A PROXY EFFECT ANALYIS AND FAIR ADATPATION ALGORITHM FOR MULTIPLE COMPETING DYNAMIC ADAPTIVE STREAMING OVER HTTP CLIENTS

Christopher Mueller, Stefan Lederer, and Christian Timmerer

Multimedia Communication (MMC) Research Group
Institute of Information Technology (ITEC)
Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria
Email: *{firstname.lastname}*@itec.aau.at

ABSTRACT

Multimedia streaming technologies based on the Hypertext Transfer Protocol (HTTP) are very popular and used by many content providers such as Netflix, Hulu, and Vudu. Recently, ISO/IEC MPEG has ratified Dynamic Adaptive Streaming over HTTP (DASH) which extends the traditional HTTP streaming with an adaptive component addressing the issue of varying bandwidth conditions that users are facing in networks based on the Internet Protocol (IP). Additionally, industry has already deployed several solutions based on such an approach which simplifies large scale deployment because the whole streaming logic is located at the client. However, these features may introduce drawbacks when multiple clients compete for a network bottleneck due to the fact that the clients are not aware of the network infrastructure such as proxies or other clients. This paper identifies these negative effects and the evaluation thereof using MPEG-DASH and Microsoft Smooth Streaming. Furthermore, we propose a novel adaptation algorithm introducing the concept of fairness regarding a cluster of clients. In anticipation of the results we can conclude that we achieve more efficient bottleneck bandwidth utilization and less quality switches.

Index Terms—Dynamic Adaptive Streaming over HTTP, DASH, Fair Adaptation, Proxy Cache, Multimedia

1. INTRODUCTION

Multimedia is nowadays ubiquitous in the Internet and many Web applications are also using the Hypertext Transfer Protocol (HTTP) for multimedia streaming by adopting progressive download. However, there are certain disadvantages when using HTTP for multimedia streaming. In particular, this protocol has been initially designed for best effort and not for real-time multimedia transport. A major problem of HTTP streaming approaches adopting progressive download is that it is not able to handle varying bandwidth conditions of IP-based networks. Therefore, ISO/IEC MPEG has recently ratified an advancement of that basic HTTP streaming approach which is referred to as Dynamic Adaptive Streaming over HTTP (DASH) [1]. In comparison to the traditional HTTP streaming this approach is able to handle the varying bandwidth conditions while maintaining its advantages such as NAT/firewall traversal and flexible deployment. Additionally, major industry players including Microsoft [2], Apple [3], and Adobe [4] have deployed their solutions and, interestingly, all are based on this same principles. That is, the streaming logic is located at the client and multiple versions of the content, e.g., different resolution, bitrate, etc. have been segmented and stored on legacy Web servers. While it is obvious that this approach scales very well, it may introduce some new drawbacks as a consequence that the streaming logic is located at the client, i.e., clients are not aware of each other and the network infrastructure such as proxy caches.

This paper concentrates on the negative effects introduced when multiple clients are competing for a bottleneck and how proxies are influencing this bandwidth competition. As mentioned above, the clients request individual portions of the content based on the available bandwidth which is calculated using throughput estimations. A consequence of this requesting scheme is that only some parts of the content are stored on proxy servers, which are intercepting the connection between the client and the content server. This uncontrolled distribution of the content influences the adaptation process that assumes that the measured throughput is the throughput to the content server. The impact of this falsified throughput estimation could be tremendous and leads to a wrong adaptation decision which may impact the Quality of Experience (QoE) at the client.

In anticipation of the results we can conclude that this false interpretation of the throughput estimation introduces unnecessary frequent quality switches and could produce an unsmooth session which immensely decreases the QoE [5]. Furthermore, the bottleneck will be more stressed than needed which could influence other applications.

The remainder of this paper is organized as follows. Related work is described in Section 2 while Section 3 highlights potential scenarios where this effect could occur. Our adaptation logic which handles such falsified throughput estimations is described in Section 4. The methodology of our experiments is described in Section 5

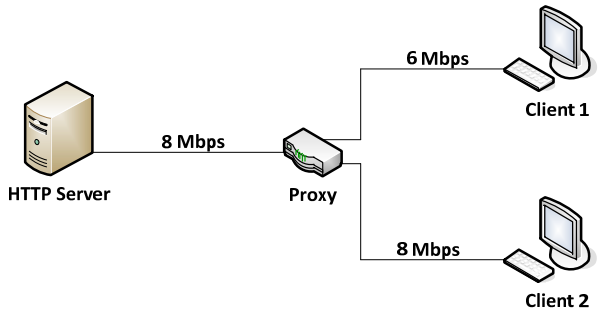

Figure 1. DASH Proxy Scenario

and the evaluation in Section 6. The paper is concluded with Section 7.

2. RELATED WORK

Evensen et al. [7] has proposed a system that could use multiple heterogeneous access networks. However, this system is based on throughput estimations that do not take the network infrastructure or other, competing clients into account. Liu et al. [8] has used a smoothed HTTP throughput for their rate adaptation algorithm that does not consider the negative proxy effects or the locality of the segments. Kuschnig et al. [9] has evaluated existing algorithms in terms of quality using PSNR, however no negative proxy effects have been considered. Houdaille et al. [10] has shown that shaping of HTTP adaptive streams could increase the user experience. However, their design needs a network bandwidth shaper at each home but they do not consider the proxy effects and the locality of the segments. Mok et al. [11] has proposed a QoE-aware DASH system which shows that users prefer a gradual quality change. Their adaptation logic is based on that finding and the measured throughput but like the others they do not take the locality of the segments and the network infrastructure, e.g., proxies and other clients into account.

3. DASH-BASED PROXY EFFECTS

This section describes problems that could occur during a DASH session when multiple clients are competing for a limited bottleneck bandwidth. The scenario depicted in Figure 1 consists of a content server, a proxy, and two clients that are competing for the bottleneck bandwidth, i.e., 8 Mbps between the content server and the proxy. Client 1 is connected over a 6 Mbps link with the proxy server and Client 2 is connected over an 8 Mbps link with the proxy. The content server hosts a DASH session with two quality levels that will be consequently called base and enhancement quality in this section. The base quality has a bitrate of 5 Mbps and the enhancement quality has a bitrate of 7 Mbps.

Due to the fact that the bandwidth of Client 1 is restricted to 6 Mbps, it will only select segments of the base quality during the whole streaming session. Assuming that this client will start the streaming slightly before Client 2 implies that the base quality will be cached on the proxy. Client 2 will start the streaming session and request the base quality at the beginning to minimize the startup delay and to fill its buffer as fast as possible. At this point the proxy maintains one connection to the content server, which could utilize the full available bandwidth of 8 Mbps (i.e., both clients are using the same quality). Subsequently after Client 2 has stabilized its buffer, it will try to adapt to the maximal available bandwidth due to the throughput that has been measured on previous segments. Obviously, the measured throughput at Client 2 is 8 Mbps because the segments of the base quality which Client 2 is streaming at this point are cached at the proxy server, due to the selection scheme of Client 1. The consequence of this throughput estimation is that Client 2 will switch to the enhancement quality, because it assumes that there is enough bandwidth available to stream this quality level in a smooth way. The problem is now that the proxy has to maintain two connections to the content server, i.e., one for the base quality and one for the enhancement quality.

We are assuming that these two connections will be shared more or less in a fair way, which means that each connection could utilize 4 Mbps of the bottleneck bandwidth. Obviously it is not possible to stream the base quality smoothly with 4 Mbps, which means that this quality switch at Client 2 will influence Client 1 and could produce an unsmooth session at Client 1. Moreover the throughput at Client 2 will collapse to 4 Mbps so that this client will switch down to the base quality, which means that the proxy will close the second connection and maintain only one connection with 8 Mbps for the base quality. Afterwards the base quality is cached at the proxy server, due to Client 1 and Client 2 will switch to the enhancement quality again. This effect will occur over the whole streaming session and therefore the clients are negatively influencing each other without any changes in the network conditions. This will decrease the QoE of both clients due to the frequent quality switching [12] at Client 2 and potential unsmooth playback at Client 1 [5].

4. FAIR ADAPTATION

Our fair adaptation scheme (FAS) aims to address the problem identified in Section 3. Our first and probably simplest approach to decrease the frequent switching and as a consequence the negative effects, that could be caused due to that switching, is an adaptation logic with an exponential backoff. This approach decreases the number of switch up points if a switch down occurs. But this technique does not consider whether a bandwidth fluctuation is self-caused or network caused. As described in Section 3, self-caused bandwidth fluctuations, i.e., frequent quality switching, get introduced because the adaptation logic does not consider the uncontrolled distribution of segments over the proxy

```
if backoff > 0
  backoff := backoff - γ
endif
quality_level := find (measured_bandwidth)
if quality_level > quality_last_segment
  if backoff <= 0
    if probe(quality_level)
      count := 0
    else
      backoff := (int) α * e^(β * count)
      count := count + δ
      quality_level := quality_last_segment
    endelse
  else
    quality_level := quality_last_segment
  endelse
endif
return quality_level
```

Algorithm 1. Exponential Backoff with Probe.

caches. Obviously, these negative effects only occur when a client switches to a higher quality level due to a wrong interpretation of the throughput estimation. Therefore, we have used a probe method, which will be further described at the end of this section, with the previously mentioned exponential backoff [13]. Every time when the adaptation logic identifies a valuable switch up point, that is also permitted by the exponential backoff, we do a kind of double-check. The probe method will then identify the effective available bandwidth for the next segment.

Algorithm 1 depicts our adaptation logic that returns the quality level for the next segment. The backoff could be adjusted to the network characteristics with the parameters α and β. In our experiments we set them to 1 for simplicity reasons. Additionally, it is possible to accelerate or decelerate the backoff process with the parameters γ and δ. Furthermore, this algorithm uses the previously mentioned probe method to identify the effective available bandwidth for the next segment. This means that every adaptation decision which leads to a switch up will be verified.

In the following we will briefly describe the different techniques which can be used for the probing method to identify the effective available bandwidth:

1. The server could provide a non-cacheable object. This object guarantees that the bandwidth to the server will be measured due to the fact that it will not be stored on any proxy. Hence, it can be estimated if enough bandwidth is available for a given quality.
2. The client could simply download the first few bytes or a random byte range of the next segment to estimate the effective available bandwidth. This method works very well even for multiple clients since most proxies do not cache byte range requests.
3. The proxy server could actively modify the MPD and remove the qualities that could not be served due to bandwidth limitations.
4. The proxy server could offer a service that provides information about the effective available bandwidth.

We have decided to use method 2 for our system as it does not require any changes on the network side. This is very important because one of the major advantages of DASH is that it could be deployed over the top of existing infrastructures and does not take care of the underlying network elements, i.e., proxies, caches and CDNs.

5. METHODOLGY

This section describes the methodology and metrics that have been used to evaluate the proxy effects. We have used Big Buck Bunny [14] for all experiments and the content has been encoded with x264 [15] with a GOP size of 48 frames which is necessary to provide a uniform length of 2 seconds for each segment. The length is restricted by Microsoft Smooth Streaming (MSS) which only supports segments with that length.

We used the metrics from [16] for our experiments which are continuously captured. The first metric is the *average bitrate* that could be seen as the overall performance of the system at a particular test setup. The *number of quality switches* describes the variance of the session, where high values indicate very frequent switching which can lead to a decreased Quality of Experience (QoE) [12]. In addition to this the *buffer level* describes the current fill state of the buffer. We have measured it based on the download timestamp of the segments at the proxy and the presentation timestamp of each segment. Finally the *number of unsmooth seconds* metric describes the smoothness of the session and will immensely influence the QoE [5]. 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.

6. EXPERIMENTS

The architecture of our evaluation network is depicted in Figure 2 and consists of five elements namely, *HTTP Server*, *Proxy*, *Shaper*, *Client 1*, and *Client 2*. The proxy and the shaper are both based on Ubuntu 10.04 and are used for all experiments with the same configuration. The shaper controls the bandwidth of the clients with the Linux traffic control system (tc). Furthermore, the hierarchical token bucket (htb) has been used which is a classfull queuing discipline (qdisc). The available bandwidth for both clients remains static over the whole evaluation, i.e., 1100 Kbps for client 1 and 2200 Kbps for client 2. The proxy is based on the Squid [17] caching proxy in transparent mode. Furthermore, it also limits the bandwidth to the shaper with tc and htb, i.e., the bottleneck bandwidth. All evaluations have been performed with the same content, i.e., Big Buck Bunny at 700 Kbps and 1300 Kbps.

Please note that for this experiment the available bandwidth will not change during the whole streaming session. However, dynamic bandwidth conditions may influence the negative effects even more, e.g., the client

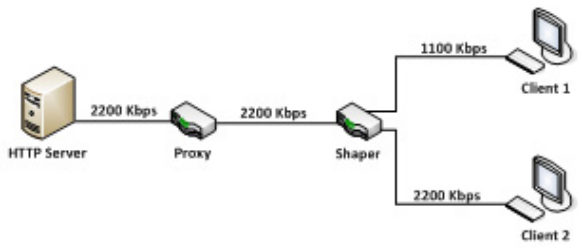

Figure 2. Experimental Setup

makes an unfavorable adaptation decision when the network bandwidth drops. These evaluations under dynamic bandwidth conditions will be part of our further research.

6.1. Microsoft Smooth Streaming

For the evaluation of the Microsoft Smooth Streaming (MSS), the content server was based on Microsoft Windows Server 2008. The client was based on Windows 7 and has used Silverlight 5 and Microsoft Internet Explorer for the playback.

Figure 3 shows the behavior of MSS for both clients. Figure 3 *(a)* shows the adaptation process, Figure 3 *(b)* shows the behavior of the proxy server and the cache hits for each request, and Figure 3 *(c)* shows the buffer fill status, which corresponds to the *buffer level* metric. Interestingly, MSS behaves exactly like assumed, i.e., Client 2 is constantly switching between the 700 Kbps and 1300 Kbps quality, due to the false interpretation of the available throughput, which leads to a high *number of quality switches* metric and thus a decreased QoE [12]. Furthermore, Figure 3 *(b)* shows that this switching is related to the proxy. Every time when Client 2 requests the 700 Kbps quality it gets it directly from the proxy server which is indicated as a proxy hit in Figure 3 *(b)*. Subsequently, Client 2 will measure an available bandwidth of 2200 Kbps for that segment as a consequence that this segment is cached at the proxy server. Afterwards, Client 2 will switch to the 1300 Kbps quality due to the fact that it assumes that a bandwidth of 2200 Kbps is effectively available which implies that Client 2 would be able to stream the 1300 Kbps quality level smoothly. At this point the proxy has to maintain two connections to the content server, i.e., one for the 700 Kbps quality and one for the 1300 Kbps quality. We are assuming that both connections share the bottleneck in a fair way, which means that each connection would be able to utilize 1100 Kbps. This assumption implies that it is obviously not possible to stream the 1300 Kbps quality with an effective bandwidth of 1100 Kbps. Therefore, Client 2 will switch down to the 700 Kbps quality as a consequence of the measured throughput of approximately 1100 Kbps. Now the proxy can close the second connection and the full bottleneck bandwidth could be utilized for the 700 Kbps quality. Afterwards Client 2 will get the segments of the 700 Kbps quality from the proxy and the measured bandwidth will be 2200 Kbps so that the previously mentioned behavior occurs periodically during the whole session. Due to Figure 3 *(a)* and *(b)* where Client 2 switches subsequently after selecting the 700 Kbps quality and generating a proxy hit to the 1300 Kbps quality, our assumption could be seen as validated.

6.2. MPEG-DASH

This section describes the evaluation of our assumption on negative proxy effects with the MPEG-DASH client from [6] which is based on the well know VideoLan VLC media player. The content server for this experiment is based on Ubuntu 10.04 and the Apache Webserver. We have used the same content for this experiment, which we have already used for the MSS experiment, i.e., 700 Kbps and 1300 Kbps quality. However, due to the more efficient implementation (e.g., using persistent connections and pipelining) which is not used by MSS, we had to modify the network conditions from Figure 2, otherwise both clients were able to stream

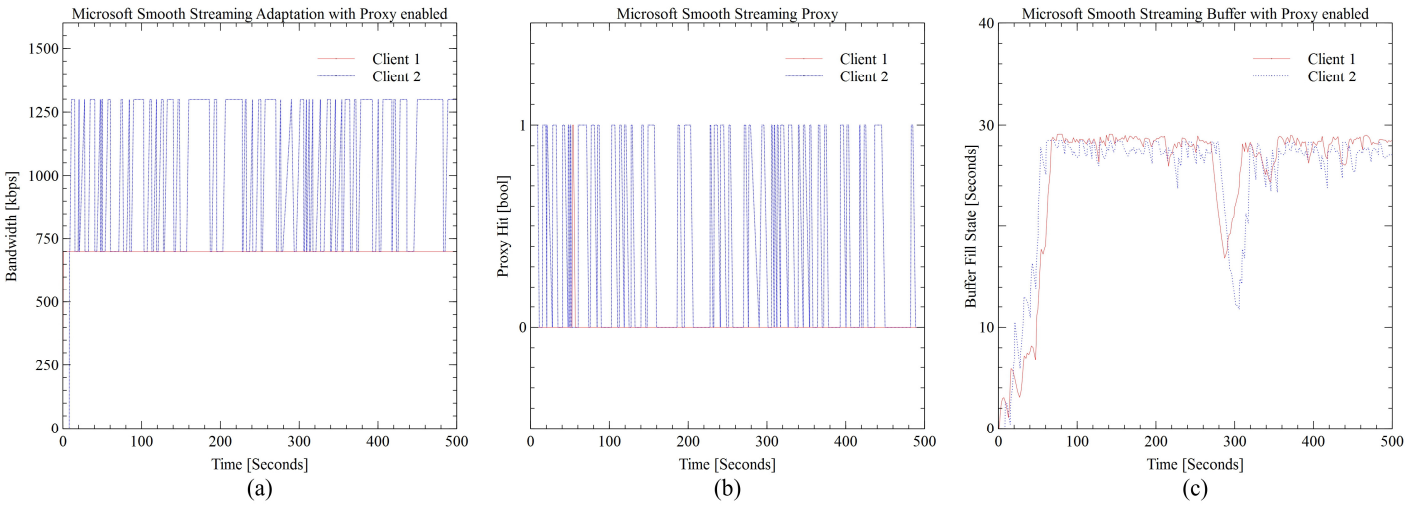

Figure 3. Microsoft Smooth Streaming

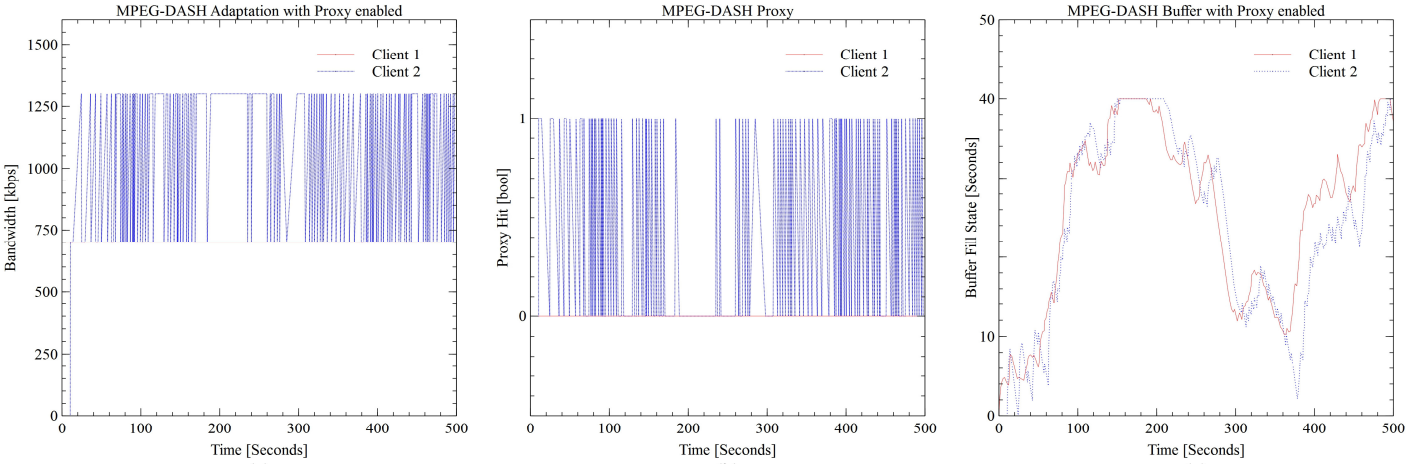

Figure 4. MPEG-DASH

the 700 Kbps and 1300 Kbps qualities smoothly in parallel. Therefore, we have decreased the bottleneck bandwidth to 1700 Kbps and the connection between Client 1 and the proxy to 1000 Kbps, as well as the connection between the Client 2 and the proxy has been decreased to 1700 Kbps. The results of this evaluation are organized like the MSS experimental results and presented in Figure 4. Interestingly, the quality switching effect occurs much more frequently at Client 2 compared to MSS (cf. Figure 4 *(a)*). This is due to the more aggressive adaption process of their implementation. Additionally, Client 2 produces an unsmooth playback like shown at second 20 of Figure 4 *(c)*. Overall, the system behaves nearly equal to MSS and also validates our assumption that the uncontrolled distribution of the media content could negatively influence streaming clients that do not take these effects into account.

6.3. Fair Adaptation

On top of the MPEG-DASH VLC plugin from [6] we have implemented our own fair adaptation logic thanks to its extensible design. The experimental setup, i.e., server and clients as well as the available bandwidths are the same like in the MPEG-DASH experiment of Section 6.2.

Figure 5 is organized like the MSS and MPEG-DASH figures and depicts the evaluation of our fair adaptation logic. The main improvement of this logic is the probe method that has been described in Section 4.1. We have used the probe method in combination with an exponential backoff. In particular, every time when the adaptation logic identifies a switch up point due to throughput estimations and the probe method has shown that this was a false interpretation of the effective available throughput, we increase the distance to the next potential switch up point exponentially. This is needed due to the fact that the probing needs streaming bandwidth which will be wasted if every segment gets probed.

The probe points are depicted with green vertical lines in Figure 5 *(a)* which also shows that the fair adaptation logic eliminates this frequent switching effect at Client 2 which results in a significantly lower *number of quality switches* metric for this experiment. Furthermore, it

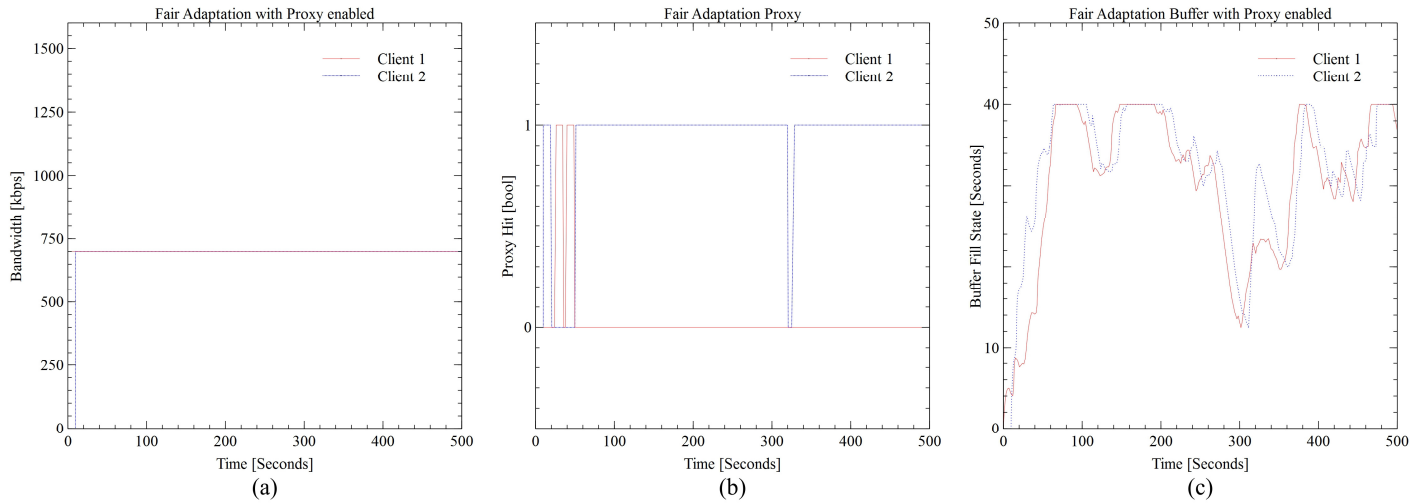

Figure 5. Fair Adaptation

enhances the cache reuse performance shown in Figure 5 *(b)* and it decreases the used bottleneck bandwidth which is especially important for other applications that are using this bottleneck beside the two streaming applications. Moreover, the unsmoothness of Client 2 from the MPEG-DASH experiment has been avoided, i.e., the buffer in Figure 5 *(c)* does not reach zero and both clients are maintaining a more stable buffer. This is especially important when it comes to the dynamic case which is part of our further research. Currently, the bottleneck bandwidth stays the same over the whole experiment but dynamic bandwidth fluctuations could further increase the negative effects due to the fact that a false segment selection could occur at a point in time when the bottleneck bandwidth decreases which could lead to an unsmooth playback.

7. CONCLUSION

In this paper we have described negative effects that could occur when multiple DASH clients are competing for a bottleneck. Furthermore, we have evaluated our assumptions on these negative effects that are related to proxies with Microsoft Smooth Streaming and the MPEG-DASH implementation from [6]. Both evaluations have shown that our assumed negative effects, i.e., frequent quality switching and potentially jerky playback with a high number of unsmooth seconds are related to the proxy server and the false interpretation of the available throughput. Moreover, we have specified and evaluated our own adaptation logic which eliminates these negative effects and decreases the utilized bottleneck bandwidth with an enhanced cache reuse at the proxy server. Our future research comprises the evaluation of the other major industry solutions as well as the evaluation of these systems and our own fair adaptation logic under dynamic bandwidth conditions with more than two clients and competing non-DASH traffic. We assume that the negative effects will be increased in such a scenario.

8. ACKNOWLEDGMENTS

This work was supported in part by the EC in the context of the ALICANTE (FP7-ICT-248652), SocialSensor (FP7-ICT-287975) projects and partly performed in the Lakeside Labs research cluster at AAU.

8. REFERENCES

[1] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP – Standards and Design Principles", *ACM Multimedia Systems*, San Jose, CA, USA, Feb. 2011.

[2] A. Zambelli, "IIS Smooth Streaming Technical Overview," *Technical Report*, Microsoft Corporation, March 2009.

[3] R. Pantos et al., "HTTP Live Streaming," *Draft Pantos HTTP Live Streaming 07*, Internet Engineering Task Force, Sep. 30, 2011.

[4] Adobe HTTP Dynamic Streaming, http://www.adobe.com/products/httpdynamicstreaming/ (last access: Apr., 2012).

[5] T. Hossfeld et al., "Quantification of YouTube QoE via Crowdsourcing", *In Proceedings of IEEE International Symposium on Multimedia (ISM) 2011*, pp.494-499, 2011.

[6] C. Müller, C. Timmerer, "A VLC Media Player Plugin enabling Dynamic Adaptive Streaming over HTTP," *In Proceedings of the ACM Multimedia 2011*, Scottsdale, Arizona, Nov. 2011.

[7] K. Evensen, D. Kaspar, C. Griwodz, P. Halvorsen, A. F. Hasen, P. Engelstad, "Improving the Performance of Quality-Adaptive Video Streaming over Multiple Heterogeneous Access Networks", *ACM Multimedia Systems 2011*, San Jose, CA, USA, Feb. 2011.

[8] C. Liu, I. Bouazizi, M. Gabbouj, "Rate Adaptation for Adaptive HTTP Streaming", *ACM Multimedia Systems 2011*, San Jose, CA, USA, Feb. 2011.

[9] R. Kuschnig, I. Kofler, H. Hellwagner, "An Evaluation of TCP-based Rate-control Algorithms for Adaptive Internet Streaming of H.264/SVC", *ACM SIGMM Conference on Multimedia Systems (MMSys 2010)*, ACM, New York, NY, USA, 2010.

[10] R. Houdaille, S. Gouache, "Shaping HTTP adaptive streams for a better user experience", *ACM Multimedia Systems 2012*, Chapel Hill, North Carolina, USA, Feb. 2012.

[11] R. K. P. Mok, X. Luo, E. W. W. Chan, R. K. C. Chang, "QDASH: A QoE-aware DASH system", *ACM Multimedia Systems 2012*, Chapel Hill, North Carolina, USA, Feb. 2012.

[12] 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.

[13] B. J. Kwak, N. O. Song, L. E. Miller, "Performance Analysis of Exponential Backoff", IEEE Transactions on Networks, vol. 13, no. 2, Apr. 2005.

[14] Big Buck Bunny Movie, http://www.bigbuckbunny.org (last access: Apr. 2012).

[15] x264, http://www.videolan.org/developers/x264.html, (last access: Apr. 2012).

[16] C. Müller, S. Lederer, C. Timmerer, "An Evaluation of Dynamic Adaptive Streaming over HTTP in Vehicular Environments," *In Proceedings of the 4th Workshop on Mobile Video (MoVid12)*, Feb. 2012.

[17] Squid, http://www.squid-cache.org/, (last access: Apr. 2012)