An Evaluation of Dynamic Adaptive Streaming over HTTP in Vehicular Environments

Christopher Müller, Stefan Lederer and Christian Timmerer
Alpen-Adria-Universität Klagenfurt
Universitätsstraße 65-67
9020 Klagenfurt am Wörthersee, Austria
+43 (0) 463 2700 3600
{firstname.lastname}@itec.aau.at

ABSTRACT
MPEGs' Dynamic Adaptive Streaming over HTTP (MPEG-DASH) is an emerging standard designed for media delivery over the top of existing infrastructures and able to handle varying bandwidth conditions during a streaming session. This requirement is very important, specifically within mobile environments and, thus, DASH could potentially become a major driver for mobile multimedia streaming. Hence, this paper provides a detailed evaluation of our implementation of MPEG DASH compared to the most popular propriety systems, i.e., Microsoft Smooth Steaming, Adobe HTTP Dynamic Streaming, and Apple HTTP Live Streaming. In particular, these systems will be evaluated under restricted conditions which are due to vehicular mobility. In anticipation of the results, our prototype implementation of MPEG-DASH can very well compete with state-of-the-art solutions and, thus, can be regarded as a mature standard ready for industry adaption.

Categories and Subject Descriptors
D.5.1 [**Multimedia Information System**]: Video.

General Terms
Algorithms, Measurement, Standardization, Documentation.

Keywords
Dynamic Adaptive Streaming over HTTP, MPEG-DASH, Microsoft Smooth Streaming, Adobe HTTP Dynamic Streaming, Evaluation, Apple HTTP Live Streaming, Mobile Networks, Vehicular Mobility.

1. INTRODUCTION
Media streaming over the hypertext transfer protocol (HTTP) and in a further consequence streaming over the transmission control protocol (TCP) has become omnipresent. Content providers such as Netflix, Hulu, and Vudu do not deploy their own streaming equipment but use the existing Internet infrastructure as it is and they simply utilize their own services over the top (OTT). This streaming approach works surprisingly well without any particular support from the underlying network due to the use of efficient video compression, content delivery networks (CDNs), and adaptive video players. The assumption of earlier video streaming research, which mostly recommended the user datagram protocol (UDP) and the real time transport protocol (RTP), that it would not be possible to transfer multimedia data smoothly with TCP, because of its throughput variations and large retransmission delays, could be seen as a delusion from today's point of view. HTTP streaming and especially its most simple form which is known as progressive download has become very popular over the past few years because it has some major benefits compared to RTP streaming. As a consequence of the consistent use of HTTP for this streaming method the existing Internet infrastructure, consisting of proxies, caches and CDNs could be used. Originally this architecture was designed to support best effort delivery of files and not real time transport of multimedia data. Nevertheless, also real time streaming based on HTTP could take advantage out of this architecture, in comparison to RTP which could not utilize any of the aforementioned components. Another benefit that results from the use of HTTP is that the media stream could easily pass firewalls or network address translation (NAT) gateways which was definitely a key for the success of HTTP streaming. However, HTTP streaming is not the holy grail of streaming as introduces also some drawbacks compared to RTP. For example, as HTTP is based on TCP an overhead is introduced that is approximately twice the media bitrate [1].

Akhsabi et al. [2] evaluated Microsoft Smooth Streaming, Adobe HTTP Dynamic Streaming, and the Netflix Player using simulated bandwidth traces. They used different test content for each system in question and, thus, the results are difficult to compare. Yao et al. [3] evaluated the possibility of using HTTP streaming under vehicular mobility with 3rd generation mobile networks. The evaluation is based on real world bandwidth traces using their own, proprietary client. However, their evaluation focused on the comparison of their system with non-adaptive HTTP streaming, i.e., progressive download whereas our evaluation is based on systems already deployed by the industry and standards under development such as ISO/IEC MPEG and 3GPP. Therefore, the results of this evaluation [3] demonstrate that dynamic HTTP streaming is more suitable for mobile networks than non-adaptive HTTP streaming. In their previous work [4] they have also introduced bandwidth road maps to increase the adaption of their TCP based video streaming system, which can also be used to increase the precision of rate-adaption algorithms in dynamic HTTP streaming.

One of the first standards on how to handle varying bandwidth conditions with HTTP streaming has been proposed by 3GPP as Adaptive HTTP Streaming (AHS) [5]. The basic idea is to break up the media file into segments of equal length which can be encoded at different resolutions, bitrates, etc. The segments will be stored on an ordinary Web server and can be accessed through HTTP GET requests from the client. As a consequence, this streaming system is pull based and the entire streaming logic is on the client side. This means that the client fully controls the bitrate

MoVid'12, February 24, 2012, Chapel Hill, North Carolina, USA.

Figure 1. Bandwidth Traces

of the streaming media on a per segment basis which has several advantages, e.g., the client knows its bandwidth requirements and capabilities such as codecs, resolution, and language best. Furthermore, this system scales very well because the content can be simply distributed utilizing CDNs. In order to describe the relationship between the media segments and the corresponding bitrate, resolution, and timeline, 3GPP introduced the Media Presentation Description (MPD). The MPD is an XML document comprising HTTP uniform resource locators (URLs) which point to segments with individual capabilities, e.g., bitrate, resolution that correspond to an exact point in time. The MPD is typically the first object that will be downloaded for a dynamic HTTP streaming session and provides means to initialize the session. With the information provided through the MPD the client is able to start the session and dynamically adapt to bandwidth fluctuations, if needed. In addition to standardized solutions, a lot of proprietary solutions have been deployed by the industry, e.g., Microsoft Smooth Streaming [6], Adobe HTTP Dynamic Streaming [7] and Apple HTTP Live Streaming [8], but interestingly all of them utilize some kind of MPD and follow nearly the same architecture where the logic is located at the client side and the media will be chopped into segments and encoded at different bitrates or resolutions. Finally, ISO/IEC MPEG has started a new work item referred to as Dynamic Adaptive Streaming over HTTP (DASH) [9][10] which aims to combine the features of the above mentioned systems within a single standard. Such HTTP streaming systems are designed to handle varying bandwidth conditions which are very valuable, specifically within mobile environments due to the fact that these conditions could change dramatically from one moment to another. As DASH is a recent standard it is currently not clear how these systems perform in mobile environments and, thus, this paper evaluates the behavior of dynamic HTTP streaming systems in *mobile – i.e., vehicular – environments*. Hence, in the following the terms mobile and vehicular are used synonymously. Vehicular environments are more challenging than traditional mobile environments (e.g., pedestrians) because bandwidth fluctuations will occur more frequently and with higher amplitudes. Therefore, if a dynamic, HTTP streaming system performs well in vehicular environments it will also perform well in the traditional mobile environments.

In order to understand the problems and requirements of this highly specialized use case, this paper experimentally evaluates four streaming systems, i.e., Microsoft Smooth Streaming, Adobe Dynamic Streaming, Apple Live Streaming and our prototype implementation of MPEG-DASH. This paper addresses three fundamental questions:

1. How do these systems react on the high frequency bandwidth fluctuations?
2. Do they guarantee a smooth playback under these highly restricted bandwidth conditions?
3. Could they utilize the maximum available bandwidth with a minimum number of quality switches?

In anticipation of the results we can conclude that none of the systems achieves the maximum available bandwidth with a minimum number of quality switches. Furthermore, not all of them guarantee a smooth playback which is definitely the minimum criteria for a streaming system in that case.

The reminder of this paper is organized as follows. Section 2 describes our methodology while the results are presented and discussed in Section 3. The paper is concluded in Section 4.

2. METHODOLOGY

This section provides information about the experimental methodology and the metrics that are used in this paper. All experiments have been performed using Big Buck Bunny [11]. The content has been encoded using x264 [12] at 14 different bitrates (100, 200, 350, 500, 700, 900, 1100, 1300, 1600, 1900, 2300, 2800, 3400, and 4500 kbit/s). For the encoding a GOP size of 48 frames has been used, which is necessary to chop the video into segments of 2 seconds length which is required by Microsoft Smooth Streaming. The experimental setup for each evaluation comprises four devices: the client, the bandwidth shaping, the network emulation, and the Web server. These components will be further described in Section 3. All systems have been evaluated under three different network emulation settings that have been recorded during separate freeway car drives with a HUAWEI E169 HSPDA USB Stick using a SIM-card of the Austrian cellular network provider A1. Therefore, we used the A2 freeway in Carinthia/Austria shown in Figure 1 which has speed limits between 100 and 130 km/h:

- **Experiment 1 / Track 1 (601 seconds)**: Drive on the freeway A2, passing by the city of Villach in the direction to Klagenfurt.
- **Experiment 2 / Track 2 (575 seconds)**: From the Alpen-Adria-Universität Klagenfurt on the freeway A2 until the service area around Techelsberg.
- **Experiment 3 / Track 3 (599 seconds)**: From the service area around Techelsberg on the freeway A2 to the exit of Klagenfurt.

Wireshark has been used to capture the behavior of each system in a consistent way which simply records the HTTP GET requests and marks them with timestamps. Furthermore, four metrics have been used that utilize the data provided by Wireshark in order to make each system comparable: average used bitrate, number of quality switches, buffer level, and stalls. Each system will be evaluated using the traces from the three tracks and compared with the metrics briefly introduced in the following.

The **average bitrate** $\mu_{\mathbf{bitrate}}$ could be seen as the overall performance of the system at a particular test setup and will be computed with the equation below:

$$\mu_{\text{bitrate}} = \frac{\sum_{i=0}^{N} f(s_i) * t_i}{t_n}$$

$i \in [0, N] \ldots$ *Segment Index* $\quad t_i \ldots$ *Length of Segment* i

$t_n \ldots$ *Length of the Session* $\quad s_i \ldots$ *Segment* i

$f(s_i) \ldots$ *The function returns the Bitrate of Segment* i

The **number of quality switches** is another metric that describes the variance of the session. High values indicate very frequent switching which can lead to a decreased Quality of Experience (QoE) [13]. The following formulas have been used to calculate the number of bitrate switches:

$$\sigma^2 = \sum_{i=0}^{N} g(s_i) \qquad g(s_i) = \begin{cases} 1 & if \quad i = 0 \\ 1 & if \quad f(s_{i-1}) \neq f(s_i) \\ 0 & else \end{cases}$$

The **buffer level** must be estimated as a consequence that not all systems provide this information through an interface. The estimation used in this paper is based on two timestamps namely the download timestamp (DTS) and the presentation timestamp (PTS). The current buffer level in seconds can be estimated with the following formula:

$$b = [PTS_i - DTS_i]^+$$

Where $[x]^+$ denotes the maximum of x or 0. The DTS could be observed on the network emulation component, which uniformly captures and marks all request with a timestamp. This timestamp is called DTS. The PTS of each segment could be calculated as a consequence that the length of each segment is known (i.e., 2 seconds in our case).

The **number of unsmooth seconds metric** describes the smoothness of the session and will immensely influence the QoE. It could be derived from the buffer level metric and describes the time when the buffer is empty. Therefore, a high value of unsmooth seconds indicates a more jerky session.

3. EXPERIMENTS

The architecture of the experimental setup is depicted in Figure 2 and consists of four devices, i.e., *evaluation client*, *bandwidth shaping*, *network emulation*, and *HTTP server*. The main components of this architecture are the bandwidth shaping and the network emulation nodes which are both based on Ubuntu 11.04. The bandwidth shaping node controls the maximum achievable bandwidth for the client with the Linux traffic control system (tc) and the hierarchical token bucket (htb) which is a classfull queuing discipline (qdisc). The available bandwidth for the client will be adjusted every 2 seconds due to the recorded bandwidth traces. The 2 seconds interval has been chosen as a consequence of the segment length which is required by Microsoft Smooth Streaming. The network emulation node controls all network related parameters such as round trip time (RTT). Based on our measurements the RTT has been set to 150 ms [14] with the Linux Network Emulator (netem). The client and server components of this architecture as well as the content generation tools vary from one evaluation to another simply because not every system is platform independent and there is no universal tool to generate content for each individual system.

3.1 Microsoft Smooth Streaming

The evaluation setup for Microsoft's Smooth Streaming (MSS) is based on Windows 7 and Microsoft Silverlight. Furthermore,

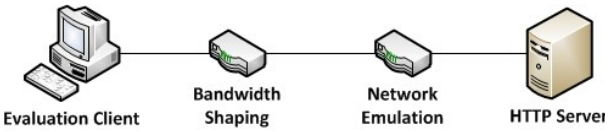

Figure 2. Experimental Setup

Mozilla Firefox 7 has been used consistently on the client within all experiments. The server component is based on Windows Server 2008 and the Internet Information Service (IIS) with Media Services 4.0. The multiplexing of the previously encoded content is performed through the IIS Transform Manager (IISTM) 1.0 Beta [15] which transforms .mp4 files to so-called "H.264 Smooth Streams", i.e., segmented .mp4 files. Additionally, it generates the corresponding metadata client and server manifest files) which are XML-based and comparable with the MPD that has been described in the introduction. Unfortunately, Microsoft's manifest files do not contain fully qualified URLs which restricts MSS to IIS Web servers. Hence, the IIS Web server must transform all requests for media segments that will be sent during a dynamic HTTP streaming session. Each request that will be produced by the MSS client contains the video bitrate and a timestamp that corresponds to the presentation time of the segment (PTS). The bandwidth emulation server provides the download timestamp (DTS) and the buffer level is estimated as described in the previous section.

Due to page count limit, Figure 3 shows the behavior of the MSS client for experiment 3 / track 3 only. For the results of the other experiments/tracks, the interested reader is referred to [16]. Figure 3(a) shows the *captured* and, thus, available bandwidth compared to the throughput, i.e., utilized bandwidth at the client (*adaptation*) and Figure 3(b) shows the *buffer fill state*. Interestingly, MSS maintains the same maximum buffer level over all experiments/tracks which is approximately 30 seconds. All experiments start with a very high bandwidth that is around or over 4Mbit. The adaptation process seems to recognize this and starts to increase the quality in a stepwise manner. The advantage of this stepwise approach is that the quality will be increased much more smoothly which could potentially increase the QoE. This stepwise state transition is typical for Microsoft's system only if the measured bandwidth decreases dramatically, e.g., around second 100, where the adaption process decreases the quality with bigger steps to guarantee a smooth playback. Generally speaking, MSS acts very conservative which is not a

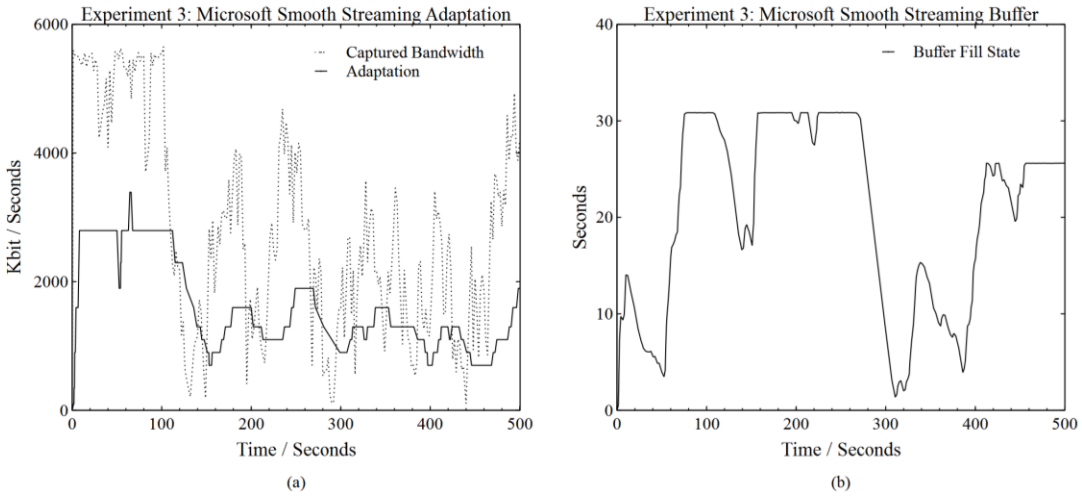

Figure 3. Microsoft Smooth Streaming

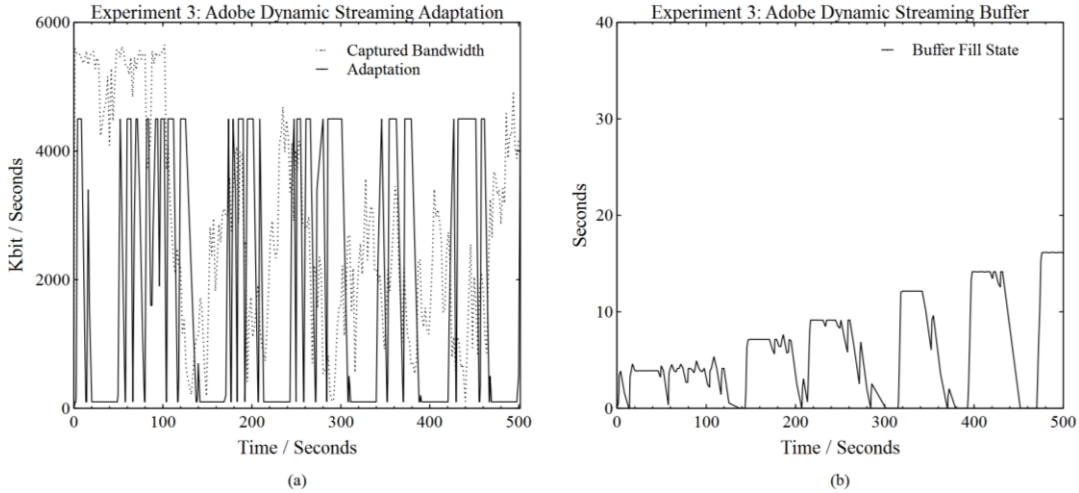

Figure 4. Adobe Dynamic Streaming

bad idea especially for our experiments which have high bandwidth fluctuations but it does not react on very short notice, e.g., experiment 2 / track 2 around second 300 [16]. Furthermore, it seems that the adaptation process maintains something like a safety margin. This means that the measured bandwidth must be significantly and continuously above the bitrate of a representation until the client will eventually choose this representation.

3.2 Adobe HTTP Dynamic Streaming

The client for Adobe HTTP Dynamic Streaming (ADS) is based on Ubuntu 11.04, Firefox 7, and the Open Source Media Framework (OSMF) Player [17]. The server component hosts the Flash Media Server in development edition [18] and content generation has been achieved through Adobe's File Packager for ADS. The video that has been used for MSS has been encoded at the same bitrates and resolutions for ADS to be consistent.

The Adobe adaptation process is very unpredictable and is depicted in Figure 4 for experiment 3 / track 3. In comparison to Microsoft's system ADS is very aggressive and does not act in a stepwise manner. Interestingly, it switches most of the time between the highest and the lowest representation even if the bandwidth for the highest representation is not available over a longer time span, e.g., between second 300 and 450 or for experiment 1 / track 1 between second 100 and 350 [16]. Even if the bandwidth for the highest representation is available, e.g., at the beginning of the session, ADS does not use this representation continuously due to the small buffer size at the beginning. ADS handles such stalls by simply increasing the playback buffer in a linear way. Every time when a stall occurs, e.g., jerky playback, the buffer will be increased by a fixed value. This idea of a "learnable" buffer is smart but from our point of view not consequently implemented, i.e., it would be better to use an exponential increase rather than a linear one. However, ADS with the OSMF player is definitely not suitable for mobile networks due to the fact that it behaves unpredictable and more binary, switches between the highest and the lowest representation, than smooth. Furthermore, it does not guarantee a smooth playback and introduces a serious number of stalls followed by re-buffering which potentially annoys the user and, therefore, decrease the QoE tremendously.

3.3 Apple HTTP Live Streaming

The Apple HTTP Live Streaming (HLS) client is based on Mac OS X Snow Leopard 10.6 and the Safari 5 Web browser. Fortunately, the Microsoft Transform Manager offers a possibility to transform "H.264 Smooth Streams to Apple HTTP Live Streams" which trans-multiplexes the .mp4 based smooth streams to MPEG-2 Transport Streams (TS). The TS will be chopped into segments with a length of 2 seconds, instead of 10 seconds which is usually required by HLS. HLS is the only system that uses MPEG-2 TS instead of .mp4 files or another ISO Base Media File Format (IBMFF) based container which will add a significant overhead of approximately 25% in relation to the audio/video data

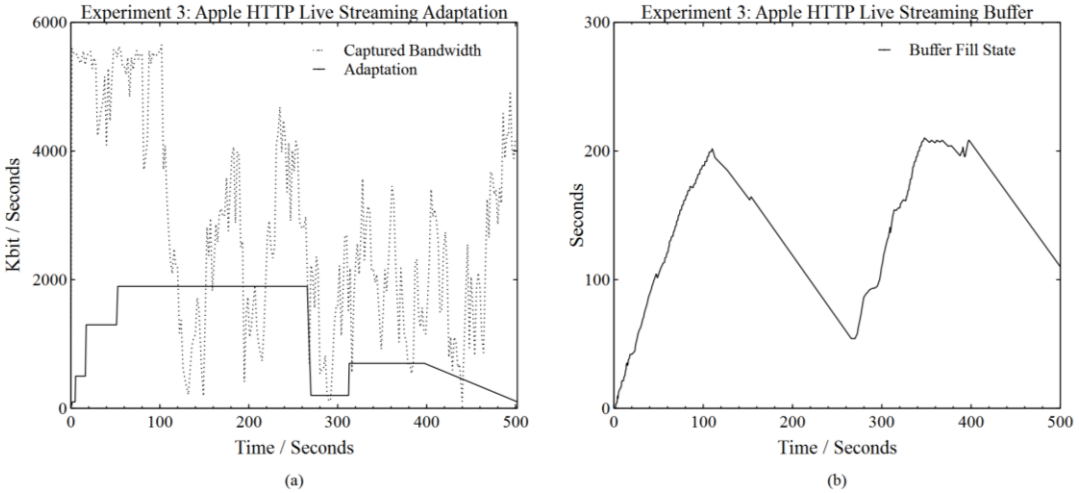

Figure 5. Apple Live Streaming

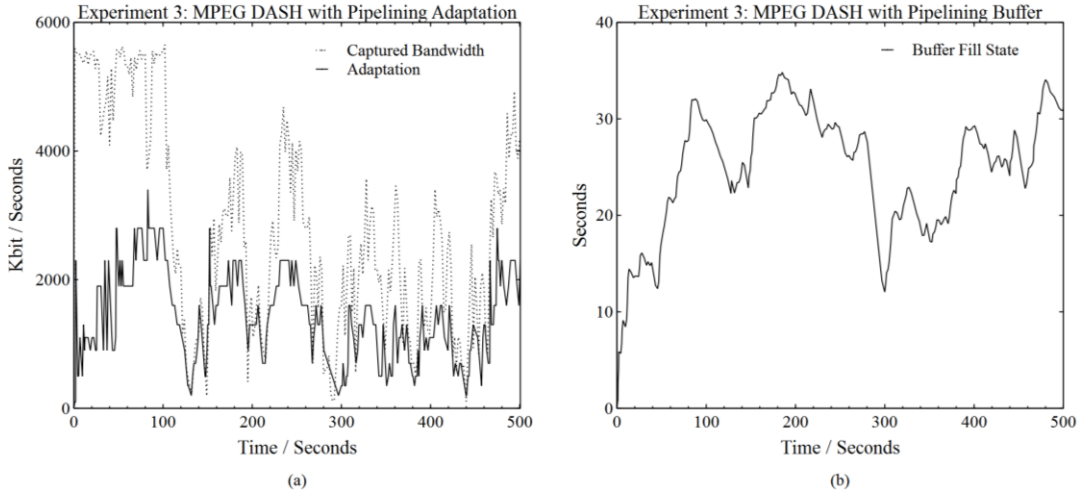

Figure 6. MPEG DASH with Pipelining

[19]. The server for this experiment is based on Windows Server 2008 which hosts the IIS Web server that provides the Apple HLS streams. In contrast to the other systems Apple HLS has been especially designed for mobile environments and can also bundle requests. This means that it could request more than one segment with one request which is a kind of pipelining. These features could definitely lead to a more efficient use of the connection and should be considered by new adaptation logics. The results of the Apple HLS experiment 3 / track 3 are depicted in Figure 5. Apple HLS also uses the stepwise approach for representation transitions like Microsoft but it seems that the step size is larger compared to MSS. Furthermore, also the buffer size seems to be very large (approx. 200-250 seconds). When the buffer reaches its maximum Apple HLS issues a bundle request for 75 segments (i.e., 150 seconds) which corresponds to second 100 to 250 in Figure 5 (experiment 1 / track 1 second 300 to 450, experiment 2 / track 2 second 100 to 250). This bundled request always leads to buffer decrease in the three experiments as a consequence that Apple HLS does not consider the bandwidth fluctuations in-between this bundled request. However, due to the huge buffer size it can compensate this "false prediction" during periods with extremely low bandwidth.

3.4 Dynamic Adaptive Streaming over HTTP

The MPEG-DASH experiment is based on our implementation [20] comprising a DASHEncoder [21], which generates the content and a DASH compliant MPD, and the DASH VLC Plugin (DCP) [22]. The server for this experiment is based on Ubuntu 11.04 which hosts an Apache Web server. The DCP has been modified by adding a 30 second buffer (i.e., 15 segments) to compensate high bandwidth fluctuations. In order to avoid the reconnection after each segment the DCP uses HTTP/1.1 persistent connections. The adaptation algorithm that is used by the DCP simply measures the download time of each segment and build an adaptation decision out of this download time and the average measured bitrate of the whole session. This process is depicted in the following formulas where *maxbw(s$_i$)* returns the maximum bandwidth that is available for *segment i*, as a consequence the maximum quality that also guarantees a smooth playback:

$$maxbw(s_i) = \begin{cases} bw(s_{i-1}) * 0.3 & if \quad 0.00 \le bl_i < 0.15 \\ bw(s_{i-1}) * 0.5 & if \quad 0.15 \le bl_i < 0.35 \\ bw(s_{i-1}) & if \quad 0.35 \le bl_i < 0.50 \\ bw(s_{i-1}) * (1 + 0.5 * bl_i) & if \quad 0.50 \le bl_i \le 1 \end{cases}$$

$i \in [1, N]$ … *Segment index*

bl_i … *Buffer level at DTS of segment i*

$bw(s_i)$ … *The function returns the bandwitdth which was measured during the download of segment i*

In comparison to Microsoft and Apple the DCP uses the non-stepwise approach, depicted in Figure 6, like Adobe which leads to more representation transitions but could potentially utilize a higher average bitrate. Furthermore, the DCP guarantees a smooth playback with a more or less constant bitrate. A potential improvement which has been tested is the use of HTTP/1.1 pipelining that could compensate the relatively high RTTs in mobile networks. Our experiments have shown that such an improvement could increase the average bitrate of the DCP by approximately 35%.

3.5 Comparison

Table 1 depicts from left to right the average bitrate, average number of representation switches, and the average number of unsmooth seconds of all systems. The average bitrate has been calculated over all three experiments and shows that Microsoft Smooth Streaming (MSS) performs very well. Furthermore Microsoft's adaptation logic achieves these results with fewer switches than Adobe or our prototype implementation of MPEG-DASH. Only Apple HTTP Live Streaming (HLS) has a lower switch count but it could not utilize the same bitrate as MSS. Adobe Dynamic Streaming (ADS) is the only system that did not guarantee a smooth playback. The average number of unsmooth seconds column shows that Adobe introduces more than 60 seconds which is over 10% of the session where the playback is

Table 1. Comparison

Name	Average Bitrate	Average Switches	Average Unsmoothness
Unit	[kpbs]	[Number of Switches]	[Seconds]
Microsoft	1522	51	0
Adobe	1239	97	64
Apple	1162	7	0
DASH	1045	141	0
DASH Pipelined	1464	166	0

unsmooth. Moreover this unsmoothness is spread over the whole sessions and occurred once per minute which simply makes the streaming session unwatchable. Our prototype implementation of MPEG-DASH performs surprisingly well especially with HTTP/1.1 pipelining it achieves the second best average bitrate. The improvement of our implementation will be part of our further research.

4. CONCLUSION

This paper provides a detailed evaluation of state-of-the-art proprietary dynamic HTTP streaming solutions: Microsoft Smooth Streaming, Adobe Dynamic Streaming, Apple HTTP Live Streaming, and our prototype implementation of the emerging MPEG-DASH standard. We have evaluated these systems under real world mobile network conditions using bandwidth traces that have been captured under vehicular mobility. Microsoft Smooth Streaming performs very well in this scenario and achieves the highest average bitrate as well as the second lowest number of stream switches. Apple Live HTTP Steaming has been especially designed for video transmission to Apple devices such as iPhone and iPad. Interestingly, it is the only system that uses MPEG-2 TS which adds an additional overhead in comparison to the ISOBMFF-based containers that have been used by the other evaluated systems. Furthermore, it utilizes the lowest overall bitrate compared to the other commercial systems. Adobe's Dynamic Streaming is the only system that does not achieve a smooth playback. Additionally, the adaptation process does not behave very predictable as it is more a binary decision between the highest and the lowest representation which lead in combination with several stalls and long re-buffering periods to low QoE. Our prototype implementation of MPEG-DASH shows promising results indicating the capabilities of the standard and could definitely compete with the commercial systems. Furthermore, it achieves the minimum criterion which is a smooth playback and the second best overall bitrate. Improving this adaptation process with the aim to maximize the QoE will be part of our future work.

5. ACKNOWLEDGMENTS

This work was supported in part by the EC in the context of the ALICANTE (FP7-ICT-248652), SocialSensor (FP7-ICT-287975), and QUALINET (COST IC 1003) projects.

6. REFERENCES

[1] B. Wang, J. Kurose, P. Shenoy, D. Towsley, "Multimedia Streaming via TCP: An Analytic Performance Study", *ACM Transactions on Multimedia Computing, Communication and Applications,* vol. 4, no. 2, May 2008, pp. 16:1-16:22.

[2] S. Akhshabi, A. Begen, C. Dovrolis, „An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP", *ACM Multimedia Systems*, San Jose, California, USA, Feb. 2011, pp. 157-168.

[3] J. Yao, S. Kanhere, I. Hossain, M. Hassan, "Empirical evaluation of HTTP adaptive streaming under vehicular mobility", *Proceedings of the 10th international IFIP TC 6 conference on Networking (Networking'11),* Valencia, Spain, May 2011, pp. 92-105.

[4] J. Yao, S. Kanhere, M. Hassan, "Quality Improvement of Mobile Video Using Geo-Intelligent Rate Adaptation", Wireless Communications and Networking Conference (WCNC 2010), Sydney, Australia, April 2010, pp. 1-6.

[5] 3GPP TS 26.234, "Transparent end-to-end packet switched streaming service (PSS)", Protocols and codecs, 2010.

[6] Microsoft Smooth Streaming, http://www.iis.net/download/smoothstreaming (last access: Dec., 2011).

[7] Adobe HTTP Dynamic Streaming, http://www.adobe.com/products/httpdynamicstreaming/ (last access: Dec., 2011).

[8] R. Pantos, W. May, "HTTP Live Streaming", IETF draft, http://tools.ietf.org/html/draft-pantos-http-live-streaming-07 (last access: Dec, 2011).

[9] ISO/IEC DIS 23001-6. 2011, Information technology – MPEG systems technologies – Part 6: Dynamic adaptive streaming over HTTP (DASH), http://mpeg.chiariglione.org/working_documents/mpeg-b/dash/dash-dis.zip (last access: Dec, 2011).

[10] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP – Standards and Design Principles", *ACM Multimedia Systems*, San Jose, California, USA, Feb. 2011, pp. 133-143.

[11] Big Buck Bunny Movie, http://www.bigbuckbunny.org (last access: Dec. 2011).

[12] X264, http://www.videolan.org/developers/x264.html, (last access: Dec. 2011).

[13] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, P. Halvorsen, "Spatial Flicker Effect in Video Scaling", Proceedings of the third international Workshop on Quality of Multimedia Experience (QOMEX'11), Mechelen, Belgium, Sept. 2011, pp. 55-60.

[14] P. Romirer-Maierhofer, A. Coluccia, T. Witek, "On the Use of TCP Passive Measurements for Anomaly Detection: A Case Study from an Operational 3G Network", *Traffic Monitoring and Analysis Workshop TMA 2010*, Zürich, Switzerland, April 2010, pp. 183 – 197.

[15] Transform Manager 1.0 Beta, http://www.iis.net/download/TransformManager, (last access: Dec. 2011).

[16] Additional Results, http://www-itec.uni-klu.ac.at/dash/movid/additional_results.pdf (last access: Dec. 2011).

[17] Getting Started with OSMF for Developers, http://www.osmf.org/developers.html, (last access: Dec. 2011).

[18] Flash Media Server Developer Center, http://www.adobe.com/devnet/flashmediaserver.html, (last access: Dec. 2011).

[19] H. Riiser, P. Halvorsen, C. Griwodz, D. Johansen, "Low Overhead Container Format for Adaptive Streaming", *ACM Multimedia Systems*, Phoenix, Arizona, USA, Feb. 2010, pp. 193-198.

[20] DASH at Alpen-Adria-Universität Klagenfurt, http://www.itec.uni-klu.ac.at/dash (last access: Dec. 2011).

[21] S. Lederer, C. Mueller, C. Timmerer, "Dynamic Adaptive Streaming over HTTP Dataset", *ACM Multimedia Systems*, Chapel Hill, North Carolina, USA, Feb. 2012.

[22] C. Müller, C. Timmerer, "A Test-Bed for the Dynamic Adaptive Streaming over HTTP featuring Session Mobility", *ACM Multimedia Systems*, San Jose, California, USA, Feb. 2011, pp. 271-276.