Towards Peer-Assisted Dynamic Adaptive Streaming over HTTP

Stefan Lederer, Christopher Müller, 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— **This paper presents our peer-assisted Dynamic Adaptive Streaming over HTTP (pDASH) proposal as well as an evaluation based on our DASH simulation environment in comparison to conventional approaches, i.e., non-peer-assisted DASH. Our approach maintains the standard conformance to MPEG-DASH enabling an easy and straightforward way of enhancing a streaming system with peer assistance to reduce the bandwidth and infrastructure requirements of the content/service provider. In anticipation of the results our system achieves a bandwidth reduction of Content Distribution Networks (CDN) and as a consequence the corresponding infrastructure costs of the content/service providers by up to 25% by leveraging the upstream capacity of neighboring peers. Furthermore, the cost savings have been evaluated using a cost model that is based on the current Amazon CloudFront pricing scheme. Furthermore, we have also evaluated the performance impact that various combinations of quality levels of the content could have in a peer-assisted streaming system as well as the client behavior in such an environment.**

Keywords- Peer-Assisted Streaming, MPEG-DASH, Dynamic Adaptive Streaming over HTTP, CDN Bandwidth Reduction, Peer-to-Peer Streaming.

I. INTRODUCTION

Dynamic Adaptive Streaming over HTTP (DASH) [1] comprises standardization efforts within ISO/IEC MPEG and 3GPP. In particular, the MPEG-DASH standard has been ratified recently [2]. Additionally, also the industry has previously deployed various solutions in this area, i.e., Microsoft Smooth Streaming [3], Adobe Dynamic HTTP Streaming [4] and Apple HTTP Live Streaming [5].

However, all of these systems follow nearly the same architecture which means that they adopt a chunk-based HTTP streaming approach. This approach is simple but effective as the media will be encoded at different bitrates, resolutions, etc. and will then be chopped into segments that could be individually accessed by the client via HTTP requests. A version of the media content with a specific characteristic (e.g., bitrate, resolution) is referred to as *representation*. These representations which consist of segments, i.e., the chopped media content, will be transferred on top of the current Internet infrastructure following a client/server-based paradigm. In comparison to traditional HTTP streaming (i.e., progressive download of a single file) this approach dynamically adapts the stream to the users' bandwidth requirements or capabilities also during the streaming session. Due to the fact the logic of such systems is located at the client it also scales very well as infrastructure wise it is possible to leverage existing HTTP-based content delivery networks (CDN) and proxy cache infrastructures. The costs of those are significant lower than dedicated streaming servers like the Flash Media Server or other competing products and may be a good reason for providers to switch to adaptive streaming over HTTP.

In practice, however, the current traffic of those services comprises approximately 50% of the U.S. peak traffic and the future traffic of such systems shown in [1] that will be significantly higher than the current one. Hence, content and network providers will appreciate improvements that could reduce the server load and downstream traffic.

Interestingly, none of these commercial HTTP streaming deployments leverage the upload bandwidth of the clients consuming their streams. In our approach we will exploit this fact, i.e., all clients act like a peer in a peer-to-peer (P2P) system which means that they could download media content from other clients that actually consume the same stream or from the server. Thus, the P2P traffic could be seen as assistance for the conventional client/server-based download scheme. As the concept of peer-assisted streaming has been presented in the past (cf. Section II), the actual novelty of this paper is the integration of peer-assisted streaming into MPEG-DASH and evaluation thereof without compromising the standard. This approach fits very well, because in DASH the content has been already separated into segments to support adaptive bitstream switching during the session. Additionally, the media presentation description (MPD) describing all these segments is used as basis for additional information about neighboring peers. In doing so the standard conformance of DASH is maintained as well as all benefits of the system like the possibility to leverage HTTP-based infrastructures. In the following we adopt the term pDASH when referred to our peer-assisted DASH approach.

The remainder of this paper is organized as follows. Section II describes related work and the details of pDASH are presented in Section III. The simulation environment is described in Section IV followed by the results of our experiments in Section V. Finally, Section VI provides conclusions and points out some future work.

II. RELATED WORK

The online music streaming service Spotify is a well proven example for a peer-assisted streaming system with practical importance. As shown in [8] and [9] only 8% of the played music is streamed from Spotify servers, the rest is coming from

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

local buffers and other peers, a fact which has a significant impact on the companys' infrastructure costs. This cost reduction is also one main reason why it is possibly for Spotify to offer their service at relatively low fees. In comparison to other providers with the additional use of advertisements in their streams, they are even able to offer a free service to their customers. Similar to our pDASH system, Spotify does not maintain a P2P overlay network for, e.g., search, because in their opinion, this would introduce too much overhead. Considering the high churn rate of this audio streaming use case, the real-time requirements and the low upload resources of the clients, it may be a good compromise for keeping this system as simple as possible by the usage of a centralized approach with supporting P2P traffic rather than a decentralized P2P overlay [10].

Another commercial peer-assisted streaming framework is Adobe Cirrus [11] which is implemented in Adobe Flash Player since version 10.1. However, it is disabled by default and needs an explicit user action to enable it. Although Cirrus is mostly applied in combination with Adobe HTTP Dynamic Streaming it is based on Adobes' Real Time Media Flow Protocol (RTMFP) which uses UDP as transport layer and therefore introduces the know UDP specific problems like unreliable transfer and NAT traversal issues. In contrast to Spotify it uses a distributed hash table for searching the needed segments.

Furthermore, also in research there is much literature about peer-assisted video streaming but here we want to especially emphasize the work of [6] and [7] which are investigating the performance and boundaries for live and on demand streaming use cases. However, these papers focus on a more general investigation of peer-assisted streaming whereas our approach focuses on the integration of peer-assisted streaming into DASH and the evaluation thereof.

III. pDASH: Peer-assisted DASH

DASH content comprises fragmented segments described in a corresponding media presentation description (MPD) to enable adaptive HTTP streaming. However, this is also the case in several peer-to-peer based file sharing and video streaming approaches. The challenge within P2P-based video streaming is that the majority of users have an asymmetric Internet connection with a significant lower upload than download bandwidth, e.g., 8-16 Mbit/s download vs. 1-2 Mbit/s upload bandwidth. That is, while consuming a HD video stream one is not able to share the same amount of data received to other peers and, thus, it is not possible to guarantee a smooth playback at best quality for all peers.

As a good compromise between conventional client-server and distributed P2P systems, peer-assisted streaming seems to be a promising candidate. In those systems the client-server download is supported by downloading files or parts thereof from other peers which already have downloaded the content in the past. First commercial streaming services like the audio streaming service Spotify already use this architecture successfully, maintaining significant reductions in infrastructure and bandwidth needs as well as still providing the same quality of service (QoS) [8].

```
<MPD>
 <BaseURL>
    http://www.cdn.com/tracker.php?file=
 </BaseURL>

 <Period>
   <AdaptationSet bitstreamSwitching="true">

    <Representation bandwidth="2000000"....>
     <BaseURL>http://client1-IP/example</BaseURL>
     <BaseURL>http://client2-IP/example</BaseURL>
     <SegmentList duration="4">
      <SegmentURL media="segment1.mp4">
     </SegmentList>
    </Representation>

    <Representation bandwidth="4000000"....
     <BaseURL>http://client1-IP/example</BaseURL>
     <!-- further base urls and Segments -->
    </Representation>

   </AdaptationSet>
 </Period>

 <Period>
   <AdaptationSet bitstreamSwitching="true">
    <Representation bandwidth="2000000"....
     <BaseURL>http://client2-IP/example</BaseURL>
     <SegmentList duration="4">
      <SegmentURL media="segment2.mp4">
     </SegmentList>
    </Representation>
     <!-- further representations -->
   </AdaptationSet>
  </Period>
  <!-- further periods -->
</MPD>
```

Listing 1: Simplified peer-assisted DASH MPD.

In our approach of pDASH we want to show an easy and straightforward way of using DASH as basis for a peer-assisted streaming system for high quality Video on Demand (VoD) services. Hence, an important requirement is to allow only DASH-compliant communication mechanisms between the client and the server as well as among the other clients, i.e., peers. Therefore, the MPD was modified to include the required information about peers having already downloaded some other parts of the currently consumed content. This can be achieved by using one period per segment and adding additional BaseURLs to the corresponding representations. Please note that the resulting MPD (comprising one-segment periods with multiple BaseURLs per representation) is fully compatible with the standard [2]. A simplified example of such a fully DASH-compatible MPD enabling peer-assisted streaming is shown in Listing 1. The client has the choice of downloading a segment directly from the CDN, with the URL set in the BaseURL on MPD level, or it could also choose the option to download the segments or parts thereof from another client by using the BaseURLs that have been provided on representation level.

A. Peer-Assisted DASH Client

The functionality on the client to enable pDASH is as follows. Due to the fact that we use the MPD as reference to indicate the location of segments that have already been downloaded by other peers, we do not have to maintain any overlay network or distributed hash table [10] for searching segments. The consequence of this simple but effective design is that we only need three modifications described in the following.

First, every DASH client that participates as a peer needs a local HTTP Web server which handles the HTTP requests from other peers. Considering the limited upload capacity of Internet connections it makes sense to split up those requests for a segment in several smaller requests for different parts of the segment and send those requests to different peers. In doing so the traffic is distributed more evenly over several peers as well as the possibility of getting at least some parts of the segment from other peers is increased. In our evaluation we have split up the file into eight parts that will be consistently called chunks in this paper. We choose eight chunks because typically the ratio between down- and upload bandwidth of home Internet connections is eight to one. On the other hand, clients have to limit the number of incoming connections in order to guarantee a minimum upload bandwidth to individual requesting peers (e.g., four incoming connections have proven as a good empirical heuristic, however this needs further evaluations in future work to find the optimum). If the amount of concurrent incoming connections reaches the maximum, all other incoming requests will be discarded until one of the currently active transfers is finished.

The second modification concerns the download logic as well as the stream switching algorithm to handle the simultaneous download of chunks from different peers. In doing so we decided to use a rather simple adaption logic to reduce the system complexity and side effects to the peer connections. After session initiation, and also in case of low buffer, the client chooses the representation with a bitrate of 50% of the clients' available bandwidth to reach a minimum buffer level, i.e., a stable buffer state. Subsequently, the representations are selected based on the maximum available bandwidth. This adaption process is described as follows:

$$maxbw(s_i) = \begin{cases} bw(s_{i-1}) * 0.5 & if \quad 0.0 \le bl_i < 0.3 \\ bw(s_{i-1}) & if \quad 0.3 \le bl_i \end{cases}$$

$i \in [1, N]$ ... *Segment index*
bl_i ... *Buffer level at decision time of segment i*
$bw(s_i)$ ... *The function returns the bandwitdth which was measured during the download of segment i*
$maxbw(s_i)$ ... *Maximum bitrate of segment i*

Additionally, the ability of the client to download chunks from other peers is restricted by the current buffer level and not enabled by default. The download from other peers is possible if the buffer fill level is higher than 50% and will be disabled if it drops below 30%. This is necessary to prevent potential stalls in playback and guarantee a smooth playback. The logic will be described in the following.

$$peerAssist = \begin{cases} true & if \quad bl_i > 0.5 \\ false & if \quad bl_i < 0.3 \end{cases}$$

The peer-assisted download algorithm works by always handling two segments in parallel, one is defined as peer-segment and the other one is defined as server-segment. At the beginning, the algorithm tries to download the peer-segment from other peers described in the MPD. For this purpose, it is split up into a given number of chunks, eight in the evaluations' case as described previously. Each of those chunks is requested randomly from one of the other peers using HTTP byte range requests [12]. Note that the requests to other peers are made in a rather optimistic way, i.e., they will be accepted only if the other peer has not yet reached its limit for incoming connections. Otherwise, the other peer discards it and the download logic will mark the associated chunks of this request as unsuccessful. Furthermore, the accepted requests for chunks have a time limit of maximum one segment length to be finished, otherwise they will be aborted and marked as unsuccessful also. If requests are successful, the associated chunks do not need to be downloaded from the server anymore, and, thus, they will be marked as downloaded. After reaching the time limit, all missing chunk requests are combined in the server-segment and requested from the Web server by the usage of a combined HTTP byte range request, which guarantees a smooth playback.

Finally, the third aspect for the peer-assisted DASH client is the size of the local cache (buffer) that influences directly the amount of different segments which the peer is able to serve. Obviously the buffer size is highly related to the P2P traffic, i.e., a bigger buffer will increase the P2P traffic and therefore it will reduce the server load and infrastructure costs. In our proposal the cache compromises all downloaded segments of the current video stream. Even bigger caches are maintained by commercial systems like Spotify, which uses up to 10 Gbytes [8] at each client. In the case of our video streaming use case this would be at least enough for one or two HD movies. The cache becomes especially interesting when one considers VoD systems based on set-top boxes. Those boxes are used for selected on-demand content but they are often just an additional media source and, therefore, they are idle if the user watches conventional broadcast television services. Due to those idle times, such devices are perfect for serving their cached content to other peers and, as a consequence, supporting the peer-assisted streaming system with additional upload capacity.

B. HTTP Web Server and Segment-tracker

For pDASH we implemented a central segment tracker in addition to the HTTP Web server used for providing the segments. In our implementation we used simple PHP scripts which can also run on such HTTP Web servers. The aim of the tracker is to monitor the segment requests of each client together with its Internet Protocol (IP) address and a timestamp, which can be stored in a file or a database (MySQL in our implementation). The integration of such a segment-tracker in DASH is rather simple because it can be placed in the BaseURL on MPD level shown in Listing 1 by the URL *http://www.cdn.com/tracker.php?file=*. In doing so each segment request to the Web server is processed via this tracker which logs the request and responds with the requested segment.

In addition to the tracker, the MPD generator – also implemented using PHP – uses the segment requests monitored by the segment tracker for upcoming MPD requests and integrates all clients having segments, which are part of the requested content, to the generated MPD. In doing so the resulting MPD is enhanced by additional BaseURLs on the representation level, which represent other peers having those

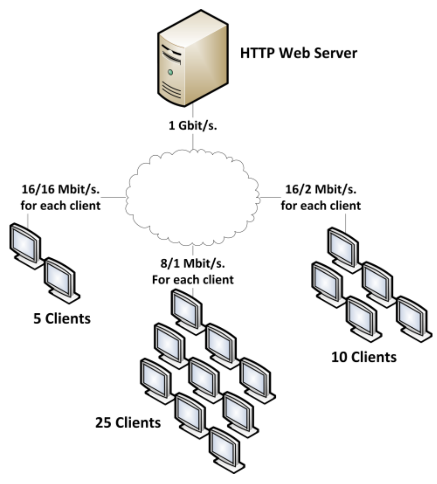

Figure 1: Simulation network with 35 clients.

segments in their cache as shown in Listing 1. For the integration of a peer in the MPD it is important to maintain a certain time window describing the maximum time allowed to be passed since the segment was downloaded. This helps the MPD generator to incorporate peers only that are still online as well as to keep the MPD size low. For example, it does not make sense to include a peer into the MPD which has downloaded a segment several days ago. Another important aspect to consider is the size of the generated MPD which can get relatively big when integrating many BaseURLs of peers having relevant segments. Therefore, we make use of ZIP compression of the response message and reduces the MPDs to 5% and less of their original size. Note that this is supported by all major HTTP servers and also compliant to MPEG-DASH.

When considering VoD platforms serving a huge number of concurrent users it becomes apparent that it does not make sense to integrate all possible clients within the MPD. Therefore, we suggest making use of clusters of clients with approximately the same bandwidth characteristics where only the clients within those groups are considered in the MPD generation. Based on the segment request log produced by the segment tracker this clustering can be done, e.g. by building clusters of clients downloading the same representation of a movie. This reduces the maximum size of the MPD and limits the known peers of a client to a manageable amount allowing for the use of persistent connections as defined in HTTP 1.1 [12] or bandwidth statistics for already known peers in further steps of improving the system.

IV. SIMULATION ENVIRONMENT

For the evaluation of our system we decided to build a simulation based on OMNeT++ 4.2 [13] in order to test our pDASH with an arbitrary number of clients having different characteristics. In particular, we used the OMNeT++-based INET framework [14] as basis for our implementation which provides the full TCP/IP stack and network components needed for a realistic simulation. Based on this we implemented a DASH client as well as a HTTP Web server with segment tracker functionality using parts of the HTTPTools framework [15] and parts of our DASH VLC plugin presented in [16].

Table 1: Representations of the simulation content.

Bitrate	Resolution
101 kbit/s.	320x240
201 kbit/s.	480x360
395 kbit/s.	480x360
700 kbit/s.	854x480
1172 kbit/s.	853x480
1992 kbit/s.	1280x720
2995 kbit/s.	1920x1080
3992 kbit/s.	1920x1080
4979 kbit/s.	1920x1080
5936 kbit/s.	1920x1080

Our test network shown in Figure 1 consists of the central HTTP Web server including the segment tracker functionality, a network representing the Internet and 35 clients. The server is connected to the network via a 1 Gbit/s connection while the clients are connected with different down and upload bandwidths. For our simulation we assumed 50% are using a DSL connection with 8 Mbit/s download and 1 Mbit/s upload bandwidth, 30% are using a faster connection with 16 Mbit/s download/2 Mbit/s upload bandwidth, and 20% use a symmetric connection with 16 Mbit/s. The simulation starts after a 15 second delay which is used to prevent the simulation from any startup configuration influences with the first client starting to request the MPD from the server. The clients are starting their request as follows:

- 8/1 Mbit/s clients: Starting at second 15 followed by the next one at second 20, adding another client every 10 seconds until the last one at second 290.
- 16/2 Mbit/s clients: Starting at second 22, adding another client every 20 seconds until the last one at second 202.
- 16/16 Mbit/s clients: Starting at second 17, adding other clients every 30 seconds until the last one at second 137.

Based on this client arrival scheme we evaluated the first 5 minutes of the simulation which shows best the characteristics of the peer-assisted streaming.

For our simulation we used the Red Bull Playstreets sequence form our DASH dataset in [17] and reduced the amount of representations to nine as shown in Table 1. However, only representations greater than 1 Mbit/s will be important for the simulation because the lower ones are not used any more after the initial bandwidth measurement, which is done with the first segment of the lowest bitrate representation. We used the version with a segment length of four seconds, which has several reasons. Our OMNeT++-based implementation uses non-persistent connections which need longer segments lengths as shown in our previous evaluation [17]. In contrast to this longer segment lengths led to bigger peer request for chunks. Such bigger requests led to longer utilization periods of the peers' upload bandwidth and this may influence the download throughput of those peers. Hence, we decided to use the four second segment length version as a good compromise.

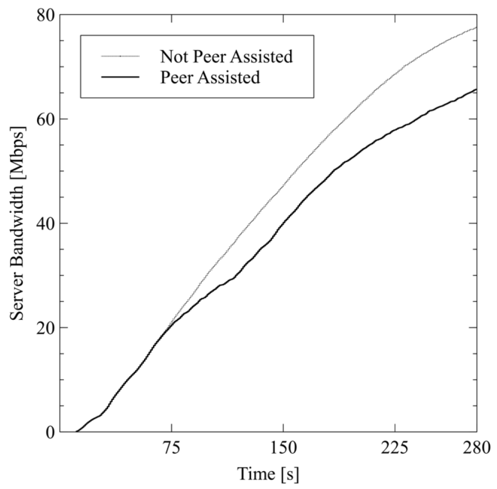

Figure 2: Simulation with 6 Mbit/s representations limit.

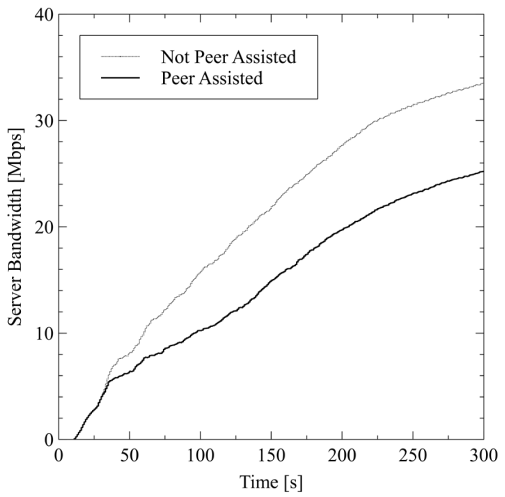

Figure 3: Simulation with 1400 Kbit/s representation limit.

V. Evaluation Results

We performed two types of simulations which differ by the quality limits in terms of bitrate of the used representations. We analyzed the utilized server bandwidth with and without activated peer assistance to show the bandwidth reduction on the server side as shown in Figure 2 for simulation one and Figure 3 for simulation two, which will be described in the following. Additionally, we analyzed the client behavior by investigating the amount of data transferred from other peers during the simulation, which is shown for a representative client of simulation two depicted in Figure 4. These simulations as well as the used representations and their results will be described in detail in the following.

In our first simulation we used all representations shown in Table 1. Due to this, the clients with 16 Mbit/s downlink are able to use representations with higher bitrates than those clients using the 8 Mbit/s downlink. Therefore, the 16 Mbit/s clients download different segments than the 8 Mbit/s ones and, thus, are not useful for each other. As shown in Figure 2 the usage of peer assistance reduces the server bandwidth by about 15% at the end of this simulation scenario. As one can see, the peer-assisted streaming starts to reduce the server bandwidth effectively after approximately 70 seconds. This is the case because earlier in the simulation there are not enough clients with the demanded segments available to leverage their upload capacity. The benefit of peer-assisted DASH increases over the time up to a bandwidth saving of 11 Mbit/s at the end of the simulation.

As already mentioned the reduction of bandwidth and especially peak bandwidth which is needed in times of high system usage as well as the amount of transferred data in general influences the infrastructure costs of a VoD service. To provide an example for such savings we evaluated them based on the saved total traffic of our simulation use cases on a per year basis using the pricing model of Amazon CloudFront [18]. Of course, this is only one part of the costs of such a system and does not cover the costs for guaranteed bandwidth capabilities or storage, maintenance, etc., but those are mainly service dependent and their costs are rather more on an individual basis than on a public pricing scheme. Although, we think that the evaluated model provides a good example how the benefits from our peer-assisted streaming system can significantly influence relevant business areas.

Based on the pricing scheme of Amazon CloudFront the bandwidth savings achieved with pDASH would correspond to a cost reduction of 15%. For our simulation scenario producing costs of US$ 4.14 per hour this would lead to cost reductions of US$ 0.62 per hour. This looks a little bit less but if we take this example further under the assumption of a weekly average of 35 hours streaming at our final bandwidth of about 78 Mbit/s this would reduce the total costs per year by US$ 1,135 based on total costs of US$ 7,530. This has already a high impact on the cost side. However, other reductions for reduced peek bandwidth capabilities on the CDN side etc. are not even included.

Several VoD providers do not offer high bitrates for their content, especially those who do not charge their users like, e.g., Hulu [19]. This is the reason why we made simulations where we limit the maximum bitrate of our content. So we recuded the maximum bitrate of the content in our second simulation to 1400 kbit/s, which is a downloadtable bitrate even for the peers with the slowest Internet connection in our evaluation network. At the end of the simulation, this results in a server-side bandwidth recuction of about 25 % compared to the same scenario without peer assistance, which is an improvement to the previous simulation. This significant increase in the percentage of the bandwidth recution results from to the lower content bitrate which is easier to transfer between the peers because of the lower bandwidth requirements for it. Furthermore all clients are limited to the same maximum bitrate representation, which means that all clients have segments of the same maximum representation in their cache, which results in a better distribution of the segments among the peers.

Based our simulations we also investigated a representative client with 8 Mbit/s download bandwidth showing the download traffic of the server and peer-to-peer traffic. The client is taken from our second simulation with a representation bitrate limit of maximum 1400 kbit/s. The client shown in Figure 4 starts at second 214 of the simulation and shows the expected behaviour which is downloading as much as it get from other peers. This peer traffic changes a lot over time

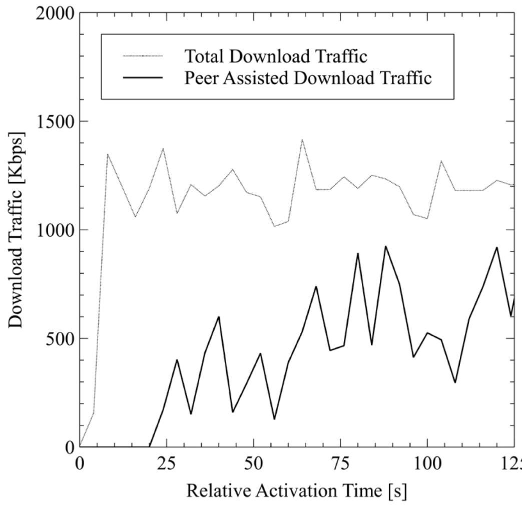

Figure 4: Download traffic composition of a client.

depending on the selection of the neighbor peers which is at the moment based on a random choice out of the number of peers offering the desired segment. As a fallback the client can always rely on the high bandwidth of the origin server to maintian a smooth playback. At the beginning the client starts to fill its buffer by downloading lower representation from the Web server. After a few seconds the buffer level is high enough to start to download also from other peers (cf. around second 20 in Figure 4). Large parts of the segments are downloaded from other peers during the rest of the session. After second 65 the P2P traffic contributes for some segments even more than 50% to the total download traffic of the client and, thus, reduces the amout of data transferred from the sever noticeably.

VI. Conclusion and Future Work

In this paper we have presented an integrating of peer-assisted streaming into the recently ratified MPEG-DASH standard without compromising its merits such as the exploitation of existing infrastructures. Our simulation results demonstrated that we are able to reduce the server bandwidth up to 25% thanks to the peer assistance. Additionally, we have shown that these bandwidth reductions could be directly converted to infrastructure costs which gives peer assistance in video streaming a significant business impact.

Future work may include more sophisticated peer selection algorithms to achieve a better load distribution between the available upload capacities of the different peers, e.g. as in [20]. Furthermore, a theoretical analysis of the system performance as well as its boundaries will be part of future work in this area. Finally, the regular update of the MPD during streaming may lead to a better utilization of other peers. Furthermore, the download logic could be extended to handle more chunks in parallel than the two which are used in our implementation. This may give the peer requests more time for transmitting the subsegments and therefore reduce the overhead caused by aborted transmission due to delayed transfer.

References

[1] T. Stockhammer, “Dynamic Adaptive Streaming over HTTP – Design Principles and Standards”, In Proceedings of the second annual ACM conference on Multimedia systems (MMSys11), ACM, New York, NY, USA, 2011

[2] ISO/IEC DIS 23009-1.2, “Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats”

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

[4] Adobe HTTP Dynamic Streaming, http://help.adobe.com/en_US/HTTPStreaming/1.0/Using/index.html , (last access: Dec. 2011)

[5] R. Pantos, W. May, “HTTP Live Streaming, IETF draft” (Jun. 2010) http://tools.ietf.org/html/draft-pantos-http-live-streaming-04 (last access: Dec. 2011).

[6] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, M. Chiang, “Performance bounds for peer-assisted live streaming”, In the Proceedings of the 2008 ACM SIGMETRICSN, New York, USA, 2008

[7] S. Lin, J. Wu, K. Xu, Z. Ma, “The Minimum Server Bandwidth in Peer-Assisted VoD Systems”, In the Proceedings of 19th International Conference on Computer Communications and Networks (ICCCN), Zürich, CH, 2010

[8] G. Kreitz, F. Niemelä, “Spotify – Large Scale, Low Latency, P2P Music-on-Demand Streaming”, IEEE International Conference on Peer-to-Peer Computing (P2P) 2010 , Delft, Netherlands

[9] M. Goldmann, F. Niemelä “Measurements on the Spotify Peer-Assisted Music-on-Demand Streaming System”, IEEE International Conference on Peer-to-Peer Computing (P2P) 2011 , Kyoto, Japan.

[10] L. Keong Eng, J. Crowcroft, M. Pias, R. Sharma, S. Lim, “A survey and comparison of peer-to-peer overlay network schemes”, *in IEEE Communications Surveys & Tutorials*, Vol.7, No.2, pp. 72- 93, Second Quarter 2005.

[11] Adobe Cirrus, http://labs.adobe.com/technologies/cirrus/ (last access: Dec. 2011)

[12] Fielding, R. et al, RFC 2068 - Hypertext Transfer Protocol -- HTTP/1.1, http://www.w3.org/Protocols/rfc2616/rfc2616.html (last access: Dec. 2011).

[13] OMNeT++, http://www.omnetpp.org/, (last access: Dec. 2011)

[14] INET Framework, http://inet.omnetpp.org, (last access: Dec. 2011)

[15] K. Jonsson, “HttpTools: A Toolkit for Simulation of Web Hosts in OMNeT++”, In proceedings of the 2nd OMNeT++ workshop, Rome, Italy, 2009.

[16] C. Müller, C. Timmerer, “A VLC Media Player Plugin enabling Dynamic Adaptive Streaming over HTTP”, In Proceedings of the ACM Multimedia 2011, Scottsdale, Arizona, November 28, 2011.

[17] S. Lederer, C. Müller, C. Timmerer, “Dynamic Adaptive Streaming over HTTP Dataset”, in Proceedings of ACM Multimedia Systems Conference 2012, Chapel Hill, North Carolina, February 22-24, 2012.

[18] Amazon CloudFront Pricing, http://aws.amazon.com/de/cloudfront/pricing/, (last access: Dec. 2011)

[19] Hulu, Technical FAQ, http://www.hulu.com/support/technical_faq, (last access: Dec. 2011).

[20] M. Zhang, Y. Xiong, Q. Zhang, L. Sun, S. Yang, “Optimizing the Throughput of Data-Driven Peer-to-Peer Streaming” , IEEE Transactions on Parallel and Distributed Systems (TPDS), vol. 20, no. 1, pp. 97-110, Jan. 2009.