Implications of the ISO Base Media File Format on Adaptive HTTP Streaming of H.264/SVC

Ingo Kofler, Robert Kuschnig, Hermann Hellwagner
Institute of Information Technology (ITEC)
Alpen-Adria-Universität Klagenfurt
Klagenfurt, Austria
Email: firstname.lastname@itec.uni-klu.ac.at

Abstract—**HTTP streaming has gained significant attraction in the last few years. Currently many commercial as well as standardized streaming systems are already offering adaptive streaming. In most cases, the adaptation is achieved by switching between separately encoded video streams in different qualities. In contrast to that, this paper focuses on the applicability of scalable video coding based on the H.264/SVC standard for adaptive HTTP streaming. Recent work has already highlighted the conceptual advantages like better cache utilization, fine-grained bit rate scalability, and lower storage requirements. This paper discusses the actual realization and design options for implementing priority streaming using the ISO Base Media File Format (BMFF). We propose three different strategies for organizing the scalable video bit stream that consider both the possibilities as well as limitations of the ISO BMFF. The proposed strategies are discussed and evaluated both conceptually and quantitatively. For that purpose, we provide a detailed analysis based on modeling both the overhead of the file format and the HTTP encapsulation. The results for all three priority streaming strategies show that the limitations of the ISO BMFF result in a high relative overhead in the case of low bit rate content. However, when applied to high quality content, priority streaming of H.264/SVC can be implemented at a very low cost. Depending on the number of layers and the offered scalability dimensions, different strategies should be chosen to minimize the overhead. Based on the analytical model and the discussion, this paper provides guidance for selecting the most efficient strategy.**

I. Introduction

HTTP video streaming is a topical issue in the field of multimedia communication [1]. Although streaming based on HTTP was considered inferior by the scientific community in the past, it has been established as the predominant solution for realizing VoD and live streaming services in the Internet. The reason for its success is that it provides a simple solution for current deployment issues of traditional UDP-based streaming approaches, e.g., middlebox traversal, congestion control, and scalability. At present, several commercial vendors offer HTTP adaptive video streaming systems that are either based on proprietary [2], [3] or standardized formats [4], [5]. Additionally, both 3GPP and MPEG have standardized an HTTP streaming system entitled Dynamic Adaptive Streaming over HTTP (DASH) [6].

Most of the approaches have in common that the video content is not fetched in a single request but is rather split up in smaller units in the order of several seconds of media playtime. Depending on the system these units are typically called segments, fragments or streamlets. Although this fragmentation introduces some overhead, it comes along with a variety of advantages. First, the units can be fetched using multiple connections and/or from multiple servers. Second, seeking to a certain point in the video can be realized very easily. Finally, the fragmentation allows switching between different qualities and/or resolutions at the units' boundaries. It is then up to the client to decide which quality should be downloaded in order to achieve the best video quality for the current networking conditions. This behavior is not normative, but it is expected that the client intelligently selects the proper unit based on bandwidth estimation or similar heuristics. However, once the download of a unit has been started, switching to a higher or lower quality is not possible.

As shown in related work [7], the use of the scalable video coding (SVC) extension of H.264/AVC offers potential advantages like fine grained bit rate scalability and reduced storage requirement at the server and in caches. More importantly, the layered organization of the scalable bit stream allows to perform priority streaming [8]. This means that the most important part of the bit stream – the base layer – is transmitted first, followed by enhancement layers that refine the video quality gradually [9]. While we already investigated the combination of priority streaming and H.264/SVC in our previous work [10], this paper discusses how to realize priority streaming using HTTP streaming based on the prevalent ISO Base Media File Format (BMFF). We highlight some shortcomings of the ISO BMFF related to H.264/SVC and discuss the impact on the system performance and overhead. For that purpose we introduce three different standard-compliant strategies and analyze the overhead of the bit stream organization and the HTTP encapsulation.

II. Background

The scalable extension to H.264/AVC was standardized in 2007 as an amendment to the successful H.264/AVC video compression standard. It provides scalability features for the video bit stream in a backward-compatible way at the cost of a relatively low bit rate overhead. The bit stream is organized in Network Abstraction Layer units (NAL units) which either contain encoded video data or additional information (e.g., for decoder initialization). Scalability is offered in the spatial, temporal, and SNR dimensions. In order to achieve backward-compatibility, the bit stream contains a base layer which is fully H.264/AVC compliant. It contains only H.264/AVC NAL units and can be decoded by any legacy decoder. The scalability is achieved by introducing new NAL units that gradually refine the quality of the base layer. The adaptation of the video can be performed by simply discarding NAL units that do not belong to a certain layer representation. The layer information is contained in the extension header of the scalable NAL units. For H.264/AVC-only NAL units this information is signaled by a so called prefix NAL unit to ensure backward-compatibility. All the NAL units that are used for reconstructing a single sampling instance in time, i.e., a picture, are referred to as *access unit*. An important aspect of the bit stream organization is that all NAL units of the access unit have to be stored sequentially in decoding order.

In the context of adaptive video streaming, we consider the SNR scalability as the most appropriate tool to adapt the video quality according to the network conditions. In H.264/SVC two different mechanisms for SNR scalability are available. The *coarse-grain quality scalable coding* (CGS) allows to encode a few different quality levels of the video in a single bit stream. For that purpose, similar mechanisms as for the spatial scalability are employed. The disadvantage is poor encoding performance when the bit rate difference between adjacent layers becomes small [9]. On the contrary, the *medium-grain quality scalability coding* (MGS) is used for encoding video streams that should offer a larger variety of different bit rates. It can be considered as a variant of CGS but offers a more flexible way to remove certain NAL units of the quality enhancement layer to trade off visual quality and the resulting bit rate. This enables a very accurate control over the bit rate since a single enhancement layer can be used to extract a multitude of different quality variations. Besides, it implies the possibility to switch the video bit rate at every access unit. When using CGS switching is only possible at certain access units.

III. ISO Base Media File Format

Since HTTP only defines the transport mechanism for fetching files or parts thereof, it is required to use a multiplexing format for the multimedia content. In this paper we focus on the ISO Base Media File Format (BMFF), that is used both by proprietary solutions like Microsoft's Smooth Streaming and in current standards [11]. Alternatives to the ISO BMFF are the MPEG-2 Transport Stream, which is however known to be inefficient for low bit rate streams [12], and proprietary formats like [3], which we do not consider in our investigations.

The Base Media File Format organizes the multimedia data and its meta data in data structures called *boxes*. A box consists of a length field, its type and its payload. The structure of the payload depends on the type of the box. Boxes can be organized sequentially or hierarchically. The composition rules of the boxes are specified in the corresponding standards and are identified by a file brand. A file brand also defines which boxes are considered mandatory or optional. The standard can be tailored for different purposes by defining a new file brand

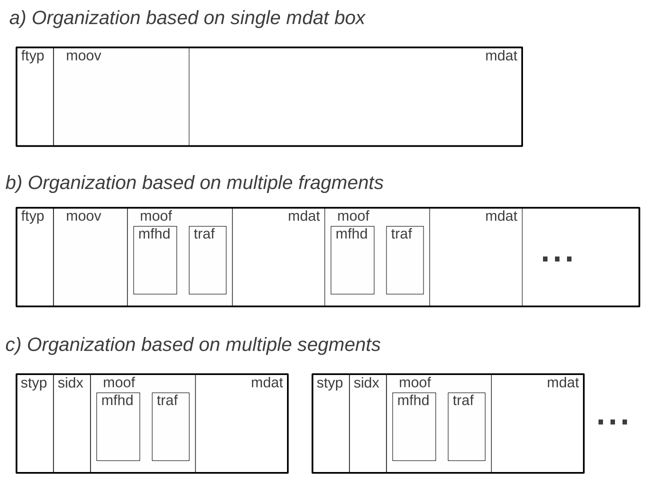
a) Organization based on single mdat box
ftyp
moov
mdat
b) Organization based on multiple fragments
ftyp
moov
moof
mfhd
traf
mdat
moof
mfhd
traf
mdat
...
c) Organization based on multiple segments
styp
sidx
moof
mfhd
traf
mdat
styp
sidx
moof
mfhd
traf
mdat
...

Fig. 1. Different ISO BMFF file organization strategies

that uses only a subset of the boxes offered by the standard. The file brand that a file is based upon is signaled at its beginning in an *ftyp* box.

The internal file organization differs significantly when using the ISO BMFF for storage only or when using it for streaming purposes. Figure 1 illustrates the different file organization methods discussed in the following. If streaming is not an issue, a file is typically separated into a kind of file header that provides means for indexing and seeking – the *moov* box – and the actual media samples in the *mdat* box. However, this organization is neither suitable for the purpose of streaming nor for incremental generation of the content at the server. Consequently, multiplexing of the indexing information and the media samples is a requirement for streaming applications. This concept is called *fragmentation* and allows for incremental generation of the media file as well as progressive download capabilities [13]. Still, the media file contains a kind of file header – the movie (*moov*) box. However, this box contains rather decoder initialization information and metadata that is valid for the whole file. The rest of the file consists of an alternating sequence of movie fragment (*moof*) and *mdat* boxes. The *moof* box contains the meta data and seeking information for a single media fragment, while the *mdat* box holds the fragment's media samples (e.g., audio or video frames). In the context of adaptive HTTP streaming, a related mechanism called *segmentation* was introduced by 3GPP [11]. It allows to split the media content into self-contained files called segments. These segments can be identified by their own URLs and can be downloaded separately. Besides, having segments that contain only parts of the content is also considered advantageous for HTTP caches. Each segment typically consists of a segment type box (*styp*), which acts as a kind of file header, an optional segment index box (*sidx*) and one or multiple fragments. Again, each fragment consists of a *moof* and a *mdat* box.

IV. H.264/SVC Priority Streaming

Adaptive HTTP streaming based on non-scalable video codecs typically works as follows. The video content is split into a sequence of segments/fragments, which are offered at different bit rates resulting in different quality levels. As an aside, these different levels are called *representations* in the terminology of the upcoming DASH standard. During streaming, the client must decide a priori which representation of a segment to download next. This decision typically incorporates a network bandwidth estimation and the consideration of the client buffer occupancy level in some heuristics. However, once this decision has been made and downloading a certain representation of a segment has been started, switching to a different representation is not possible. This means that if the decision was wrong, e.g., because the network bandwidth decreased during the download of the segment, it is no longer possible to switch to a representation at a lower bit rate. But also in the opposite case, when the bandwidth increases during the download and would allow a higher quality representation, the client is bound to its initial decision.

In this context, the use of H.264/SVC offers potential advantages [7]. The layered organization of the scalable bit stream allows to perform priority streaming. This means that the most important part of the bit stream – the base layer – is transmitted first, followed by enhancement layers that refine the video quality gradually. Obviously, the sequence of transmission is in descending priority order. Instead of deciding on a certain representation, this allows a simpler implementation where the client receives as much of the bit stream as possible in a given time interval. The more data of the enhancement layers the client retrieves, the better is the final video quality. Due to the incremental transmission, this approach can also react to bandwidth fluctuations during the download of a single segment. Besides, as shown in [10] it does not solely rely on a bandwidth estimation by the client to decide which representation to download.

While we already investigated the combination of priority streaming and H.264/SVC in our previous work [10], we now focus on how to realize this mechanism using the ISO BMFF. Unfortunately, the extensions to the BMFF for H.264/SVC do not allow to arrange the NAL units of the different layers arbitrarily in the *mdat* box. Instead, all the NAL units of the base layer and the enhancement layer that belong to an encoded video frame have to be stored sequentially in the file. Nevertheless, we identified three standard-compliant strategies how to realize priority streaming under these constraints. These strategies have in common that they do not require special support by the server, i.e., an ordinary HTTP server can be used for streaming the content. The strategies only differ in how the content is stored in the ISO BMFF files and how the client issues HTTP requests to fetch the content. As also the upcoming DASH standard makes use of the ISO BMFF, it is possible but not necessarily required to use the strategies also in the context of DASH.

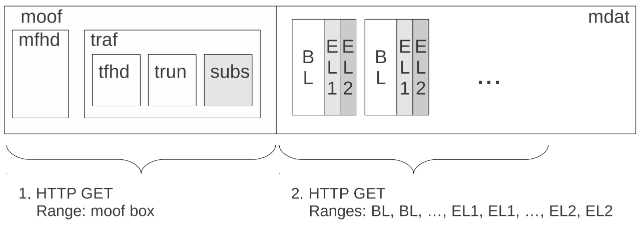

Fig. 2. Priority streaming of a single fragment based on the *subs* box

subs-box strategy: The first strategy is to use a single, fragmented media file and to add a *sub-sample information box* (*subs*) [13] to each fragment. This box acts as a fine-grained table of content of the fragment's *mdat* box. It allows for describing each sample (i.e., video frame) on a sub-sample basis. In the ISO BMFF terminology, a sub-sample is typically a single NAL unit[1], which might belong to the base or any enhancement layer. Consequently, the *subs* box allows to infer the position and size of each NAL unit in the fragment's *mdat* box as well as its layer information. Based on this information, the client can request the sub-samples (NAL units) in priority order by issuing an HTTP request specifying a list of byte ranges. In its response, the server would then transmit the NAL units in descending priority order. This implies that the client has to initially download the fragment header which contains this *subs* box. This strategy of file organization and HTTP download is illustrated in Figure 2.

While this approach can be ideally combined with medium-grained scalability, it comes also with some drawbacks. First, the *subs* box introduces an additional overhead which depends both on the size of the fragment (the number of samples) as well as the number of enhancement layers (number of sub-samples per sample). Second, specifying multiple byte ranges in a single HTTP request causes the server to organize its response as multipart message [14]. This mechanism also introduces an overhead for each sub-sample (i.e., NAL unit) delivered by the server. An exact analysis of the overhead will be provided in the next section.

Multi-track strategy: The second strategy is also based on a fragmented media file but the scalable video stream is split into multiple tracks. Typically, tracks are used to represent different audio and video streams within a media file. However, in the case of scalable video a single video stream can also be represented by multiple tracks, where each track represents a single layer. This mechanism was extensively discussed in [15] but in a different context. There, the authors demonstrated how multiple tracks can be used for erosion storage support. This track-based approach allows to store all NAL units of a single layer sequentially in a track. This allows to fetch all NAL units of the fragment that belong to the same layer (= track) in a single HTTP byte range request.

[1]Note that in certain cases it is possible to aggregate NAL units into a single sub-sample, e.g., if they are belonging to the same layer like the prefix NAL unit and its associated AVC NAL unit.

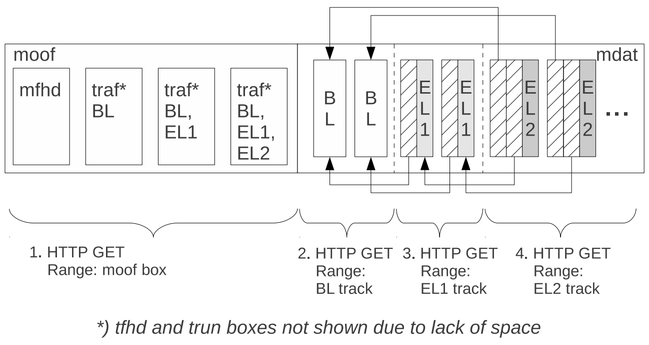
moof
mfhd
traf* BL
traf* BL, EL1
traf* BL, EL1, EL2
mdat
BL
BL
EL1
EL1
EL2
EL2
...
1. HTTP GET Range: moof box
2. HTTP GET Range: BL track
3. HTTP GET Range: EL1 track
4. HTTP GET Range: EL2 track
*) tfhd and trun boxes not shown due to lack of space

Fig. 3. Priority streaming of a single fragment using multiple tracks

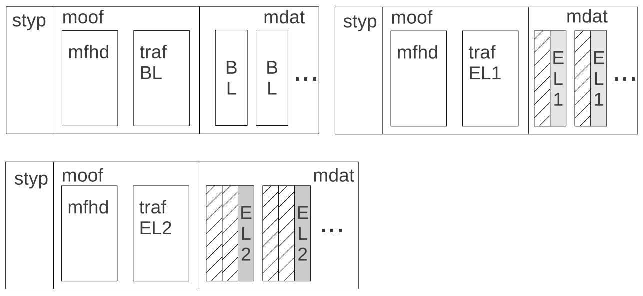

Fig. 4. Priority streaming using multiple segments

Decoding dependencies exist among the tracks of the scalable video since higher layers can only be decoded based on lower layers. The references to dependent layers are realized by so called *extractors*, which represent a special type of NAL units. The extractors are used to reference NAL units of the base layer or other layers used for prediction in order not to break the standard's requirement to store all NAL units of a sample sequentially in the file. The organization of a single fragment following this strategy is illustrated in Figure 3. The extractor NAL units are represented by the hatched boxes in this figure.

Compared to the subs-box strategy, this approach avoids specifying a large number of byte ranges in the HTTP request. Instead, each layer can be addressed by a single byte range which reduces the overhead of the multipart response. While the overhead of the HTTP delivery is lower, the overhead of the ISO BMFF container increases. First, each track that contains a layer results in a track fragment (*traf*) box in the *moof* box of the fragment. Second, the extractor NAL units that are required for reference purposes also require additional storage in the *mdat* box. Both requirements obviously limit the number of quality levels that can be described efficiently. Consequently, it is not well suited for medium-grained scalability where the bit rate is controlled by selectively discarding NAL units of a certain layer. This would imply that the video is organized in up to 64 different tracks which is obviously not a reasonable option.

Multi-segment strategy: The third strategy is similar to the proposed approach that uses multiple tracks. But instead of storing the tracks that contain the different layers in a single fragment, each track is stored in its own segment (i.e., file) as shown in Figure 4. Consequently, we denote this approach strategy as multi-segment strategy. This approach is inspired by the emerging DASH standard, which allows to split scalable content into multiple segments and provides means for signaling their interdependency. Storing the layers in different segments allows an easier and separate caching of the distinct layers by HTTP caches which can be very beneficial in certain scenarios [7]. Besides, using byte ranges in the HTTP request is not necessary, since the whole files are fetched at once. Compared to the second approach this also reduces the number of HTTP requests by one, as the first request that fetches the *moof* box only is not necessary. The drawback of this approach is the increased overhead by the ISO BMFF because each segment will contain an *styp* box to make it self describing and also the *moof* box and the relevant *mfhd* box must be part of each segment.

V. Overhead Analysis

In order to analyze the overhead of the three strategies, we model both the size of the required boxes of the ISO BMFF and the overhead introduced by the HTTP responses. This overhead is then set into relation to the media bit rate to study the relative overhead imposed by each strategy. We varied the number of layers contained in the bit stream and investigated the resulting overhead relative to the A/V bit rate. The file header common to all of the three strategies including the *moov* box is neglected in this model. Its size is typical around a few KiB and can be considered as insignificant when streaming a video with a reasonable playback duration. The file organization was modeled as described above but considers an additional non-scalable AAC audio track. We consider this as more representative than just study the overhead of video-only transmission. As the overhead of the fragmentation depends on the fragment duration, we assume a rather conservative fragment duration of 4 seconds for our model. While the majority of the boxes have a fixed size, the size of the *trun* box depends on the number of samples it describes. The *trun* box is an index that allows to extract the samples and their timestamps from the *mdat* box. For our analysis, we assume a 30 fps video content and an audio sampling rate of 44.1 kHz to determine the size of the *trun* box. To model the overhead of the HTTP protocol, we assume that each HTTP request results in a response with a response header size of 280 bytes. This value was derived by observing the response of a typical HTTP server. In case of multipart HTTP responses, the overhead for fetching multiple byte ranges with a single request is significantly higher. Based on our observations we assume an average overhead of 92 bytes for each byte range requested.

The results of the overhead analysis for A/V bit rates below 1 Mbps is given in Figure 5. The graphs indicate that the overhead of all three strategies is very high when considering 5 video layers. The overhead of the subs-box strategy is far

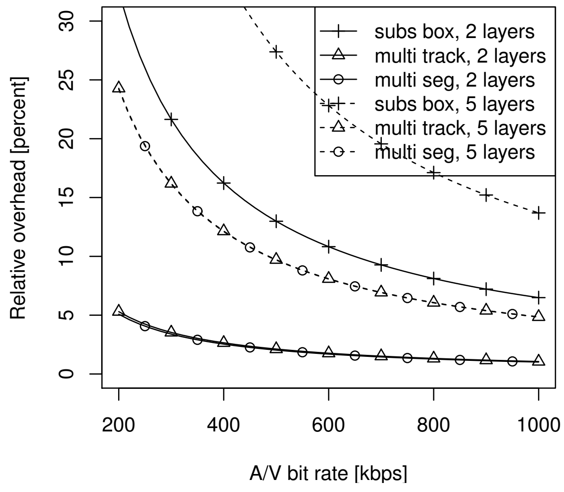

Fig. 5. Overhead analysis for low bit rate content

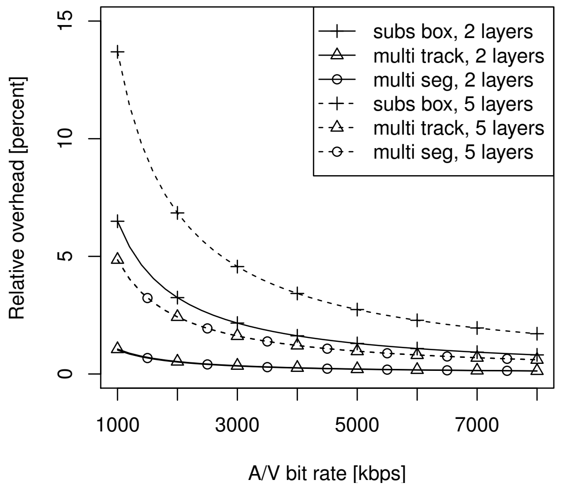

Fig. 6. Overhead analysis for high bit rate content

from being reasonable at these low bit rates. Surprisingly, the multi-track and multi-segment strategies perform almost equally well for both 2 and 5 layers. It turned out that in the case of multiple segments the larger overhead of the ISO BMFF is compensated by the lower HTTP overhead. Although these two strategies perform better than the first one, the overhead is still more than 5 percent. When considering this container overhead and the additional overhead of about 10 percent imposed by the scalable video coding, it is questionable if H.264/SVC can compete with stream-switching approaches based on traditional video coding at these bit rates.

As one can see from Figure 6, the proposed strategies are more competitive when applied to high definition content. In this case, the overhead of the ISO BMFF and the HTTP response header can be compensated by video bit rates on the order of several Mbps. The multi-track and multi-segment strategies lead to a relative overhead lower than 2 percent for bit rates higher than 2.5 Mbps. This renders the approaches well suited when providing two or three spatial resolutions combined with coarse-grained quality scalability. The analysis also suggests that the subs-box strategy can be used advantageously for medium-grained scalability. Its flexible access mechanism by fetching each NAL unit based on a separate byte range allows to achieve a reasonable bit rate scalability. Using a base layer and a single MGS enhancement layer up to 64 quality degradations (and bit rate steps) can be realized at a competitive overhead. Compared to adaptive streaming that uses traditional video coding this strategy combines higher adaptivity and a lower storage requirement.

Finally, one can conclude that the subs-based strategy is recommendable when using medium-grained scalability for an HD video offered at a single spatial resolution. The multi-track and multi-segment strategies are better suited when using different spatial resolutions and/or coarse-grained scalability.

VI. Conclusions

In this paper, we studied the applicability of H.264/SVC priority streaming based on the ISO BMFF. The main problem with the ISO BMFF is the inflexible arrangement of the enhancement layers in the media file. We discuss three different strategies to work around this problem. The proposed strategies either lead to a high overhead due to HTTP byte range requests or to inefficient structures in the file organization. Unfortunately, these shortcomings render priority streaming based on the BMFF as too inefficient for bit rates lower than 1 Mbps. Consequently, it can be only considered as a beneficial option for high-definition content.

References

[1] A. Begen et al., “Watching video over the Web, part II: Applications, standardization and open issues,” *IEEE Internet Computing*, 2010.

[2] “Akamai HD Network homepage,” http://www.akamai.com/hdnetwork.

[3] “Move Networks HD streaming homepage,” http://www.movenetworks.com/.

[4] A. Zambelli, “IIS Smooth Streaming technical overview,” Microsoft Corporation, Tech. Rep., March 2009.

[5] R. Pantos et al., “HTTP live streaming,” Internet Engineering Task Force, Internet Draft draft-pantos-http-live-streaming-04, Jun. 2010.

[6] T. Stockhammer, “Dynamic adaptive streaming over HTTP: Standards and design principles,” in *Proceedings of the ACM MMSys 2011*, Feb. 2011, pp. 133–144.

[7] Y. Sánchez et al., “iDASH: Improved dynamic adaptive streaming over HTTP using scalable video coding,” in *Proceedings of the ACM MMSys 2011*, Feb. 2011, pp. 257–264.

[8] C. Krasic et al., “Quality-adaptive media streaming by priority drop,” in *Proceedings of the NOSSDAV’03*. New York, NY, USA: ACM, 2003, pp. 112–121.

[9] H. Schwarz et al. , “Overview of the scalable video coding extension of the H.264/AVC standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

[10] R. Kuschnig et al., “An evaluation of TCP-based rate-control algorithms for adaptive Internet streaming of H.264/SVC,” in *Proceedings of the ACM MMSys 2010*, 2010, pp. 157–168.

[11] “3GPP TS 26.244, 3GPP file format (3GP),” 2010.

[12] H. Riiser et al., “Low overhead container format for adaptive streaming,” in *Proceedings of the ACM MMSys 2010*, Feb. 2010, pp. 193–198.

[13] “ISO/IEC 14496-15:2010, Information technology – Coding of audio-visual objects – Part 12: ISO base media file format,” 2010.

[14] R. Fielding et al., “Hypertext Transfer Protocol – HTTP/1.1,” Internet Engineering Task Force, RFC 2616, Jun. 1999.

[15] P. Amon et al., “File format for scalable video coding,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1174–1185, Sep. 2007.