms of QoE.

5. CONCLUSION AND FUTURE WORK

In this paper we develop IDMS for pull-based streaming by using a DCS to negotiate a reference playback timestamp among the peers participating in a session. Specifically, we introduce 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. Following this, we describe a DCS for negotiating a reference playback timestamp among the peers in an IDMS Session. The proposed DCS was evaluated with respect to scalability and the time required to synchronize a certain number of peers. The results show that *Merge and Forward* scales very well with the number of peers. Furthermore, the overhead saved may allow the peers to request higher quality streams which can improve the overall QoE of the IDMS system.

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 introduce a re-synchronization method that allows a peer to influence the calculation of the reference playback timestamp such that all peers are guaranteedly able to synchronize their multimedia playback. We leave the problem of determining an appropriate weight for a given peer's playback timestamp as subject to future work.

The synchronization of an IDMS system is crucial because it directly impacts the QoE, with skipping and pausing leading to an undesirable degradation. As a result, we adopt AMP and formulate an optimization problem which selects appropriate parameters such that the impact of the resulting playback rate adjustment on the QoE is minimized while avoiding asynchronism. We use a distortion metric as the objective function whose validity for modeling the impact of playback rate adjustments on the QoE is supported by the results of a subjective quality assessment. The evaluation of our IDMS approach in a real world environment and the QoE provided by the system remain as future work.

6. ACKNOWLEDGMENTS

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. The authors thank the anonymous reviewers for their helpful and insightful comments that have greatly improved this paper.

7. REFERENCES

[1] Rec. ITU-R BT.500-11. Technical report.

[2] F. Boronat, J. Lloret, and M. García. Multimedia group and inter-stream synchronization techniques: A comparative study. *Inf. Syst.*, 34(1):108–131, 2009.

[3] F. Boronat, M. Montagud, and V. Vidal. Master Selection Policies for Inter-destination Multimedia Synchronization in Distributed Applications. In *IEEE 19th Int. Symp. on MASCOTS*, pages 269–277, 2011.

[4] F. Boronat Seguí, J. Guerri Cebollada, and J. Lloret Mauri. An RTP/RTCP based approach for multimedia group and inter-stream synchronization. *MTAP*, 40:285–319, 2008.

[5] K. Christensen, A. Roginsky, and M. Jimeno. A New Analysis of the False Positive Rate of a Bloom Filter. *Inf. Process. Lett.*, 110(21):944–949, Oct. 2010.

[6] H.-C. Chuang, C. Huang, and T. Chiang. Content-Aware Adaptive Media Playout Controls for Wireless Video Streaming. *IEEE Trans. on Multimedia*, 9(6):1273–1283, 2007.

[7] P. Erdös and A. Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.

[8] D. Geerts, I. Vaishnavi, R. Mekuria, O. van Deventer, and P. Cesar. Are we in sync?: synchronization requirements for watching online video together. In *Proc. of the SIGCHI*, CHI '11, pages 311–314, New York, NY, USA, 2011. ACM.

[9] 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 Comm. Mag.*, 48(6):54–65, 2010.

[10] T. Hossfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz. Quantification of YouTube QoE via Crowdsourcing. In *IEEE ISM*, pages 494–499, 2011.

[11] Y. Ishibashi, T. Hasegawa, and S. Tasaka. Group synchronization control for haptic media in networked virtual environments. In *12th Int. Symp. on HAPTICS.*, pages 106 – 113, 2004.

[12] Y. Ishibashi and S. Tasaka. A distributed control scheme for causality and media synchrnoization in networked multimedia games. In *IEEE Proc. of the 11th ICCN*, pages 144–149, 2002.

[13] M. Kalman, E. Steinbach, and B. Girod. Adaptive media playout for low-delay video streaming over error-prone channels. *IEEE Trans. on Circuits and Systems for Video Technology*, 14(6):841–851, 2004.

[14] M. Mauve, J. Vogel, V. Hilt, and W. Effelsberg. Local-lag and Timewarp: Providing Consistency for Replicated Continuous Applications, 2002.

[15] Microworkers. http://www.microworkers.com.

[16] M. Montagud. Design, development and evaluation of an adaptive and standardized rtp/rtcp-based idms solution. In *Proc. of the 21st ACM MM*, pages 1071–1074, New York, NY, USA, 2013. ACM.

[17] M. Montagud and F. Boronat. On the Use of Adaptive Media Playout for Inter-Destination Synchronization. *IEEE Communications Letters*, 15(8):863–865, 2011.

[18] M. Montagud, F. Boronat, and H. Stokking. Design and Simulation of a Distributed Control Scheme for Inter-destination Media Synchronization. In *IEEE 27th AINA*, pages 937–944, March 2013.

[19] M. Montagud, F. Boronat, H. Stokking, and R. Brandenburg. Inter-destination multimedia synchronization: schemes, use cases and standardization. *Multimedia Sys.*, 18:459–482, 2012.

[20] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, 1965.

[21] OMNET++ 4.3.1. http://www.omnetpp.org/.

[22] B. Rainer and C. Timmerer. A quality of experience model for adaptive media playout. In *6th QoMEX*. IEEE, September 2014.

[23] I. Sodagar. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE MultiMedia*, 18(4):62–67, 2011.

[24] H. Stokking, M. Van Deventer, O. Niamut, F. Walraven, and R. Mekuria. IPTV inter-destination synchronization: A network-based approach. In *14th ICIN*, pages 1–6, 2010.

[25] Y.-F. Su, Y.-H. Yang, M.-T. Lu, and H. H. Chen. Smooth control of adaptive media playout for video streaming. *Trans. Multi.*, 11(7):1331–1339, Nov. 2009.

[26] W. Verhelst and M. Roelands. An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech. In *IEEE ICASSP*, volume 2, pages 554–557, April 1993.

APPENDIX

A. NETWORK ADDRESS TRANSLATION

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 signalled 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 signalled information to open its NAT. After this *handshake* the P2P synchronization protocol can be carried out as described in Section 3.2. 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 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 paper.