Signal Processing: *Image Communication* 27 (2012) 329–342

Contents lists available at SciVerse ScienceDirect

Signal Processing: *Image Communication*

journal homepage: www.elsevier.com/locate/image

Efficient HTTP-based streaming using Scalable Video Coding

Y. Sanchez [a,*], T. Schierl [a], C. Hellge [a], T. Wiegand [a], D. Hong [b], D. De Vleeschauwer [c], W. Van Leekwijck [c], Y. Le Louédec [d]

[a] *Fraunhofer HHI, Germany*
[b] *N2N Soft, France*
[c] *Bell Labs, Alcatel-Lucent, Belgium*
[d] *Orange-FT, France*

ARTICLE INFO

Available online 8 October 2011

Keywords:
HTTP streaming
Live
Scalable Video Coding
Adaptation
Caching

ABSTRACT

HTTP-based video streaming has been gaining popularity within the recent years. There are multiple benefits of relying on HTTP/TCP connections, such as the usage of the widely deployed network caches to relieve video servers from sending the same content to a high number of users and the avoidance of traversal issues with firewalls and NATs typical for RTP/UDP-based solutions. Therefore, many service providers resort to adopt HTTP streaming as the basis for their services. In this paper, the benefits of using the Scalable Video Coding (SVC) for a HTTP streaming service are shown, and the SVC based approach is compared to the AVC based approach. We show that network resources are more efficiently used and how the benefits of the traditional techniques can even be heightened by adopting the Scalable Video Coding (SVC) as the video codec for adaptive low delay streaming over HTTP. For the latter small playout-buffers are considered hence allowing low media access latency in the delivery chain and it is shown that adaptation is more effectively performed with the SVC based approach.

1. Introduction

In spite of the general tendency of using the UDP protocol [1] for video and audio real-time delivery due to its lower latency, HTTP streaming has raised the interest of many researchers and service providers in the last years. Relying on HTTP [2]/TCP [3] allows for reusing existing network infrastructures such as the widely deployed network caches, which reduce the amount of outbound traffic that servers have to support and consequently prevent scalability issues with respect to the system size. Although it is technically not difficult to build a similar infrastructure for RTP/UDP, it is much more costly to build it from scratch than to reuse the HTTP infrastructure that already exists. Furthermore, the use of HTTP/TCP resolves the common traversal issues with firewalls and NATs, which typically arise when the data is transmitted over RTP [4]/UDP. Moreover, implementation of HTTP streaming systems is simple, where servers are typical web servers agnostic of the actual data they provide and therefore do not need any special functionality to deal with the media files.

Due to the mentioned benefits of HTTP streaming, there has been a sharp increase in the interests of the market into HTTP streaming. A clear evidence of it is the standardization processes lead by different standardization organizations on this field, such as in MPEG's Dynamic Adaptive Streaming over HTTP (DASH) [5], 3GPP Adaptive Streaming over HTTP [6,7] and OpenIPTV Forum HTTP Streaming specification [8]. There are also different proprietary solutions such as the Adobe's RTMP [9], IIS Smooth Streaming [10] and Apple's Live Streaming [11].

Streaming over HTTP can be simply realized by downloading a whole media file and starting the decoding and presentation process after a certain safe part of the media

* Corresponding author.
E-mail address: yago.sanchez@hhi.fraunhofer.de (Y. Sanchez).

file has been downloaded. In this context "safe" means, that even if during streaming, the available download rate is temporarily lower than the actual required rate for the media, the content can be played out without any interruptions. This assumes that from time to time also higher download rates than the actual media rate are available. This approach is also known as progressive download in Video on Demand (VoD). An improvement to this approach is not to download the whole media file at once, but to download it in chunks related to certain time intervals of the media content. This is sometimes also called chunky or chunk-based streaming. Having access to chunks of the actual media file allows for adaptive streaming over HTTP. In adaptive HTTP streaming, the receiver is responsible for initiating the media download and performing adaptation, if required. Adaptation is performed by requesting a different representation (version of the media data at a given bitrate) among the multiple available representations, each of which has a different bitrate. That is, each chunk of the media corresponding to a certain interval of the media playout time is available at different encoding rates. The receiver selects on the fly, which rate is the most appropriate at a certain point of time, e.g., this rate matches the current network path conditions or the capabilities of the receiver at best. There are different ways of providing multiple representations of a media. One method may be to encode the media data at multiple bitrates with a single layer codec such as AVC [12] for video, which requires a representation to be a complete and independently encoded part of the media. Another method may be encoding the media with a scalable media coding method such as provided by SVC [13] for video, which allows storing layers of the video as different representations, i.e. the representations are additive to each other. In any case, a description of the characteristics of the media, such as bandwidth of a representation or language of an audio file, has to be provided to the client, so that the client is aware of all representations at the server and can choose the one which matches at best its capabilities and interests. Such a selection is frequently performed and can be adapted during streaming.

In previous works [14,15], we have already presented benefits of using Scalable Video Coding (SVC) for HTTP streaming. The benefits have been shown in terms of web caching efficiency and saved uplink bandwidth at the server in comparison to the use of H.264/AVC. In this work, we summarize benefits related to web caching and we show additional benefits, i.e. the faster response to network throughput variations and the better match to the available network resources in live services.

The remainder of this paper is organized as follows. In Section 2 a short overview of MPEG's Dynamic Adaptive Streaming over HTTP (DASH) standard is presented. Section 3 explains the behavior of the receiver in DASH. In Section 4 the Scalable Video Coding (SVC) is introduced and its applicability to DASH is pointed out. Section 5 summarizes the benefits in terms of caching efficiency for DASH-based Internet TV services using SVC. Section 6 describes an scheduling mechanism using SVC in terms of rate adaptation for a general case. In Section 7, an analysis of the issues specific to live streaming services is carried out. Section 8 describes specific issues on adaptation in a live streaming scenario and the benefits of using SVC-based HTTP streaming compared to AVC-based HTTP streaming are shown. In Section 9 the conclusions of this work are summarized.

2. DASH

Dynamic Adaptive Streaming over HTTP (DASH) [5] is an emerging MPEG-Standard, which defines a format for multimedia delivery over HTTP. It basically consists of two elements: the format of the media to be downloaded and the description of the media to be downloaded—the Media Presentation Description (MPD).

The media file as a whole is divided for delivery into smaller parts, called segments, previously referred to as chunks. The format of the file segments, which are the resources assigned to an HTTP-URL for download (possibly with an additional byte-range HTTP parameter [2]), are defined as follows. In DASH [5] two container formats are considered for data encapsulation: MPEG-2 Transport Stream [16] and ISO base media File Format [17]. Furthermore, a guideline for extensibility is specified in order to allow other formats to be used in combination with DASH.

For any format used with DASH, there are different types of segments where the basic ones are the following two (for more information about further segment types the reader is referred to [5]): initialization segments and media segments. The former are segments that contain all the initialization information necessary for accessing the media data. Initialization segments contain Program Association Tables (PAT), Program MAP Tables (PMT) and Conditional Access Tables (CAT) for MPEG2-TS [16] or the File Type box ('ftyp'), and Movie Box ('moov') with a Movie Extension Box ('mvex') indicating the presence of movie fragments for ISO base media File Format (ISOMFF) [17]. For more information about initialization information the reader is referred to the DASH standard [5] and to MPEG2-TS [16] and ISOMFF [17] standards. The media segments correspond to the actual media data which may be contained in MPEG-2 Transport Stream [16] or the ISOMFF [17]. The use of these two container formats provides some additional metadata describing the media, such as timing information or position of access units (AU) within the segment. Media segments may also be self-initializing, which means that there is no separate initialization segment required and the initializing information is contained within the media segment prior to the media data.

The Media Presentation Description (MPD) is an XML document that describes the media available at the server, so that the client can make the selection of the media that matches at best its requirements and equipment/network capabilities. As shown in Fig. 1, the presentation time in DASH is logically divided into smaller time intervals called periods (one or more). Each period contains different media components, such as audio or video at different version, which are collected into representations. Each representation is further structured as a sequence of one or more segments. For more information about the organization of the MPD, the reader is referred to Ref. [5].

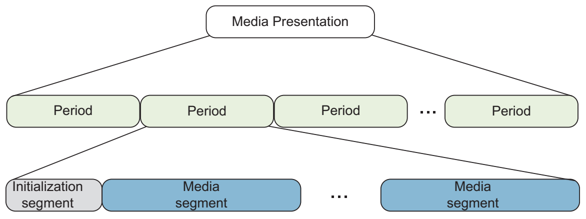


Fig. 1. Media presentation time organization.

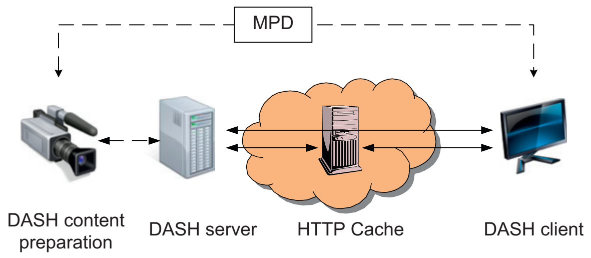


Fig. 2. Example for DASH architecture.

Although segments are the smallest entity addressed in the MPD smaller parts of the media presentation may be defined, which are called sub-segments. Sub-segments are a set of complete access units within a segment. In DASH the segment index box ('sidx') defined in [21] is used for signaling of subsegments, both for MPEG2-TS and ISOMFF. In the ISOMFF case, further restrictions are applied to sub-segments, i.e. a sub-segment consists of one or more movie fragments [17] (as for Segments) with their corresponding movie fragment boxes ('moof') and related media data boxes ('mdat'), being the smallest sub-segment a single movie fragment. As mentioned, how to access these sub-segments is not described in the MPD but within the media container itself, where the segment index box ('sidx') [21] is added. For more information on the usage of 'sidx' and indexing information for sub-segments with each media format, the user is referred to Ref. [5]. In Ref. [21] the 'ssix' box is a further signaling method, which may be used to access fractions of sub-segments, for instance for trick modes, which characteristics may be described in the MPD as subRepresentations.

Effective transport of SVC with DASH is achieved by offering different layers of SVC in different representations. Since SVC layers share some dependencies, i.e. enhancement layers depend on lower layers for decoding, representations containing enhancement layers are called *dependent Representation* [5]. The dependencies are indicated in the MPD and presentation issues are solved at encapsulation level. SVC content encapsulation in both MPEG2-TS and ISOMFF are well defined. Further information about SVC encapsulation for MPEG2-TS and ISOBMFF can be found in Refs. [18–20].

In Fig. 2, a possible DASH architecture is shown. It consists of a DASH content preparation component, which is responsible for preparing the segments and MPD, a DASH server, which is a normal web server where the DASH segments are stored, and, where the client can access for downloading them and possibly also the MPD, possibly a HTTP cache, which is useful to relieve the load on server when many clients try to access data, and the DASH client, which fetches the segments and perform the appropriate operations so that the content can be presented to the user.

HTTP caching allows reducing the scalability issues of the service, since the outgoing traffic at the server is decreased and therefore it is of great importance. As shown in Refs. [14,15] and later summarized in Section 5, the effectiveness of the HTTP caches can be considerably improved by considering SVC, reducing the amount of data that has to be transmitted between server and caches, which is tremendously beneficial for content providers, since the same variety of video content can be provided at a reduced cost compared to the usage of single layer coded data, as e.g. AVC.

3. Receiver's behavior

In this section, the behavior of a DASH receiver is described. Note, however, that the DASH receiver is not standarized in Ref. [5]. For this purpose the DASH client is divided into two logical components as shown in Fig. 3: DASH Access client and MPEG Media Engine [5].

The DASH client is responsible for two principle tasks. It has to manage download of the media data available at the server and present it to the users correctly. In order to do so, it has to select the most appropriate media representations among the ones described within the MPD, download them and organize the received segments in such a way that the data can correctly be rendered by the MPEG Media Engine. The DASH Access client is responsible for performing the adequate segment requests based on download rate estimates of the current network, as well as user equipment characteristics. Further, the access client is responsible for passing the received data to the MPEG Media Engine in the right order so that the media data can be decoded correctly and presented.

In the block diagram depicted in Fig. 4, a possible structure of the **DASH access client** is shown, which consists of six logical modules. The HTTP module is responsible for issuing the HTTP GET requests in order to download data based on the selections made by the scheduler/downloading controller. These selections are translated by the HTTP module into the corresponding HTTP requests with the help of the MPD parser and 'sidx'/'ssix' parser. The MPD parser parses the MPD extracting all the necessary information, such as available media representations, the URLs at which the representation can be accessed and the necessary bitrate required for downloading them. The 'sidx'/'ssix' parser is used to allow downloading smaller parts than a segment, i.e. a sub-segment or a smaller part thereof,

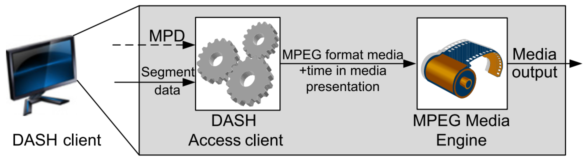


Fig. 3. DASH client.

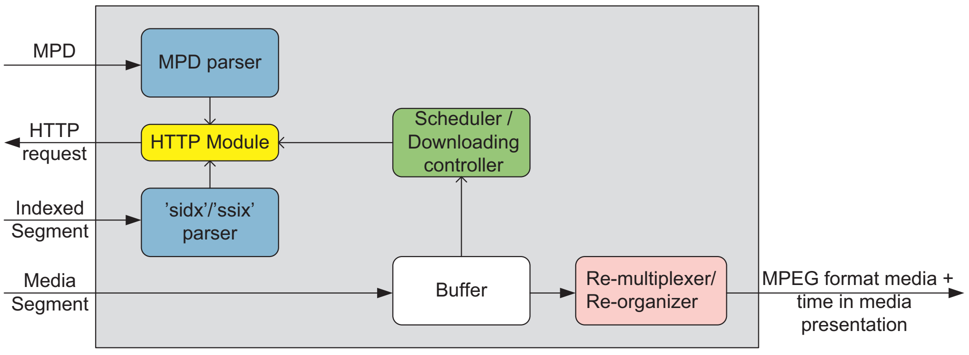

MPD
MPD parser
HTTP request
HTTP Module
Scheduler / Downloading controller
Indexed Segment
'sidx'/'ssix' parser
Media Segment
Buffer
Re-multiplexer/ Re-organizer
MPEG format media + time in media presentation

Fig. 4. DASH Access Client block diagram.

performing HTTP partial GET commands, created by combining a byte-range, which is extracted from the 'sidx' or 'ssix' boxes in the file format, and the original URL of a segment of the corresponding representation appearing in the MPD.

The scheduler and downloading controller is responsible for estimating the available throughput for selecting segments to be downloaded. This module may determine such selections by measuring the time previous segments required for download and by checking the available playout time of media data stored in the buffer. The scheduler may also decide how such segments are requested, e.g., in a sequential manner within a single TCP connection or in parallel over multiple concurrent TCP connections, which may increase the download rates for multimedia streaming [22,23]. The scheduler is also responsible for deciding how to request the data over a heterogeneous network, such as shown in Ref. [24], or for performing prioritization in download of the media as presented in Section 6 and in Ref. [25]. Note that the order of the downloaded data depends on decisions taken by the scheduler. Therefore, the received data may not be downloaded in the right processing order. The re-multiplexer or re-organizer organizes data required for playback by the MPEG Media Engine, and provides the media data in the correct processing order. In case of MPEG-2 TS, this instance may re-multiplex audio and video data as well as layers of SVC (downloaded in separate segments) if necessary so that a legacy MPEG-2 TS decoder is able to decode the data correctly. In case of ISO base file format, sub-segments of *dependent Representations* are interleaved with the sub-segments of its *complementary Representations* so that it results in a conformant ISOBMFF. For a more detail description of the interleaving process for *dependent Representation* the reader is referred to Ref. [5].

The **MPEG Media Engine** for DASH (cf. Fig. 3) typically needs some advanced capabilities in order to be able to play back the content received chunk-wise.

The engine has to cope with possible gaps in the data, i.e. segments missing, result of omissions of requests to the missing data when problems are detected in the network or overlapping of segments when switching from one representation to another, i.e. when segments among different representation do not belong to the same time interval. Other advance capabilities may be playing back different tracks in case of ISOBMFF depending on the downloaded version, etc.

4. Scalable Video Coding

The scalable extension of H.264/AVC (SVC) [13] provides features to allow for different representations of the same video within the same bit stream by selecting a valid sub-stream. SVC supports the concept of layers, which correspond to different quality, spatial or temporal representations. A SVC stream is composed of a H.264/AVC-compatible base layer, which corresponds to the lowest representation, and one or more enhancement layers, which increase the SNR fidelity, spatial and/or temporal quality of the representation when added to the base layer. SVC allows for multiple Operation Points (OP) within the same bit stream. An OP refers to a valid sub-stream at a certain quality, spatial, and temporal level and corresponding to a specific bit rate point.

In case of SNR scalability, one way for obtaining different OPs is to encode multiple quality layers either with Coarse-Grain Scalability (CGS) or Medium-Grain Scalability (MGS) [13], and select each layer as one OP. Each additionally integrated quality layer has a negative influence on the coding efficiency of the SVC stream to some extent. However, if addressed properly the overhead can be kept below 10% as shown in Ref. [26]. A possibility to achieve a low overhead would be to keep the number of layers at a reduced value and create OPs by selecting parts of the bitstream smaller than complete layers, i.e. enhancement layers may be skipped only for some of the frames. Such an approach may result in frames to be decoded with a variable number of layers.

CGS and MGS differ basically in the fact that for CGS the number of layers to be decoded is required to be constant for all frames and reconstruction is done for the highest layer only, while MGS allows for more flexibility and always reconstructs up to the highest quality layer received. Hence, for CGS, special care has to be taken at the decoder if scalability is achieved by dropping enhancements of some of the frames, so that error concealment for CGS is integrated and always the highest received quality is used for reconstruction, similar as specified for MGS in Ref. [13].

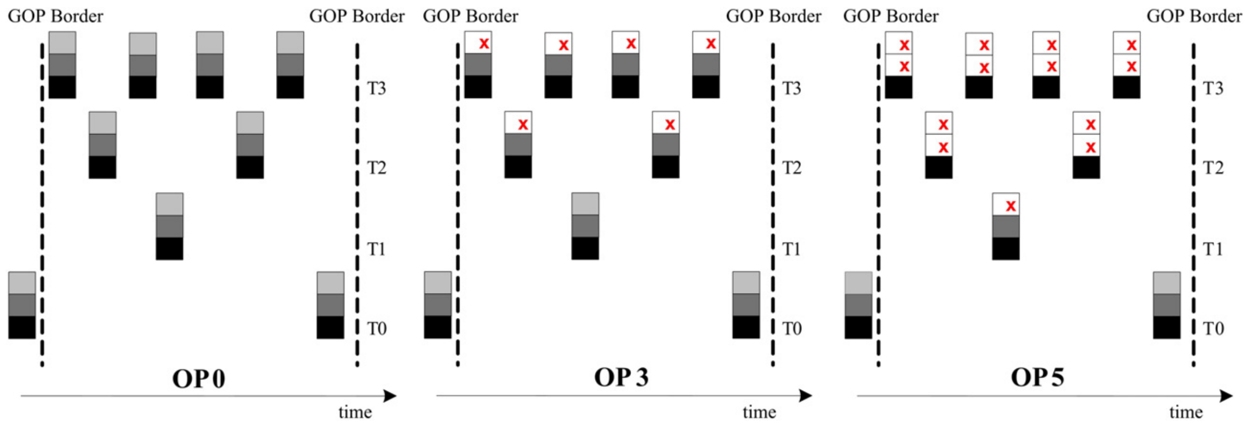


Fig. 5. Example OPs creation based on sub-layers with three OPs.

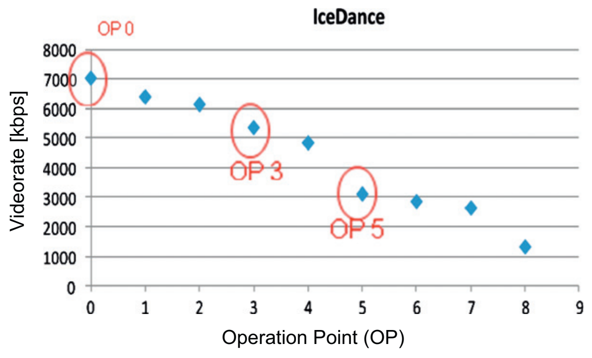


Fig. 6. Rate distribution of OPs.

In order to achieve multiple OPs we encode quality layers with a reduced number layers and select a sub-stream of the original data by additionally selecting smaller parts of a layer, also known as temporal levels in SVC [13]. This allows keeping the overall coding overhead within a suitable range. Fig. 5 illustrates how different OPs are obtained in this way. In the example, the SVC stream is comprised of base layer (black parts of the rectangle), and two quality enhancement layers (dark grey and light grey parts of the rectangle) for the coded pictures (rectangles in the figure). The different OPs are obtained by dropping enhancement layer packets from the highest temporal levels (indicated by Tx in the figure). Fig. 6 shows the bit rates of 9 OPs for the ITU-T test sequence "IceDance" at 720p HD resolution, 50 frames per second and a GOP 8 coding structure. It can be seen that dropping the "light grey" marked parts corresponding to the second quality layer in temporal levels T2 and T3 (cf. Fig. 5) reduces the video rate from 7 Mbps (cf. OP0) to 5.3 Mbps (cf. OP3). Further, dropping the "light grey" marked quality layer from all temporal levels but the temporal level T0 and removing the "dark grey" marked first quality layer pictures from temporal levels T2 and T3 results in a video rate of 3.1 Mbps (cf. OP5). In general, several OPs can be selected for bit rate optimization.

5. Caching efficiency

The use of HTTP caches within the delivery network is very beneficial, since it allows for reducing the uplink traffic at the servers. Frequently requested content, which is expected to be requested in a near future, is stored in a cache so that subsequent requests for the same content are served by a cache entity instead of by the server. Caching has been proved to be extremely beneficial in VoD scenarios. One of the main complexities in optimizing the cache performance, especially for the VoD case, where user requests are more unpredictable, is the difficulty to predict future user requests so that the files that maximize the amount of data served by the cache entity are kept in the cache. However, for the live streaming case, the request pattern is simpler: the data is only of interest for a very short period of time and thereafter is not expected to be useful anymore and therefore can be removed from the caches. Thus, dealing with live content is simple for the caches and surely also beneficial, where the cache storage can be utilized for storing and forwarding content to multiple users, similar to an overlay multicast system.

The efficiency of deployed HTTP caches can be measured by the cache-hit-ratio, which shows the proportion of HTTP requests that can be served by the caching entities. The cache-hit-ratio is directly related to the reduction in outbound traffic at the origin server. It is mainly influenced by the storage capacity of the caches, the applied caching replacement algorithms, and the number of different content requested by the users. With DASH, the number of different content is significantly increased due to the additional number of representations (bitrate versions) required for each content to provide a smooth bitrate adaptation. Encoding each representation independently with single layer H.264/AVC drastically reduces the caching-efficiency as shown in Refs. [14,15] if compared to the case where only a single representation is offered per content and all clients request the same version for each content.

Previous works [14,15] have shown that the caching efficiency can be improved just by using SVC as the video codec for a VoD adaptive streaming service over HTTP, while keeping the same cache replacement algorithm. There are two reasons for this gain in caching efficiency:

- SVC removes redundancy between different media representations of the same content by utilizing different inter-layer prediction methods [13]. Therefore, with SVC, the cumulative bitrate of the required media representations is reduced and for the same storage capacity a higher number of representations can be cached than in the case where representation are independently encoded with single layer H.264/AVC.
- With SVC, more clients request the same data (layers) since, although requesting different representations of a same video, the clients request a set of layers, with some layers in common, e.g. the base layer. This is due to the hierarchical coding dependency of the multi-layered SVC. Thus, all requests for a single content incorporate at least the base layer representation. Consequently, the probability of a cache-hit for files containing the lowest layers of SVC streams, which most of the users are interested in, is increased.

The cache-hit-ratio is further influenced by the cache replacement algorithm. Common caching algorithms used in practical applications are Least Recently Used (LRU) and Least frequently used (LFU) algorithm [27]. Numerous caching algorithms exist, which aim to optimize the caching performance based on a certain metric or criteria [27]. One exemplary algorithm, designed for chunk-based streaming, the Chunk-based Caching (CC) algorithm [28], is compared in Ref. [15] to the LRU algorithm. CC shows to improve the cache-hit-ratio compared to LRU in scenarios, where media is delivered in chunks. Furthermore, by combining CC with SVC gains in caching efficiency are obtained. The results shown in the following correspond to the LRU case, and show that just by using SVC the caching performance in terms of cache-hit-ratio is notably enhanced, while applying a simple cache replacement algorithm.

Figs. 7–9 show the average cache-hit-ratio of a Video on Demand (VoD) service using DASH based on SVC, in the figures referred to as SVC–VoD, in comparison to DASH based on the single layer codec H.264/AVC, in the

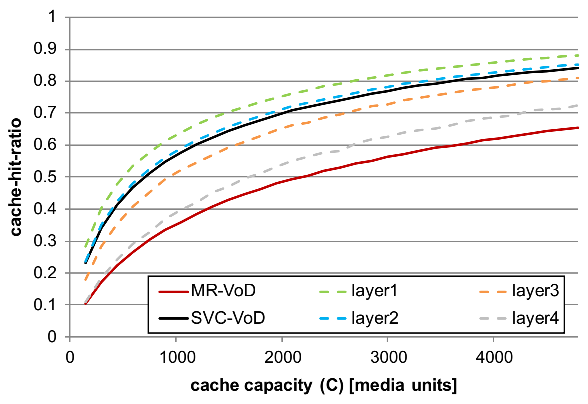


Fig. 7. Cache performance for users with different equipment capabilities.

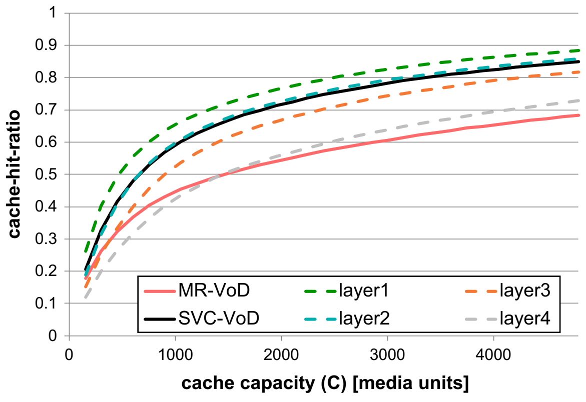


Fig. 8. Cache performance for a scenario with heavy cross traffic.

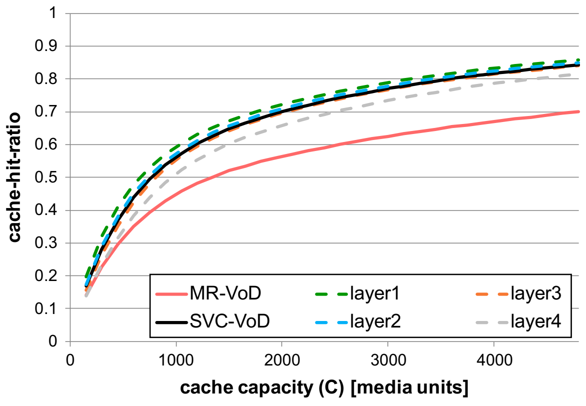


Fig. 9. Cache performance for a scenario with light cross traffic.

figures referred to as MR–VoD (Multiple Representation-VoD). In these figures, the average cache-hit-ratio over different values of overall cache capacity is shown. The cache capacity, i.e. storage size, is measured in media units, which are equivalent to the size of a video clip of 90 min at 500 kbps (1 media unit=337.5 MB). The simulations are based on real data statistics extracted from a real VoD service. For further details on the simulation assumptions the reader is referred to Refs. [14,15].

The results in Fig. 7 have been already presented in Ref. [14]. In the conducted evaluation, four different types of users have been considered. Each type of user has a different display resolution and requests a different representation. Furthermore, a uniform distribution of the users is considered, which means that each user type is responsible for 25% of the requests. In Fig. 7, it can be seen that the average cache-hit-ratio for SVC–VoD is much higher (about 20%) than for MR–VoD. Also, the average cache-hit-ratio for each of the layers of SVC is higher than the average cache-hit-ratio for MR–VoD, where the cache-hit-ratio for the base layer (layer 1 in the figure) is higher than the average cache-hit-ratio for each of the higher layers (layer 2 to layer 4 in the figure) and up to 15% higher than the cache-hit-ratio for the highest enhancement layer (layer 4). This effect is due to the caching gain over simul-storage of the MR–VoD data.

Figs. 8 and 9 show results for users with the same equipment capabilities but with a varying throughput. To keep the service presentation without interruption users need to dynamically adapt their requests to representations that match the available throughput. The results presented in Fig. 8 correspond to a "heavy cross traffic" in the access links, whereas the results in Fig. 9 correspond to a "light cross traffic" scenario. The impact of the cross-traffic within the "heavy cross traffic scenario" can be summarized as users requesting 25% of the time each of the available representations, with a mean stable time (i.e. time requesting a certain representation without performing adaptation) of 40 min. In the case of the "light cross traffic" scenario users request the two lowest representations around 9% of the time, the second highest representation around 19% of the time and above 62% of the time the highest representation. The mean stable time for each representation is about 2 min for the lowest two representations, 10 min for the second highest representation and 40 min for the highest representation. For more information about the cross traffic modeling, the reader is referred to Ref. [15].

In Figs. 8 and 9 it can be shown that the results are quite similar to the ones presented in Fig. 7, i.e. SVC–VoD outperforms MR–VoD. Since for "heavy cross traffic" the diversity in request is higher than in the "light cross traffic" case the gain in terms of cache-hit-ratio is higher for the former case. In the "light cross traffic" case, most of the time users request the highest quality representation. Nonetheless, in both cases this gain is significant and above 15%.

6. Rate adaptation

As shown in Section 5 and in Refs. [14,15], SVC leads to great improvements of an HTTP streaming service in terms of cache efficiency, which reduces the outbound traffic at the server and the required throughput within the delivery network. Thus, it allows service providers to reduce operational expenditures or improve the average service quality at the same costs compared to single layer H.264/AVC. However, in Refs. [14,15], the advantages for users was not shown, e.g. an enhanced adaptability compared to approaches based on single layer media codecs.

In Ref. [25], a scheduling mechanism called Priority-based Media Delivery (PMD) is presented, which aims to prioritize the most important data (e.g., lower layers in SVC). Download for additional SVC layers is initiated only if the more important layers meet specific buffering constraints (minimum buffer fullness for a defined buffer level). In Ref. [25], the benefits of the combination of PMD and SVC are shown in a typical scenario, where users have limited resources, e.g., buffering/storage capacity. It is further shown that with the presented combination it is possible to react faster to network variations than with simple Multiple Representation Streaming (MR-Streaming) with H.264/AVC, thereby improving the video quality at periods with reduced available download rate. MR-Streaming is a more general term than MR–VoD (cf. Section 5), which does not only refer to VoD services, but also Multiple-Representations Live Streaming (MR-Live). In the following we will refer to MR-Streaming for both MR–VoD and MR-Live. Analogously, SVC-PMD comprises both SVC–VoD (cf. Section 5) and SVC-Live and refers to the application of the PMD technique in combination with SVC.

The PMD technique can be implemented in the Scheduler/Download controller module of the block diagram in Section 3 (cf. Fig. 4). The flowchart diagram in Fig. 10 describes the working principle of the PMD algorithm, where i refers to current operation point of a layered file and Buff[x] refers to the playout-buffer for a certain operation point x in the client. The simple algorithm always tries starting from the most important operation point $i=0$ to fill up the priority buffers. Only if all buffers are meeting the target fullness, the algorithm fills up the operation point with the lowest fullness, i.e. the data with the closest playout deadline is downloaded.

Defining the buffer levels which dominate the PMD performance is subject to optimization of the specific service requirements. E.g. in today's VoD systems, higher video playout robustness is preferred over a low startup delay, i.e. VoD systems typically employ a playout-buffer to be able to overcome jitter or connection problems during streaming, while for live services low startup delays and latency to the live signal are very important so typically small playout-buffers are defined. Given defined levels for a service the working principle is as follows. The playout-buffer is built during the pre-buffering phase at least once at the beginning of streaming. In this phase usually no data is played back, although other approaches start playing back immediately and build the playout-buffer while playing back the media. Note that the smallest playout-buffer corresponds to the segment length (or sub-segment length cf. Section 2), since this amount of data is received as response to an HTTP GET request. The pre-buffering phase can be re-entered at any point of the streaming service if temporarily the received data rate has been below the consumed media rate (playout rate) and the desired playout-buffer fullness is not achieved. If the pre-buffering phase is re-entered, users may request media of a lower bitrate than the available network throughput to use the additional throughput for downloading data to build up the playout-buffer. The playout-buffer allows a stable service and reduces the need for quality adaptation within a certain timeframe determined by the playout-buffer length. However, a good system design requires a trade-off between playout-buffer length and startup delay. Also the different capabilities of target receivers, e.g. set-top boxes and mobile devices, in terms of storage capacity must be taken into account.

For MR-Streaming, there is no logical division of the buffer into different buffer levels for each video rate. Instead, adaptation is performed by requesting alternative encodings of the data at a different bitrate to reach the requirements of the unique logical buffer. For this purpose different values of filling level of the playout-buffer are defined which are used as indicators for performing adaptation. Such values, referred to as adaptation-thresholds, denote the filling level of the buffer at which a lower rate is requested and re-buffering is performed until the buffer has been refilled.

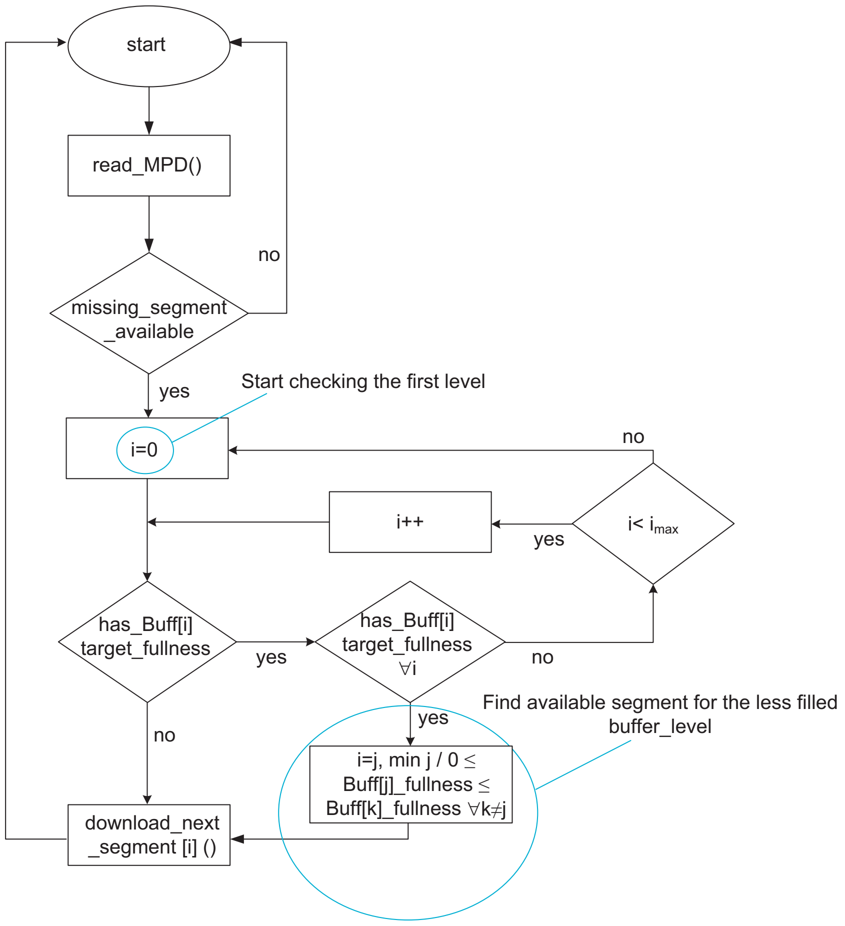

start
read_MPD()
missing_segment _available
no
yes
Start checking the first level
i=0
no
i++
yes
i< i_max
has_Buff[i] target_fullness
yes
has_Buff[i] target_fullness ∀i
no
Find available segment for the less filled buffer_level
yes
no
i=j, min j / 0 ≤ Buff[j]_fullness ≤ Buff[k]_fullness ∀k≠j
download_next _segment [i] ()

Fig. 10. Flowchart for PMD.

7. Considerations on live streaming

As aforementioned, for live streaming the possible values for target fullness for each level in the playout-buffer are restricted by the acceptable latency compared to the actual live signal, i.e. the values of required target fullness for each level in the playout-buffer cannot be higher than the acceptable latency. In fact, trying to reduce the latency and the playout delay to the minimum would lead to preventing from buffering multiple segments of the downloaded content, and SVC-PMD would be applied on a per segment basis, as well as MR-Streaming.

In a live scenario, the downloaded segments of the media data need to be of a small duration to reduce the capturing and transport delay and thereby the latency to an acceptable level. Furthermore, the smaller the segments are the faster adaptation can be performed, which is very important due to the reduced playout-buffer, i.e. if adaptation is not performed in a fast enough manner playout interruptions may occur. As a consequence, the download rate of each of the video segments varies from segment to segment due to the behavior of TCP [3]. We consider TCP Reno [29,30], since it is the most popular implementation today. The difference to other TCP implementations lies on the congestion avoidance phase. In the congestion avoidance phase, typically one additional packet is sent per round compared to the round before. One "round" lasts a round trip time, which is the time elapsing between sending a packet and the ACK for that packet arriving. In other words, the congestion window, which is the size of a logical window equal to the number of packets sent in a burst (unACKed in the network), is incremented by one if all the packets sent in the previous burst (round) are acknowledged. If not all packets in the previous round are acknowledged different actions may be taken. For TCP Reno, if triple-duplicate ACKs (three or more acknowledgements for the same packet are received) the window size is halved and the congestion avoidance phase is restarted. Triple-duplicates are received if at least three packets are received after a packet loss (all these received packets acknowledge the last received packet before the lost packet). If on the other hand, less than three duplicate ACKs are received a time-out would occur and the TCP connection would start with a slow start phase.

Due to the congestion control algorithm of TCP, the download rate of the segments varies continuously and cannot be considered to be the average transmission rate, as possible for big data sizes, i.e. data much bigger than the average transmitted data during rate adaptation cycles (time between two window size reduction events) of the TCP connection. In Fig. 11 the rate of a TCP session is illustrated, as well as the effect of the small segment sizes (compared to the adaptation cycles of TCP) on the segment download rate. We neglected the fact that if the window increases the queuing delay increases as well as the round trip time. This would make the additive increase observed in TCP slightly less than the linear increase shown in the figure.

While being a necessary and beneficial issue for live streaming, small video segments lead to a highly varying segment download rate and therefore a varying segment fetch time. This effect can be denoted as segment jitter. Analogously to the packet jitter, this effect could be countered by using a buffer and consequently delaying the startup and staying further beyond the live signal.

In Fig. 12, the segment download rate is presented for a segment size of 0.5, 1 and 5 s at a fixed video rate of 2 Mbps. The presented download rates have been directly measured in the network simulator NS-2 [32], where a packet loss of 1% is considered, which corresponds to a theoretical TCP throughput of 2.8 Mbps [33].

In Fig. 12, it can be noticed that the smaller the segment lengths are the larger the variability in the segment download rate is, as presented for the constant 2 Mbps bitrate video. Since fast adaptability is desired, we will focus on segments of 0.5 s in the following. Fig. 13 illustrates the playout-buffer fullness distribution for different channel conditions and different downloaded media rates.

Fig. 13 shows the playout-buffer fullness over different values of downloaded media rate for two different scenarios with different throughput. The plot on the right corresponds to an average available TCP throughput slightly higher than 2.2 Mbps (i.e. packet dropping rate of 1.2%) and the plot on the left corresponds to an average available TCP throughput of around 4.2 Mbps (i.e. a packet dropping rate of 0.4%). The different colors in the figure correspond to the percentages of the time for which the playout-buffer has less data stored than the value marked in the Y-axis of the figure. It can be seen that for media rates closer to the average available throughput the variation of playout-buffer fullness is bigger and the buffer-level is with a high probability at a low value. Note that the variation is related to the download rate not being constant and different to the downloaded media rate and does not directly depend on the maximal buffered media data (10 s in Fig. 13). For different values of latency, being increased or decreased, the presented values should be therefore shifted up or down, respectively.

Since the buffered amount of data is relatively small, rate estimation cannot be performed based on long measured statistics. Smarter estimation and scheduling is needed so that fast reactivity can be achieved and long periods to detect channel state changes can be avoided, which is crucial for preventing playout interruptions, while at the same time avoiding unnecessary switching. This fact in combination with the presented buffer performance leads to the following conclusion:

Media rates close to the average available throughput make it more difficult to detect variations in the available throughput, since variation in the fullness of the buffer due to varying segment download rate and due to a higher cross traffic in the network are difficult to differentiate. Therefore, if a media rate close to the available throughput is chosen, MR-Streaming is expected not to be able to detect throughput variations fast enough resulting in a high playout interruption frequency.

8. Adaptation vs. live latency

Adaptation in live streaming scenarios is a more complicated issue than in VoD scenarios. Due to the lack of an extensive playout-buffer, decisions have to be made in a much faster way than in VoD services. In the following we will differentiate two scenarios: the first one where a reasonable playout delay is tolerated and the

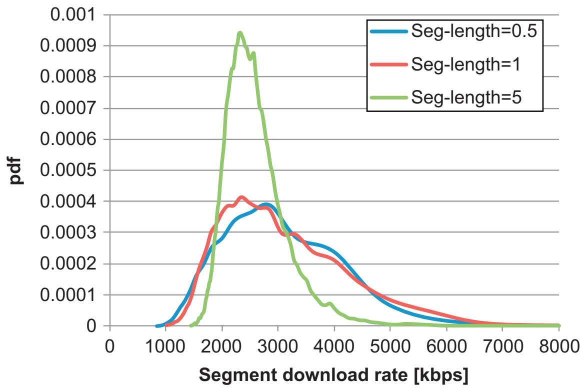


Fig. 12. Segment download rate for 0.5, 1 and 5 s segments at 2 Mbps video rate.

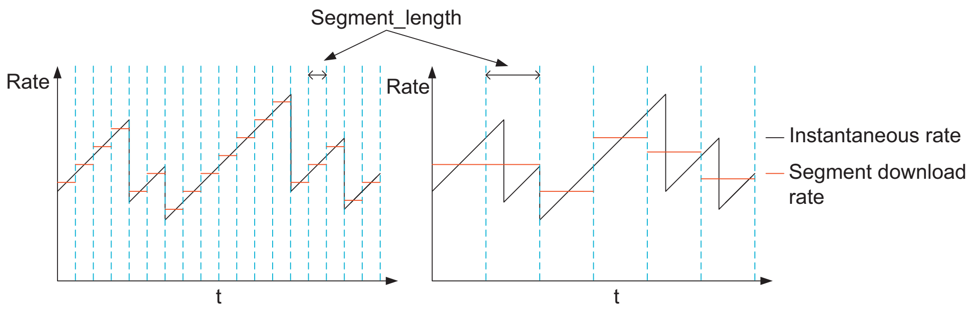


Fig. 11. Effect of the size of the segment on the segment download rate.

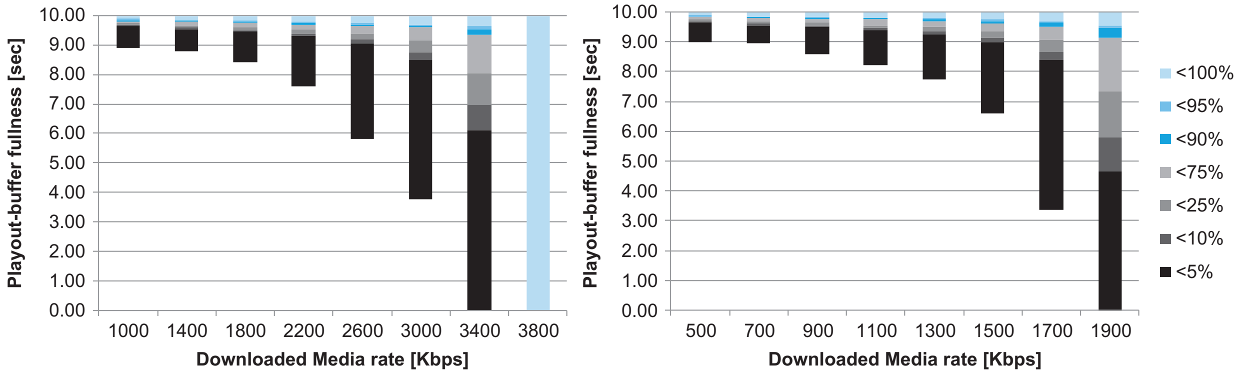


Fig. 13. Playout-buffer fullness distribution for different downloaded media rates for a packet dropping of 1.2% (right) and 0.4% (left) and a latency of 10 s to the live signal.

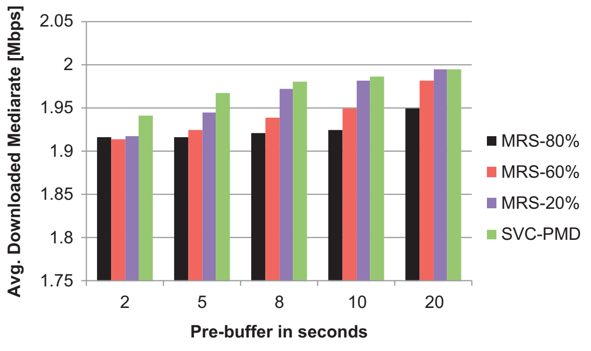


Fig. 14. Avg. download rates for MR-Streaming and SVC-PMD over different playout-buffer values.

media is played behind the live content and a second scenario with very stringent delay where the playout delay is minimized to the maximum possible, i.e. no segments are stored in the playout-buffer.

8.1. *Live streaming with an acceptable latency of some seconds*

In Fig. 14 we compare the MR-Streaming with the SVC-Streaming approach for different latency values for playout (2, 5, 8, 10 and 20). MRS-X% represents the MR-Streaming technique with an adaptation-threshold equal to X% of the playout-buffer size, which means that when the buffer-level is lower than X% of the playout latency adaptation to a lower quality is performed. For the theoretical results presented in Fig. 11, a two state Markov chain has been simulated. The two states denote the average available throughput of a user of either 2.2 Mbps or 1.1 Mbps and consequently a user downloads a 1 Mbps or a 2 Mbps video. An ideal exploitation of the network transmission rate is assumed, not taking into account TCP overhead and efficiency. The simulation step is equal to 1 ms and the mean time in the 1.1 Mbps state and 2.2 Mbps state are 5 and 50 s, respectively.

The results in Fig. 14. show that using SVC-PMD in average a higher video rate can be downloaded, especially for low values of playout-buffer, since adaptation to higher qualities can also be performed for already buffered data, i.e. enhancing base layer data. For MR-Streaming it is assumed that the pre-buffered data is played out and no adaptation to higher bitrates is possible for already buffered data. Furthermore, it can be seen that for lower values of adaptation-threshold for MR-Streaming, the difference to SVC-PMD in terms of downloaded video rate is reduced. But in case of more drastic throughput variations in presence of a low amount of pre-buffered data, such as possible in mobile environments, the playout-buffer would have less data for MR-Streaming with a low adaptation-threshold and the risk of running out of data would be higher.

Note that for both, SVC-PMD and MR-Streaming, the same video rate distributions are considered in the presented experiment: 1 Mbps and 2 Mbps. In this discussion, we are not focusing on the average video quality (e.g. in terms of PSNR), where the SVC video at 2 Mbps would have a slightly lower quality than the AVC video at 2 Mbps, due to the 10% overhead of SVC. However, the focus of this discussion lies on the ability to download the highest video quality with a higher probability when using SVC-PMD than when using the MR-Streaming approach, where users switch to the lowest quality more frequently and during longer periods of time.

8.2. *Live streaming with no delay (minimal playout buffer of a segment)*

If focusing on the drastic low-delay case where the playout-buffer is limited to a single segment of 0.5 s the adaptation becomes even more complicated. This is a rather extreme use case but may be necessary in some cases such as sport events, which are watched by large groups of the population and some clients do it via typical broadcast channels. The users may perceive some events after becoming aware of them through other means, e.g. the exclamation of spectators in the vicinity after a soccer goal if the live latency is not kept low. Another possible scenario is interactive services where the spectators may have the possibility to call in real-time to the delivered show. In such scenarios low system latency is highly desirable. As aforementioned the segment download rate varies highly from one segment to another. The perceived segment download rate distributions of a link for three

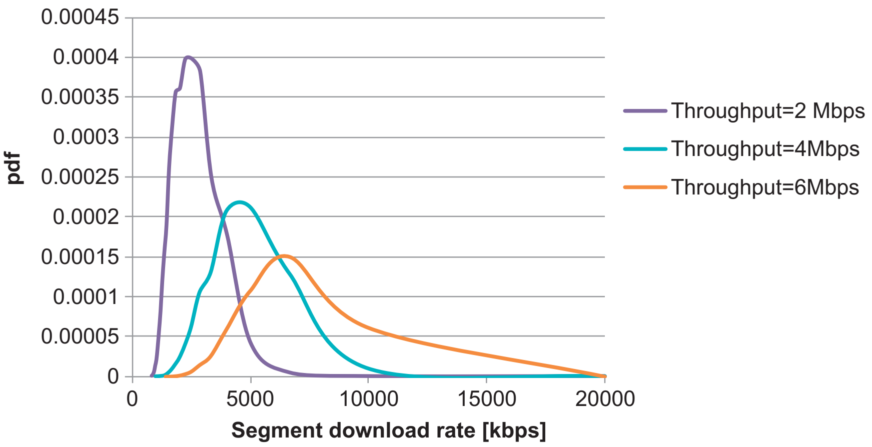


Fig. 15. Segment download rate distribution for different throughput values for 0.5 s segments at 2 Mbps video rate.

Table 1
Selected rates for the representations.

	2 Mbps ($p=2\%$)	4 Mbps ($p=0.5\%$)	6 Mbps ($p=0.22\%$)
MR-Streaming (Kbps)	860	1300	1800
SVC-PMD lowest OP (Kbps)	860	1300	1800
Additionally requested SVC enhancement layer (Kbps)	200	366	400
Avg. Rate for SVC-PMD (Kbps)	1056	1658	2192

different values of available throughput (2 Mbps, 4 Mbps and 6 Mbps) are exemplarily depicted in Fig. 15 for a segment length of 0.5 s. In order to obtain these throughput values a packet loss in the network of $p=2\%$, $p=0.5\%$ and $p=0.22\%$ have been simulated, respectively.

For instance, if we focus on the results for an average throughput of 2 Mbps, segments suffer in about 25% of time a download rate lower than 2 Mbps, resulting in the video being stopped with a high frequency, which is unacceptable for a service.

A possible way to tackle the problem of the high throughput variations for small segment sizes would be to be more conservative in terms of requested download rate, i.e. to download a representation of the video at a significantly lower rate than available throughput on average in order to prevent video playout disruptions. Here conservative means selecting the video bit rate smaller than the averaged measured throughput, e.g., equal to the 1% percentile of the segment download rate seen up to the moment. As presented in the following in Table 1, the selected representations for such a percentile for the considered links with an average available throughput of around 2 Mbps, 4 Mbps and 6 Mbps are 860 Kbps, 1300 Kbps and 1800 Kbps, respectively, which is in line with the results presented in Ref. [31]. A possibility for downloading a representation at a higher bitrate would be to use SVC and download the layers of each segment sequentially and omit request to higher layers when a reduction in the throughput has been detected. HTTP pipelining [34] is combined with this technique to avoid idle states in the TCP connection and use the available throughput effectively. Thus, the aforementioned playout disruptions suffered with AVC, if not a low enough media rate is downloaded, would transform to quality adaptation events using SVC. Therefore, using SVC allows the user for downloading at higher average video rate, at the cost of suffering adaptation sporadically, which can be assumed to be far less annoying than video disruptions and thus may increase the quality of experience of the users. The rates selected for MR-Streaming and for the lowest OP of SVC-PMD for each of the available mentioned throughput values (packet loss $p=2\%$, 0.5% and 0.22%), correspond to the rate values that have a lower probability than 0.1% for a segment to be downloaded after its playout time, i.e. a playout interruption event occurs. These values are selected, since they are quite low and are thought not to be disturbing for the viewer of a video service. For SVC-PMD an additional enhancement layer is allowed to be requested when enough throughput is available. The downloadable rates of the enhancement layer, which will be requested by the SVC client in addition to the already downloaded media (lowest OP in the table) are 200, 366 and 400 Kbps These selected additional enhancement layer rates correspond to the rates the SVC clients are able to request more than 98% of the time. The ability of additionally requesting enhancement layers as shown in Table 1 increases the average downloaded rate and does not imply clients switching often. Note that higher rates could be selected at the cost of suffering quality adaptation more frequently.

As shown in Fig. 15, the segment download rate distributions for different values of average link throughput overlap for some values. Therefore, detecting the state of the link (packet loss or average throughput) and estimating future performance is a really difficult task, since segment download rate patterns measured over short periods of time can correspond to different link states and segment download rates observed in the near future can be very different depending on the real link state, which makes estimation and prediction for selecting the most appropriate representation a difficult task. That means that unless the receivers request a continuously varying video rate that is significantly lower than the last measured segment download rate, the clients may first notice a throughput variation, when it is already too late, i.e. a wrong bitrate is requested and a playout interruption occurs. Since the monitored segment download rate has a high variation as shown in Fig. 15, selecting the media download rate continuously varying depending on the last measured segment download rate would lead to a continuous adaptation, which may be annoying to some extent. Therefore, such an approach is not considered as a possible solution.

In Fig. 16 we show the available representations as well as the possibilities for requesting segments, comparing the sequential request of the layers for SVC-PMD and MR-Streaming. As seen in the figure, for SVC-PMD there is an additional time to decide whether to download higher layers or not during download of the lower layers, while for MR-Streaming the rate selection can be only done at one step. In order to compare the MR-Streaming and the SVC-PMD approach the probability of receiving one segment too late is calculated in the following. For this purpose we consider the scenario shown in Fig. 17, corresponding to a simplified 3-state channel model described by a Markov model.

As mentioned before, as a consequence of overlapping of the segment download rate distributions and the high variability of the download segment rate, due to which long measure times are required to detect variations in the available throughput, it can be assumed that a client would first notice a variation in the available throughput when a segment is received too late.

Hence, the probability of receiving a segment too late in the MR-Streaming case is the probability of suffering a throughput reduction, which can be calculated following Eq. (1), where p_4 and p_6 are the probabilities of having a channel with an available throughput equal to 4 an 6 Mbps, respectively, and can be obtained by calculating the steady state vector, i.e. eigenvector corresponding to the eigenvalue 1 of the matrix P

$$P_{interruption} = p_4 * p_{42} + p_6 * p_{62} + p_6 * p_{64} \quad (1)$$

For SVC, a playout interruption event is subject to additional conditions. Note that due to the working principle of SVC-PMD, if throughput reductions are detected while downloading lower layers, requests for higher layers are omitted, avoiding thus playout interruptions. Playout interruption would only occur, if the throughput reduction is not detected for low layers but during download of a certain higher layer. By late detection of throughput reduction, the requested layer download would last too long and it would not be possible to download the following segments on time, resulting in

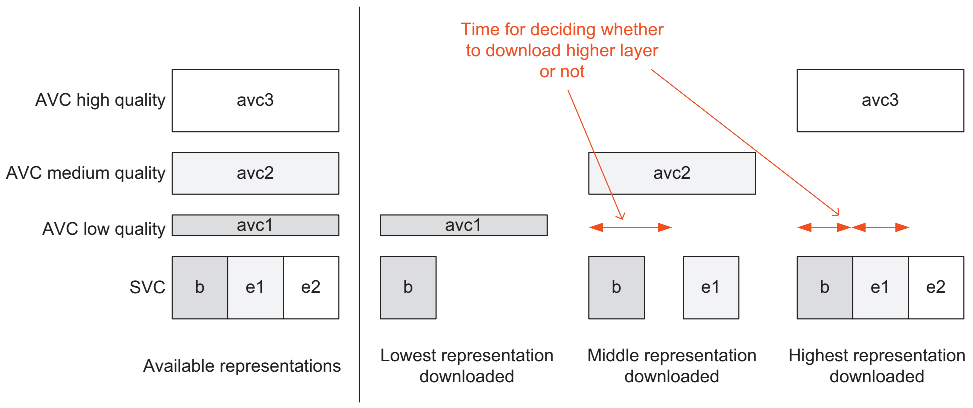
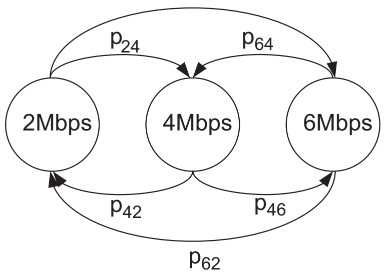

Time for deciding whether to download higher layer or not
AVC high quality
AVC medium quality
AVC low quality
SVC
avc3
avc2
avc1
b
e1
e2
Available representations
Lowest representation downloaded
Middle representation downloaded
Highest representation downloaded

Fig. 16. Available representations and download options.



$$= \begin{bmatrix} p_{22} & p_{24} & p_{26} \\ p_{42} & p_{44} & p_{46} \\ p_{62} & p_{64} & p_{66} \end{bmatrix}, with \quad p_{ii} = 1 - \sum_{\forall j \neq i} p_{ij}$$

Fig. 17. Model for channel throughput variation.

a playout interruption. Therefore, for the SVC case the addends of Eq. (1) are multiplied by a scaling factor before summing them up, as shown in Eq. (2)

$$P_{interruption} = k_{42}{*}p_4{*}p_{42} + k_{62}{*}p_6{*}p_{62} + k_{64}{*}p_6{*}p_{64} \qquad (2)$$

The scaling factor k_{xy} corresponds to the probability that the throughput reduction is detected too late at some higher layers. Layers are ordered from lower to higher ones, where the lowest layer is considered to be the base layer and layers depending on others are considered to be higher. Note that as mentioned in Section 7 rate reductions are related to TCP packet losses. Hence, k_{xy} can be calculated as the probability that no TCP packet from certain lower layers, in the following referred to as *L layers*, is lost and at least one packet in any layer higher than the *L layers* is lost. Which layers are in the set of *L layers* depends on the throughput reduction. The *L layers* consist of the maximum number of lower layers for which detection of throughput reduction would let the SVC-PMD technique to omit requests for higher layers and avoid playout interruption. Detection of throughput reduction for any layer higher than the *L layers* result in a playout interruption. The scaling factors k_{xy} for the selected scenario shown in Fig. 17 are summarized in the following:

- k_{42}=Probability that no packet from the base layer is lost and at least one packet in enhancement layer 1 (e1 in Fig. 16) is lost.
- k_{62}=Probability that no packet from the base layer is lost and at least a packet in enhancement layer 1 (e1 in Fig. 16) is lost+probability that no packet from the base layer and enhancement layer 1 is lost and at least a packet in enhancement layer 2 (e2 in Fig. 16) is lost.
- k_{64}=Probability that no packet from the base layer and enhancement layer 1 is lost and at least a packet in enhancement layer 2 (e2 in Fig. 16) is lost.

The probabilities described above can be calculated as shown in Eq. 3, as follows:

$$k = (1-p)^n{*}(1-(1-p)^N) \qquad (3)$$

where p is the packet loss probability of the new state, n is the number of packets corresponding to the layers where no packet is lost and N includes the amount of packets in the layer where the loss occurs and in the lower ones.

The Markov model considered for the presented results, can be summarized with the transition probabilities p_{ij}=0.005 for $i \neq j$ and p_{ii}=0.99 and probabilities in steady state p_i=0.33. With the rates for the considered in for the SVC stream and assuming a segment length of 0.5 s, the scaling factors for SVC-PMD are calculated following Eq. (3) as k_{42}=0.25, k_{62}=0.55 and k_{64}=0.35.

Taking these values into account the probability of suffering a playout interruption can be calculated as aforementioned, following Eqs. (1) and (2), and is shown in Table 2.

Table 2
Playout interruption probability.

	Playout interruption probability
MR-Streaming	0.5%
SVC-PMD	0.19%

9. Conclusions

HTTP-Streaming is increasingly gaining popularity due to its benefits over UDP, caused by the better traversal of NAT and firewalls, ease of deployment and built-in friendly bandwidth sharing. The Scalable Video Coding (SVC) can improve the performance of HTTP-based IPTV in a wide range of areas.

First, by only using SVC without any further system changes, the caching efficiency is significantly increased, since the storage capacity of the caches is more efficiently used than when different versions of the same video are stored with AVC.

Second, in VoD scenarios, there is a trade-off between service smoothness, in the sense that adaptation is rarely performed trying to play always the highest possible quality, and playout delay, during which a pre-buffering phase is carried out to playout-buffer data to overcome possible problems within the network during the streaming service. Especially for small values of playout-buffer SVC allows for downloading an average video rate significantly higher than with multi-rate AVC, due to better efficiency in performing the adaptation.

Finally, we show that SVC-based HTTP-Streaming outperforms AVC-based HTTP-Streaming for live IPTV. More concretely, by using SVC, receivers can request a representation that approximates more to the average TCP throughput than when AVC is considered, for which the receivers have to play a more conservative role and renounce to higher quality representations and download representations at lower bitrates in order to have a live service. Furthermore, the receivers can react much faster to network variations using SVC, which prevents the video from being disrupted when the traffic in the network unexpectedly increases.

References

[1] J. Postel, User Datagram Protocol (UDP) IETF RFC 768, August 1980, 〈http://tools.ietf.org/html/rfc768〉.

[2] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Hypertext Transfer Protocol—HTTP/1.1 IETF RFC 2616, June 1999, 〈http://www.ietf.org/rfc/rfc2616.txt〉.

[3] Transmission Control Protocol (TCP) IETF RFC 793, September 1981, 〈http://tools.ietf.org/html/rfc793〉.

[4] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real-Time Applications July 2003, 〈http://www.ietf.org/rfc/rfc3550.txt〉.

[5] Information Technology – Dynamic Adaptive Streaming Over HTTP (DASH) – Part 1, Media Presentation Description and Segment Formats, ISO/IEC DIS 23009-1, August 2011.

[6] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs (Release 9); 3GPP TS 26.234 V9.3.0 (2010-06), Section 12: Adaptive HTTP Streaming.

[7] T. Stockhammer, 3GPP dynamic adaptive streaming over HTTP – standards and design principles, in: Proceedings of Multimedia Systems Conference – Special Session Media Transport, San Jose, USA, February 23–25, 2011.
[8] Open IPTV Forum – Release 2 Specification, HTTP Adaptive Streaming, Draft V0.06 - June 7, 2010.
[9] Adobe Systems, Real Time Messaging Chunk Stream Protocol (RTMP) Specification, draft-rtmpcs-01.txt , June 2009.
[10] A. Zambelli, IIS smooth streaming technical overview, Microsoft Corporation (2009).
[11] HTTP Live Streaming Archictecture, Technical Report, Apple Inc., 2010.
[12] T. Wiegand, G.J. Sullivan, G Bjøntegaard, Ajay Luthra, Overview of the H.264/AVC Video Coding Standard, IEEE Transactions on Circuits and Systems for Video Technology 13 (7) (2003) 560–576.
[13] H. Schwarz, D. Marpe, T. Wiegand, Overview of the scalable video coding extension of the H.264/AVC standard, IEEE Transactions on Circuits and Systems for Video Technology 17 (9) (2007) 1103–1120.
[14] Y. Sánchez, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. De Vleeschauwer, W. Van Leekwijck, Y. Le Louedec, Improved caching for HTTP-based video on demand using scalable video coding, in: Proceedings of the Eighth Annual IEEE Consumer Communications and Networking Conference–Special Session IPTV and Multimedia CDN - (CCNC'2011 - SS IPTV), Las Vegas (NV), January 9–12, 2011.
[15] Y. Sánchez, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. De Vleeschauwer, W. Van Leekwijck, Y. Le Louedec, iDASH: improved dynamic adaptive streaming over HTTP using Scalable Video Coding, in: Proceedings of Multimedia Systems Conference–Special Session Media Transport, San Jose, USA, February 23–25, 2011.
[16] Information technology—Generic coding of moving pictures and associated audio information: Systems. ISO/IEC 13818-1, 2007.
[17] Information technology – Coding of audio-visual objects – Part 12: ISO base media File Format. ISO/IEC 14496-12, 2008.
[18] T. Schierl, K. Grüneberg, T. Wiegand, Scalable Video Coding over RTP and MPEG-2 Transport Stream in Broadcast and IPTV Channels, IEEE Wireless Communications Magazine 16 (5) (2009) 64–71.
[19] Information technology – Coding of audio-visual objects – Part 15: Advanced Video Coding (AVC) file format. ISO/IEC 14496-15, 2010.
[20] P. Amon, T. Rathgen, D. Singer, File format for Scalable Video Coding (SVC), IEEE Transactions on Circuits and Systems for Video Technology, SI on SVC 17 (9) (2007) 1174–1185.
[21] Information technology – Coding of audio-visual objects – Part 12: ISO base media File Format ISO/IEC 14496-12:2008/PDAM3, 2011.
[22] S. Tullimas, T. Nguzen, R. Edgecomb, S. Cheung, Multimedia Streaming using multiple TCP connections, ACM Multimedia Computing, Communications, and Applications 4 (2008).
[23] R. Kuschnig, I. Kofler, H. Hellwagner, Improving internet video streaming performance by parallel TCP-based request-response streams, in: Proceedings of the Seventh Annual IEEE Consumer Communications and Networking Conference (CCNC'2010), January, 2010.
[24] K. Evensen, D. Kaspar, C. Gridwodz, P. Halvorsen, A. F. Hansen, P. Engelstad, Improving the performance of quality-adaptive video streaming over multiple heterogeneous access networks, in: Proceedings of Multimedia Systems Conference–Special Session Media Transport, San Jose, USA, February 23–25, 2011.
[25] Y. Schierl, R. Sanchez, C. Globisch, Hellge, T. Wiegand, Priority-based Media Delivery using SVC with RTP and HTTP streaming, Springer Multimedia Tools and Application (2010) September.
[26] H. Schwarz, T. Wiegand, Further results for an rd-optimized multi-loop SVC encoder, JVT-W071, JVT Meeting San Jose, USA, 2007, ⟨ftp://avguest@ftp3.itu.int/jvt-site/2007_04_SanJose/JVT-W071.zip⟩.
[27] S. Podlipnig, L. Böszörményi, A survey of Web cache replacement strategies, ACM Computing Surveys 35 (2003) (2003) 331–373.
[28] D. Hong, D. De Vleeschauwer, F. Baccelli, A chunk-based caching algorithm for streaming video, in: Proceedings of the Fourth Workshop on Network Control and Optimization, Ghent (Belgium), November 29–December 1, 2010.
[29] H. Fall, S. Floyd, Simulation-based comparison of Tahoe, Reno and SACK TCP, ACM SIGCOMM Computer Communication Review 26 (1996) July.
[30] TCP/IP Illustrated, Vol. 1 The Protocols, 10th ed. Reading, MA: Addison-Wesley, 1997.
[31] B. Wang, J. Kurose, P. Shenoy, D. Towsley, Multimedia streaming via TCP: an analytic performance study, in: Proceedings of the 12th Annual ACM International Conference on Multimedia, New York, USA, October 10–16, 2004.
[32] The Network Simulator NS-2, ⟨http://www.isi.edu/nsnam/ns⟩.
[33] J. Padhye, V. Firoiu, D.F. Towsley, J.F. Kurose, Modeling TCP Reno performance: a simple model and its empirical validation, IEEE/ACM Transactions on Networking 8 (2000) 133–145.
[34] D. Kaspar, K. Evensen, P. Engelstad, A.F. Hansen, Using HTTP pipelining to improve progressive download over multiple heterogeneous interfaces, in: Proceedings of International Conference on Communications (ICC), Cape Town, South Africa, 2010.