Implementation and User-centric Comparison of a Novel Adaptation Logic for DASH with SVC

Christian Sieber, Tobias Hoßfeld, Thomas Zinner, Phuoc Tran-Gia
University of Würzburg, Institute of Computer Science
Am Hubland, 97074 Würzburg, Germany
Mail: prename.name@uni-wuerzburg.de

Christian Timmerer
Alpen-Adria-Universität Klagenfurt
Universitätsstraße 65–67, 9020 Klagenfurt, Austria
Mail: christian.timmerer@itec.uni-klu.ac.at

***Abstract*—The MPEG-DASH standard allows the client-centric access to different representations of video content via the HTTP protocol. The client can flexibly switch between different qualities, i.e., different bit rates and thus avoid waiting times during the video playback due to empty playback buffers. However, quality switches and the playback of lower qualities is perceived by the user which may reduce the Quality of Experience (QoE). Therefore, novel algorithms are required which manage the streaming behavior with respect to the user's requirements and which do not waste network resources. As indicated by recent studies, scalable video coding (SVC) may use the current network and content distribution infrastructure in a more efficient way than with single layer codecs. The contribution of this paper is the design and the implementation of a novel DASH/SVC streaming algorithm. By means of measurements in a test-bed, its performance and benefits are evaluated and compared to existing algorithms from an user-centric view point with objective performance metrics. Our findings show that the proposed algorithm outperforms other DASH mechanisms in terms of video quality, low switching frequency and usage of the available resources in a realistic mobile network scenario. This is a first step towards true QoE management of video streaming in the Internet with DASH and SVC.**

I. Introduction

Recent studies [2] on the current traffic composition of the Internet show the dominating role of video traffic with a traffic share of 51 %. This includes video conferencing, IPTV, as well as VoD systems like Netflix and Youtube. Video conferencing and IPTV have severe real-time constraints and thus mostly rely on RTP/UDP for the video transmission. This is in contrast to VoD solutions which take advantage of the pre-encoded content and buffers at the end users device and in general utilize HTTP/TCP for progressive video streaming.

The video streams are delivered via heterogeneous wireless and wired access networks to the customers. Varying network resources and user mobility lead to a volatile service performance and have a significant impact on the user perceived service quality, the Quality of Experience (QoE). For RTP/UDP solutions such insufficient network resources generally lead to lost packets and thus to distorted video pictures. This differs from HTTP/TCP video streaming techniques where packet delivery is guaranteed. However, additional waiting times in packet delivery may lead to a depleting video playback buffer and stalling of the video playback. Studies indicate a high, negative impact of already low stalling times and frequencies [4] on the QoE. Accordingly stalling should be avoided.

One possibility to overcome this problem is addressed by the *Dynamic Adaptive Streaming over HTTP* (DASH) standard, which provides adaptive bit-rate HTTP-based streaming. It allows a client-centric access to different quality representations of the video content, i.e., different bit-rates, in a dynamic way and thus the flexible adaptation of the video quality to the current network conditions. Accordingly, one the one hand, the stalling occurrence may be reduced by switches to lower qualities, on the other hand lower qualities are played back and quality changes happen.

From the network perspective DASH enables the efficient and ease of use of existing content distribution and network infrastructure components such as CDNs, HTTP caches, NATs and firewalls. The protocol is typically used with single layer codecs like H.264/AVC, however, recent studies [15] showed that with scalable video codecs like H.264/SVC a more efficient usage of the infrastructure and a better played back video quality can be achieved. Further, a scalable video codec allows more download flexibility since already downloaded parts of the video clip can be enhanced at a later time.

Several download algorithms for DASH clients have been proposed recently, both, for single layer and multi-layer codecs. These algorithms either do not provide the best available quality or cause frequent quality changes. The *first contribution of this paper is a novel flexible algorithm for H.264/SVC DASH streaming* which adapts its downloading behavior to the type of network and optimizes the user perceived quality based on the QoE influence factors initial delay, stalling delays and frequencies, number of quality switches and played back video quality. The *second contribution is a user-centric comparison of existing adaptation mechanisms* with an implementation of the novel scheme by means of measurements in a test-bed for a mobile network scenario.

The remainder of this paper is structured as follows. Section II gives a short background on DASH and SVC and reviews related work on QoE. Section III describes the DASH adaptation algorithms and our proposed algorithm. Section IV provides the evaluation methodology, the setup of the test environment, and the evaluation metrics. The measurement results are analyzed and compared for the algorithms in Section V. Section VI summarizes the main findings and gives an outlook towards QoE management for DASH with SVC.

This work was partly funded by Deutsche Forschungsgemeinschaft (DFG) under grants HO 4770/1-1 and TR257/31-1, in the framework of the EU ICT Project SmartenIt (FP7-2012-ICT-317846), and the COST QUALINET Action IC1003. The authors alone are responsible for the content.

II. Background and Related Work

This section provides background information on the technology used in this paper, i.e., DASH and scalable video coding (SVC), and related work in this area focusing on QoE for HTTP video streaming.

The first proposals for multimedia streaming over HTTP/TCP were based on progressive download providing good performance when the TCP throughput is roughly twice the media bitrate [18]. More recent approaches follow a segment approach where the media segments are requested by individual HTTP requests enabling adaptive delivery with respect to the network conditions and device capabilities. Most industry solutions such as Adobe HTTP Dynamic Streaming, Apple HTTP Live Streaming (HLS), and Microsoft Smooth Streaming adopted this approach which finally led to the MPEG-DASH standard published by ISO/IEC in 2012 [6]. It specifies a Media Presentation Description (MPD) which describes one or more media representations where segments (based on ISO base media file format or MPEG-2 transport stream) are accessible using a HTTP uniform resource locator (URL). As DASH provides reliable delivery thanks to TCP, no packet loss occurs (as opposed to UDP-based streaming) but problems in the network or insufficient network resources may lead to an interrupted playback, referred to as stalling. Stalling effects have a significant impact on QoE for end users [3], [4].

Scalable Video Coding (SVC) is an amendment to Advanced Video Coding (AVC) and offers temporal, spatial, and fidelity scalability [17]. Temporal and spatial scalability allow for the adaptation of the frame rate and the content resolution respectively while fidelity scalability provides different levels of image quality. By applying a hierarchical coding schema SVC allows the selection of a suitable sub-bitstream for the on-the-fly adaptation of the media bitstream to device capabilities and current network conditions. A valid sub-bitstreams contains at least the AVC-compatible base layer and zero or more enhancement layers. Note that all enhancement layers depend on the base layer and/or on the previous enhancement layer(s) of the same scalability dimension.

The subjective evaluation of various SVC configurations is well-known [7], [14] but papers dealing with DASH and QoE focus on AVC and the integration of DASH and SVC is evaluated using objective metrics [9] and simulations [5], [16]. The adaptation mechanisms as deployed in DASH try to avoid stalling times and delays by switching to a more adequate media representation. Variations of the image quality due to the adaptation process can lead to flicker effects which can also have a negative impact on the QoE [11], [19].

ments and the MPD file standard HTTP server/client components are sufficient.

III. Investigated DASH Adaptation Algorithms

For the evaluation three algorithms from the literature were taken into account. Namely the algorithm proposed in [8], henceforth referred to as *TRDA*, the algorithm published in [10], henceforth referred to as *KLUDCP*, and the algorithm from [13], henceforth referred to as *Tribler*. *TRDA* and *KLUDCP* were designed for single-layer content (e.g. AVC) whereas *Tribler* was developed for layered content (e.g. SVC). We also propose a new SVC-based adaptation strategy, *Bandwidth Independent Efficient Buffering*, henceforth referred to as *BIEB*. SVC-based strategies allow for two-dimensional adaptation strategies where different representations of the same time slot can be requested independently. Single-layer strategies can request different representations of the same time slot, but only one can be used for decoding.

A. Existing Algorithms

TRDA takes three input parameters. The average bit-rate of each representation, the average throughput measured during the download of recent segments and the current buffer status in playback seconds. To minimize the initial delay, a fast-start phase was introduced in addition to the normal mode of operation. The algorithm reacts differently depending on the buffer level. For a buffer level less or equal to a configured minimum threshold the algorithm switches to the lowest representation. For a level less than a configured low threshold, the algorithm switches to the next lower representation if the measured bandwidth is less than the average bit-rate of the current representation. For a high buffer level, the algorithm delays subsequent requests if the measured bandwidth is not enough to switch to the next higher representation (compared to the average bit-rate of the next representation), otherwise it increases the quality by one. The current bandwidth is estimated by the throughput of recent segments. Additionally the estimation is modified depending on the current buffer level.
KLUDCP takes three input parameters to decide which representation to choose for segment i. The throughput measured during the download of segment $i-1$, the current buffer level as ratio of the configured maximum and the average bit-rate of each representation. An estimation of the current available bandwidth is then compared to the average bit-rates of each representation and the representation with the highest bit-rate less or equal the estimation is selected for segment i. The bandwidth estimation is a function of the current buffer level and the throughput measured for segment $i-1$. The estimation is decreased if the buffer level is less then 35% and increased if the buffer level is equal or higher then 50%.
Tribler does not take any input parameters and only two configuration parameters, t_1 and t_{max}. Starting from the current segment i it downloads only the lowest quality (i.e. base layer) up to $i+t_1$. Starting from $i+t_1$ the algorithm tries to download all representations of all segments between $i+t_1$ and $i+t_{max}$.

B. Proposed Algorithm

In contrast to the algorithms TRDA and KLUDCP, our proposed algorithm does not rely on estimations of the available bandwidth and does not assume a constant bit-rate of the content. In contrast to Tribler it uses a more dynamic segment-picking approach. The current revision of the algorithm does assume a constant size relation between the segments of each representation. A number of parameters are used during the playback. $r_{avg}(i)$ gives the average bit-rate of representation i *without* the dependency layers needed for encoding, $br(i)$ is defined as $br(i) = \frac{r_{avg}(i)}{r_{avg}(0)}$, i_{curr} is the currently selected representation, i_{min} and i_{max} are the lowest (i.e. the base layer) respectively the highest representation, p_{curr} returns the segment number of the current playback position, γ is the base number of segments which should be buffered per each

selected representation and $\delta(i)$ returns the number of segments currently buffered for representation i.

The desired buffer level for each selected representation is defined by

$$\beta(i) = \begin{cases} \gamma + br(i_{curr} - i) & \text{if } i \leq i_{max}\,, \\ \gamma + (i - i_{max} + 2) \cdot br(i_{max}) & \text{if } i > i_{max}\,. \end{cases}$$

It has to be noted that the definition may be further improved to reach a theoretical optimum. Figure 1 illustrates the algorithm by example where $p_{curr} = 15$, $\gamma = 5$ and each segment contains 2 seconds of playback time (e.g. $p_{curr} = 15$ translates to 30 seconds). The example shows a situation where the current quality layer is $i = 1$ and a growing phase is initiated to switch to $i = 2$. The algorithm is called after a segment

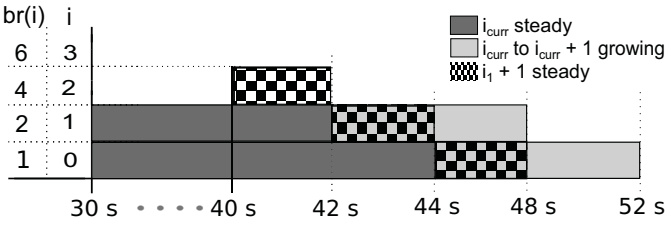

Fig. 1. Example for steady phase Q1, Q2 and growing phase from Q1 to Q2

$i = r_{min}$;
// Get i_{curr}
$i_{curr} = i_{max}$;
while $\delta(i_{curr}) == 0$ **do**
  $i_{curr} - -$;
end
// Steady Phase
while $i \leq i_{curr}$ **do**
  if $delta(i) < \beta(i)$ **then**
    request next segment of representation i;
    exit;
  end
  $i + +$;
end
// Growing Phase
while $i \leq i_{curr}$ **do**
  if $delta(i) < \beta(i + 2)$ **then**
    request next segment of representation i;
    exit;
  end
  $i + +$;
end
// Quality Increase
if $i \neq i_{max}$ **then**
  $i_{curr} = i_{curr} + 1$;
  request segment $p_{curr} + \gamma$ of representation i
else
  // idle until p_{curr} *increases*
end

Algorithm 1: BIEB Adaptation Algorithm

finished downloading. The algorithm loops from representation $i := i_{min}$ to $i \leq i_{curr}$ until it encounters a representation where the number of currently buffered segments $\delta(i)$ is less then the desired buffer level for this representation $\beta(i)$. If all representations including i_{curr} have reached their desired buffer level already, the algorithm increases the desired buffer level for each representation ($i_{curr} + 2$ instead of i_{curr}) as preparation for the switch to the next higher representation $i_{curr} + 1$. The pseudo code for the algorithm is given in Algorithm 1.

IV. Evaluation Methods and Measurement Setup

The evaluation of the adaptation algorithms from an objective point of view is done by a series of experiments in a test-bed environment at the Department of Communication Systems at the University of Wuerzburg. Realistic traffic patterns were used for the traffic shaping to mimic real-world use-cases and different DASH adaptation algorithms were implemented into a DASH evaluation framework. In this paper we propose a new algorithm which we have evaluated against three existing algorithms from the literature which are described in Section III. Figure 2 outlines the evaluation process.

The *realistic network scenarios* (i.e. the traffic patterns), the *DASH adaptation algorithms*, and the *test-content* are input parameters for the evaluation done in the *test-bed environment*. The DASH evaluation framework applies each algorithm in all scenarios using every content for a defined number of times and records statistics of the playback behavior during each run. From the collected statistics of the playback behavior significant objective metrics are identified (e.g. number of quality switches, initial delay and stalling times). Future publications will deal with the development of scenario-based models of the behavior of the adaptation algorithms. The developed models will be used for subjective laboratory- and crowd-sourcing-based evaluations to assess the impact of the identified objective metrics on the user perceived Quality of Experience. The results from subjective evaluations will support QoE-aware refinements of the adaptation algorithms.

A. Setup of the Test Environment

The test-bed consists of one or more clients connected to a sequence of two gateways (one for bandwidth shaping, one for delay shaping) and one HTTP content server running the apache web server. The traffic shaping is done using the Linux Advanced Routing & Traffic Control framework. The shaping gateways and the HTTP server were configured with Debian 6.0.6, the DASH client is running on a node set-up with Kubuntu 12.10. The shaping process is started simultaneously with the playback using the currently selected traffic pattern. The traffic pattern is run in a loop with random starting point.

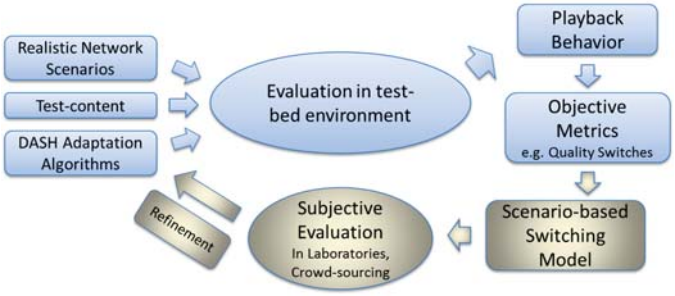

Fig. 2. Evaluation Process

TABLE I. TEST CONTENT RESOLUTIONS

Resolution	Average bit-rate	Maximal bit-rate
320x180	0.29 Mbps	1.28 Mbps
640x360	0.95 Mbps	3.37 Mbps
1280x720	2.67 Mbps	10.46 Mbps

The traffic patterns used for the evaluation were recorded in realistic wired and vehicular mobility situations, by measuring the throughput of a single large HTTP GET request on the network interface and simultaneously the delay of echo requests to google.de. The vehicular mobility patterns were recorded using a state-of-the-art UMTS/LTE stick while driving in and around Klagenfurt, Austria. The length of the network trace used is 11 min with an average bandwidth of 2.88 Mbps and a standard deviation of 1.44 Mbps. For the evaluation the delay component of the scenario was disregarded and only the bandwidth shaping was performed. The traffic pattern was played in a loop with random starting point for each run.

For the test content we used the open source live-action/computer generated-imaginary (CGI) short film Tears of Steel [1] which was encoded by the SVC JSVM reference encoder with three spatial resolutions as shown in table I. The encoded SVC bitstream was split into 367 segments with each segment having a playback length of two seconds (available for download [1]). TRDA was configured as suggested in [8], for Tribler t_1 was set to 10 segments (i.e. 20 seconds), t_{max} to 20 (i.e. 40 seconds), KLUDCP was configured as suggested by the authors and BIEB was configured with $\gamma = 8$ (segments).

B. Evaluation Metrics

We developed a cross-platform DASH client implementation in C++ based on the Boost libraries, libCURL, and pugixml. An easy integration of additional adaptation algorithms and the automatic gathering of statistics was the focus of our implementation. Currently, the implementation collects the following parameters during one run.

- Initial Delay: The delay between the first HTTP GET request and the start of the playback.
- Stalling Time: The sum of all playback interruptions.
- Number of quality switches: The total number of quality switches during the playback.
- Playback quality: The number of segments per representation and the average quality achieved.
- Inter switching times: The time between quality switches.
- Memory use: The average and maximum memory use of the algorithm during the playback.
- HTTP Utilization: The average HTTP throughput and the ratio compared to the theoretical maximum.

V. MEASUREMENT RESULTS

The goal of the measurements is the user-centric comparison of the four DASH adaptation algorithms in a demanding realistic scenario to stress their quality adaptation. The traffic traces from the vehicular mobility traffic scenario as described in Section IV were used to emulate varying network conditions of a single mobile user, while the adaptation mechanisms were reacting accordingly to the fluctuations of the available network bandwidth. Key performance measures were captured on network and on application layer, but also on system level of the video client. In particular, QoE influence factors (in terms of initial delay, stalling duration, playback quality, quality switching frequency) and efficiency (in terms of bandwidth utilization, wasted data, memory usage) were analyzed and compared. The measurements were performed in the test-bed at the University of Würzburg in Dec. 2012. For each of the four adaptation algorithms, the measurements were repeated 30 times in order to get statistically significant results. If not stated otherwise, the mean value and the maximum are derived over the 30 runs; the error-bars in the figures depict a 95 % confidence interval for the mean of the corresponding performance.

A. QoE Influence Factors and Playback Quality

According to [3], initial delays impact the QoE only to a small extent, while stalling events during the streaming session significantly degrades the QoE. The measurement results for the four adaptation algorithms show however that all algorithms succeed in preventing stalling of the video playout. Furthermore, the observed initial delay was below 2.5 s in all runs which has no impact on the QoE [3]. Thus, only the playback quality of the video determines the QoE in the considered mobile scenario.

To be more precise, the playback quality from a users' point of view is influenced by two factors, that are the image quality and the flicker effects [12], [19] caused by the adaptation process. For comparing the DASH algorithms, it is sufficient to assess the image quality by the corresponding number of segments per representation (the higher the representation, the better the image quality). The flicker effects are evaluated in terms of the frequency of quality adaptations defined as the ratio between the number of quality switches and the video duration and in terms of the time period with same quality level without any quality switches.

Figure 3 shows the cumulative distribution function (CDF) of continuous segments with the same quality level during the playback of one run. It can be seen that Tribler and KLUDCP are close together as well as TRDA and BIEB. While Tribler and KLUDCP show a high probability to have only a few consecutive segments at the same quality level, TRDA and BIEB result in a longer steady quality. With TRDA or BIEB the probability of a quality switch after 10 segments (i.e. 20 s) of playback with steady quality is with 35 % about three times less likely then with KLUDCP or Tribler (95 %). Figure 4 depicts the mean and maximum frequency of quality changes per minute. The playback using the TRDA or BIEB algorithm is less likely to have a high frequency of quality changes than using the KLUDCP or Tribler algorithm. The quality switching frequency is about 10 times higher for KLUDCP and Tribler compared to BIEB and TRDA. On average, KLUDCP and Tribler change the quality of the playback every 5.15 s and 6.81 s, respectively. BIEB and TRDA adapt every 61.33 s and 92 s, respectively, which are reasonable values to have only a low impact on the QoE. To put it in a nutshell, KLUDCP and Tribler are very aggressive algorithms and try to immediately adapt to the current network condition. This however leads to a high quality switching frequency. However, it has to be evaluated by means of subjective user studies how severe this impacts the QoE. Nevertheless, Proposed and TRDA are more

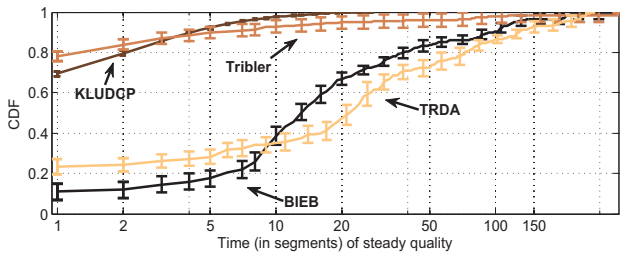

Fig. 3. Inter Switching Times CDF

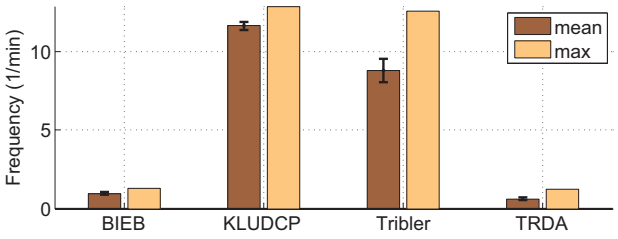

Fig. 4. Frequency of quality switches during playback

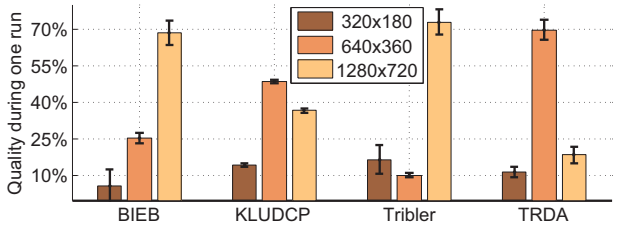

Fig. 5. Playback Quality in terms of segments per representation

conservative resulting in a low frequency and low impact on the QoE.

Figure 5 shows the playback quality in terms of segments per representation during a single run. BIEB and Tribler present 68% and 73% of the time the best image quality, respectively. KLUDCP and TRDA select the highest representation only 37% and 19% of the time, respectively. Hence, BIEB and Tribler clearly outperform KLUDCP and TRDA regarding the playback quality.

We conclude that the proposed algorithm shows the best performance from an user-centric point of view. Although Tribler also leads to a high playback quality, the QoE may suffer from the high number of quality switches which is part of our future work. KLUDCP performs worse considering playback quality and frequency. TRDA shows only medium playback quality but also leads to a low number of switches.

B. Efficiency and Usage of Resources

Especially in wireless scenarios the efficient usage of network resources is important. For mobile devices, the hardware resources may also be limiting. The efficiency and usage of resources are quantified on behalf of (a) the utilization of the available bandwidth by the DASH algorithm, (b) the number of segments which were downloaded but not used during playback, and (c) the memory usage on the video client during playback.

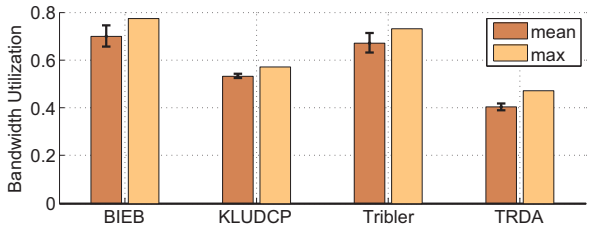

Fig. 6. Bandwidth utilization compared to the theoretical maximum

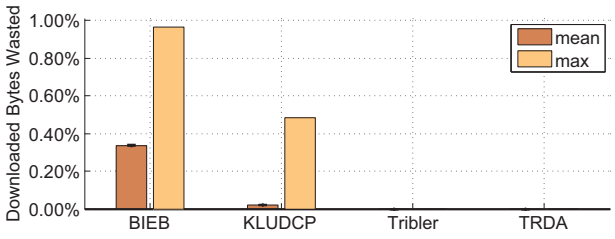

Fig. 7. Amount of wasted data downloaded but discarded during playback

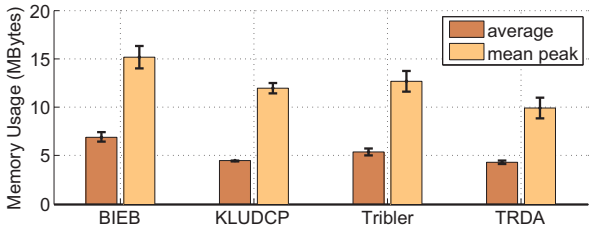

Fig. 8. Memory consumption during playback of video with size 235 MB

Figure 6 illustrates the average and maximum bandwidth utilization by the adaptation algorithms. BIEB and Tribler were able to use 70 % and 67 % of the available bandwidth, whereas KLUDCP and TRDA were using 53 % and 40 % of the bandwidth, respectively. However, the bandwidth utilization alone does not reflect the efficiency of the algorithm. There are situations where downloaded segments from enhancement layers (i.e., all representation excluding the first) can get discarded. This happens if the algorithm requests an enhancement layer segment which is too close to the current playback position. If the request takes too long (e.g., due to a drop in bandwidth) the playback position has already moved past the requested segment and it has to be discarded. Figure 7 shows the average amount of data each algorithm discarded during one run. In all 30 runs, TRDA and Tribler did not discard any segments during playback. KLUDCP discarded an insignificant amount per run. On average BIEB discarded 0.8 Mbytes per run. In 10 out of 30 runs the algorithm did not discard any segments, whereas in the 20 runs where it discarded segments, the average wasted bandwidth was 1.22 Mbytes. The problem of discarded segments is specific to using layered video codecs, but can be prevented through additional constrains for the segment selection. Future revisions of the proposed algorithm will include such constraints to increase the efficiency of the algorithm.

Figure 8 shows the memory usage at the client for buffering. We consider here the average consumption for a single run as well as the peak consumption for this run. We plot the mean of the average consumption (labeled as 'average') and the mean

TABLE II. COMPARISON OF THE ADAPTATION ALGORITHMS

	BIEB	KLUDCP	Tribler	TRDA
Using SVC or AVC	SVC	AVC	SVC	AVC
Playback quality (avg. quality, base=0)	high 1.63	med 1.22	high 1.56	med 1.07
Quality switching frequency (avg. switches per minute)	low 0.98	high 11.7	high 8.79	low 0.63
Bandwidth utilization (avg. bandwidth utilization)	high 70%	med. 53%	high 67%	low 40%
Wasted data (avg. wasted data)	med 0.33%	low 0.02%	low 0%	low 0%
Memory consumption (avg. memory consumption)	low 6.9	low 4.46	low 5.38	low 4.33

peak consumption (labeled as 'mean peak') over the 30 runs. None of the algorithms exceeds a peak consumption of 16 MB which corresponds to 6.6 % of the whole movie on the highest representation. This does not pose any problems for any kind of used client device. Similar results were obtained for CPU consumption which are omitted due to the page count limit.

C. Comparison of Proposed Algorithm with Existing

The measurement results for all adaptation algorithms are qualitatively summarized in Table II. The meaning of high, medium, and low is relative to the observed measurements. In terms of average playback quality and bandwidth utilization BIEB and Tribler can outperform KLUDCP and TRDA significantly. Both algorithms deliver a high average playback quality to the user, but demonstrate a distinct quality switching behavior. Whereas BIEB can keep the number of quality switches low during the playback, Tribler has to switch to a different presentation nine times more often which may have a severe impact on the QoE. In terms of quality switching frequency BIEB is only outperformed by TRDA. However, the observed durations with steady quality are long enough to have no or little impact on the QoE. Nevertheless, TRDA shows a better efficiency than BIEB, as no data is unnecessarily downloaded. However, compared to the size of the movie, the data discarded by BIEB are negligible and can be prevented through additional segment-picking constraints. From the objective evaluation it follows that BIEB outperforms the other algorithms.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we propose and implement an user-centric DASH/SVC streaming algorithm and compare it with existing algorithms regarding playback quality and efficiency. The comparison is conducted by measurements in our test-bed for a mobile scenario. The results show that the proposed mechanism outperforms the other algorithms in terms of video quality, switching frequency, and usage of the available resources for the investigated network scenario. We are convinced that this work represents an important first step towards true QoE management of video streaming with DASH and SVC. Building on these results, we foresee future work to address different challenges. The quantification of the impact of quality switches on the QoE will be investigated by subjective user studies as fundamental step to fine-tune the parameters of our proposed algorithm. Thereby, the measurement data for quality switching will be used as input for the user tests. Furthermore, the efficiency of the proposed algorithm can and will be improved to avoid wasted video data by setting appropriate constraints in the algorithm. Another open issue is the objective and subjective evaluation in different evaluation scenarios and the use of quality scalability. In particular, the impact of cross-traffic or background traffic as well as multi-user scenarios are of interest. Based on these results the proposed algorithm will be refined and the optimal user-centric DASH/SVC streaming strategy defined.

REFERENCES

[1] Blender Institute. *SVC encoded by Department of Communication Systems, University of Wuerzburg*. Tears Of Steel, http://www.tearsofsteel.org/, SVC encoded and segmented http://http://www-itec.uni-klu.ac.at/ftp/datasets/mmsys12/TearsOfSteel/.

[2] Cisco Systems Inc. Cisco visual networking index: Forecast and methodology, 2011-2016, June 2012.

[3] T. Hoßfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen. Initial Delay vs. Interruptions: Between the Devil and the Deep Blue Sea. In *QoMEX 2012*, Yarra Valley, Australia, July 2012.

[4] T. Hoßfeld, R. Schatz, M. Seufert, M. Hirth, T. Zinner, and P. Tran-Gia. Quantification of youtube QoE via crowdsourcing. In *IEEE International Workshop on Multimedia Quality of Experience - Modeling, Evaluation, and Directions (MQoE 2011)*, Dana Point, CA, USA, December 2011.

[5] R. Huysegems, B. De Vleeschauwer, T. Wu, and W. Van Leekwijck. Svc-based http adaptive streaming. *Bell Labs Technical Journal*, 16(4), 2012.

[6] ISO/IEC. *Information technology Dynamic adaptive streaming over HTTP (DASH) Part 1: Media presentation description and segment formats*, Apr. 2012. ISO/IEC 23009-1:2012(E).

[7] J.-S. Lee, F. De Simone, N. Ramzan, Z. Zhao, E. Kurutepe, T. Sikora, J. Ostermann, E. Izquierdo, and T. Ebrahimi. Subjective evaluation of scalable video coding for content distribution. In *Int. Conf. on Multimedia*, MM '10, New York, NY, USA, 2010.

[8] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz. Adaptation algorithm for adaptive streaming over http. In *19th International Packet Video Workshop (PV 2012)*, Munich, Germany, May 2012.

[9] C. Mueller, D. Renzi, S. Lederer, S. Battista, and C. Timmerer. Using scalable video coding for dynamic adaptive streaming over http in mobile environments. In *Signal Processing Conference (EUSIPCO)*, Bucharest, Romania, Aug. 2012.

[10] C. Müller, S. Lederer, and C. Timmerer. An evaluation of dynamic adaptive streaming over http in vehicular environments. In *4th Workshop on Mobile Video (MoVID 2012)*, Chapel Hill / USA, Feb. 2012.

[11] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. Halvorsen. Flicker effects in adaptive video streaming to handheld devices. In *19th ACM int. conf. on Multimedia (MM 11)*, Scottsdale, AZ, USA, Nov. 2011.

[12] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. Halvorsen. Spatial flicker effect in video scaling. In *Quality of Multimedia Experience (QoMEX), 2011 Third International Workshop on*, sept. 2011.

[13] S. Oechsner, T. Zinner, J. Prokopetz, and T. Hoßfeld. Supporting scalable video codecs in a P2P video-on-demand streaming system. In *21th ITC Specialist Seminar on Multimedia Applications - Traffic, Performance and QoE*, Miyazaki, Japan, March 2010.

[14] T. Oelbaum, H. Schwarz, M. Wien, and T. Wiegand. Subjective performance evaluation of the svc extension of h.264/avc. In *15th IEEE Int. Conf. on Image Processing (ICIP 2008)*, San Diego, California, USA, Oct. 2008.

[15] Y. Sanchez, T. Schierl, C. Hellge, D. Hong, D. D. Vleeschauwer, W. V. Leekwijck, Y. Lelouedec, and T. Wiegand. Improved caching for http-based video on demand using scalable video coding. In *IEEE Consumer Communications and Networking Conference (CCNC 2011)*, Las Vegas, Nevada, USA, Jan. 2011.

[16] Y. Sanchez, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. D. Vleeschauwer, W. V. Leekwijck, and Y. L. Loudec. Efficient http-based streaming using scalable video coding. *Signal Processing: Image Communication*, 27(4), 2012.

[17] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the h.264/avc standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(9), Sept. 2007.

[18] B. Wang, J. Kurose, P. Shenoy, and D. Towsley. Multimedia streaming via tcp: An analytic performance study. *ACM Trans. Multimedia Comput. Commun. Appl.*, 4(2), May 2008.

[19] M. Zink, O. Künzel, J. Schmitt, and R. Steinmetz. Subjective impression of variations in layer encoded videos. In *11th Int. Conf. on Quality of service*, IWQoS'03, Berkeley, CA, USA, 2003.