A VLC Media Player Plugin enabling Dynamic Adaptive Streaming over HTTP

Christopher Müller and Christian Timmerer
Alpen-Adria-Universität Klagenfurt, Multimedia Communication
Universitätsstraße 65-67
9020 Klagenfurt am Wörthersee, Austria
+43 (0)463 2700 3600
{*firstname.lastname*}@itec.aau.at

ABSTRACT

This paper describes the implementation of a VLC media player plugin enabling Dynamic Adaptive Streaming over HTTP (DASH). DASH is an emerging ISO/IEC MPEG and 3GPP standard for HTTP streaming. It aims to standardize formats enabling segmented progressive download by exploiting existing Internet infrastructure as such. Our implementation of these formats as described in this paper is based on the well-known VLC. Hence, it is fully integrated into the VLC structure and has been also submitted to the VLC development team for consideration in future releases of VLC. Therefore, it is licensed under the GNU Lesser General Public License (LGPL). The plugin provides a very flexible structure that could be easily extended with respect to different adaptation logics or profiles of the DASH standard. As a consequence, the plugin enables the integration of a variety of adaptation logics and comparison thereof, making it attractive for the research community.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information System**]: Video

General Terms

Documentation, Design, Standardization.

Keywords

Video, HTTP Streaming, DASH, Dynamic Adaptive Streaming over HTTP, MPEG, 3GPP.

1. INTRODUCTION

Video streaming is nowadays more and more provided over-the-top (OTT) using existing Internet infrastructures. This means that service providers do not build their own dedicated networks, they just use the Internet as it is. However, the Internet is not designed for real-time streaming, it is designed for best-effort delivery of files. Caches, proxies, and content distribution networks (CDN) are part of this infrastructure which support file transfer very well.

Existing streaming technologies like the real-time streaming protocol (RTP) could not take any advantage of this infrastructure due to their design and they also have other disadvantages, e.g., passing firewalls or network address translation (NAT) traversal. Bitrate variances are also very difficult to handle with RTP unless a video codec is used that supports a scalable transmission like Scalable Video Coding (SVC) [1]. RTP is a push protocol, i.e., the server pushes the data to the client. For that reason RTP is typically state full and the server has to keep track of all clients and the state of each session. An alternative way of streaming that is currently in heavy use on the Internet is called progressive download. As the name says it is not a "true" streaming but it is a kind of download. In comparison to RTP which is based on user datagram protocol (UDP) the progressive download is based on the hyper text transfer protocol (HTTP). Consequently, the session is stateless and the client pulls the data from the server. This has several advantages such as fully exploitation of existing Internet infrastructure (e.g., caches, proxies, CDNs) and no issues with firewalls or NAT traversals as the media data is encapsulated within HTTP. However, this approach has also some disadvantages. The HTTP protocol adds a significant overhead to the transmission which is approximately twice the media bitrate [2].

Some industry consortia try to address the drawback on how to handle varying bitrates. One of the first solutions has been specified within 3GPP as Adaptive HTTP Streaming (AHS) [3]. The basic idea is to chop the media file into segments which can be encoded at different bitrates or resolutions. The segments are provided on a Web server and can be downloaded through standard HTTP GET requests. The adaptation to the bitrate, resolution, etc. is done on the client side for each segment, e.g., the client can switch to a higher bitrate – if bandwidth permits – on a per segment basis. This has several advantages because the client knows its capabilities and its received throughput best. In order to describe the relationship between bitrates, segments, and the order of the segments AHS introduces the so-called Media Presentation Description (MPD). The MPD is a XML file that represents the different bitrates and HTTP uniform resource locators (URLs) of each individual segment. This structure

*Area Chair: Pablo Cesar, Wei-Tsang Ooi.

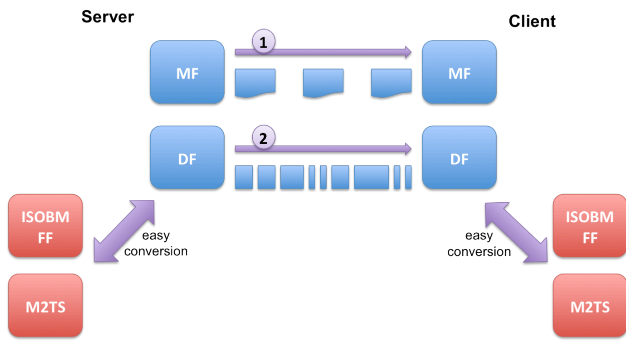

Figure 1. DASH System Architecture [7].

provides the binding of the segments to the bitrate (resolution, etc.) among others (e.g., start time, duration of segments). For example, a typical client would first request the MPD and with the information of the MPD it would then request the individual segments that fit best for its given bandwidth. If the bandwidth changes during the session the client could easily select a segment from another representation with a different bitrate that fulfills its bandwidth requirements best. This makes the progressive download method adaptive and dynamic in the same way. There are also other proprietary solutions from different companies like Microsoft's Smooth Streaming [4], Adobe's Dynamic HTTP Streaming [5] and Apple's HTTP Live Streaming [6] which more or less adopt the same approach. At the beginning of the session the client has to download a file that describes the URLs of the segments and the relation between the segments and a bitrate. After that the client could start with the download of the segments and eventually adapt dynamically to bandwidth fluctuations. The Dynamic Adaptive Streaming over HTTP (DASH) standard aims at combining the features of these proprietary solutions. The next section will give an overview of this standard.

2. Dynamic Adaptive Streaming over HTTP

During its 93rd meeting, MPEG evaluated 15 submissions from 20 organizations (including companies, research institutions and universities). The submissions provided technologies for the HTTP streaming of MPEG media in the following areas:

- Manifest File (MF), i.e., playlist, media presentation description, etc. which is mostly based on XML.
- Delivery Format (DF) as extensions/specializations of ISO Base Media File format (ISOBMFF) and MPEG-2 Transport Stream (TS).

The system architecture is depicted in Figure 1 [7] and based on that MPEG started a new work item called Dynamic Adaptive Streaming over HTTP (DASH).

On request, the manifest file will be provided to the client in order to initiate the session (cf. step-1 in Figure 1). The client will parse the manifest file and request individual segments compliant to the delivery format using HTTP and according to the information found in the manifest file (cf. step-2 in Figure 1). For the manifest file, DASH adopted the Media Presentation Description (MPD) as defined by 3GPP AHS [3] as a starting point. The MPD follows a data model comprising a sequence of one or more consecutive non-overlapping periods for which one or more representations may be available. A single representation refers to a specific media following certain characteristics such as bitrate, framerate, resolution, etc. Furthermore, each representation consists of one

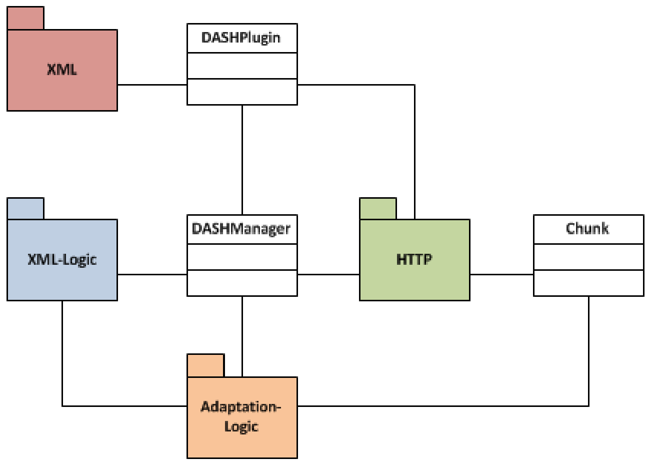

Figure 2. High-Level DASH Plugin Architecture.

or more segments that actually describe the media and/or metadata to decode and present the included media content.

The delivery format defines the format of the segments to be delivered to the client upon the HTTP requests based on the MPD. Finally, as the delivery format shall be compatible to existing MPEG formats (i.e. ISOBMFF and MPEG-2 TS), it shall be also possible to provide easy conversion from and to these formats. For example, easy conversion on the server would ease the usage of legacy content encoded in existing formats such as MP4 and its derivations. On the other hand, client-side easy conversion would facilitate repurposing of content received via DASH, e.g., in order to support legacy (decoding) infrastructures.

3. IMPLEMENTATION

This section provides information about the architecture of VLC media player and the DASH plugin which is fully integrated into VLC.

3.1 VLC Media Player

The VLC media player has a very clear and well defined layered architecture. Each layer is responsible for a concrete task and it could contain an undefined number of modules. Modules are representing a common action. The correct module for a given session will be chosen at runtime for each individual layer. This makes VLC very flexible. In the first version of the plugin it was located at the access layer because it is a kind of access. But at the newest version it has been moved one layer below because it is then independent from the access method. Due to the flexible structure of VLC and the reason that every module follows nearly the same structure this was not such a huge issue.

3.2 DASH Plugin

The DASH plugin is located at the stream filter layer of VLC and consists of four major components and two controller classes. Each of these components is depicted in Figure 2. The first component that is used at the startup of a DASH session is the *XML component* which is responsible for the XML parsing of the MPD and providing support to the other components. This component does neither provide means for the interpretation of the data nor the (adaptation) logic. The *HTTP component* handles all HTTP connections that have to be opened or closed during a DASH session. The current version of the plugin opens a HTTP connection for each individual segment. This will be improved in a future version of the plugin that will use persistent HTTP

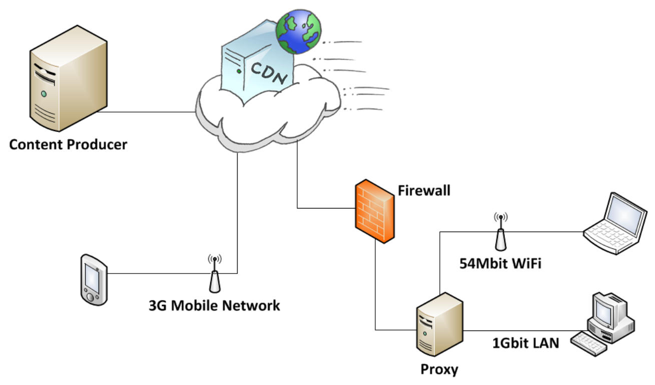

Figure 3. Example of DASH Deployment Architecture.

connections to reduce the HTTP overhead. The *XML-Logic component* adds the logic to the data representation that is provided though the XML component. As a consequence data and logic of the MPD are well separated. The plugin only supports a subset of the current DASH standard [8]. This is possible because the DASH standard defines profiles that describe such a subset. Currently the plugin only works with the *BasicCM* profile which is defined at [9]. Due to the structure it could easily be extended to support other profiles or even the full standard. The *Adaptation-Logic component* is responsible for the adaptation to the user preferences or device capabilities like bandwidth and resolution. The details of this component is also very flexible and is based on the well-known strategy pattern [10]. Thus, it is very easy to change the adaptation logic without affecting the other components. As a consequence, the plugin enables the integration of a variety of adaptation logics and comparison thereof, making it attractive for the research community.

4. DEPLOYMENT

Figure 3 shows an example of a DASH deployment. The three main components are the content producer the network and the clients. The content producer is responsible for the content production. This means that it provides the segments and the MPD to the CDN. Due to the flexible structure of DASH the content production could easily adopt to the needs of the CDN w.r.t. segment duration. Also legacy content could be reused, e.g., if there is a movie available in different bitrates, DASH could reuse these files because of the flexibility of the MPD. The content production could be done offline or online. Offline means that the video content in form of segments and the MPD will be created and then distributed over the CDN. This is the usual deployment strategy for on-demand content. DASH also supports live content preparation. Therefore, the content producer adds the segments online and updates the MPD accordingly. The MPD contains an update interval in order to inform the client that it must update the MPD during this period. On the network part nothing has to be changed because DASH uses the Internet as it is. As shown on Figure 3 the network may consist of CDNs that distribute the content. Due to the use of HTTP, DASH also passes firewalls and takes advantage of proxies. The flexible content preparation in case of different bitrates makes it possible that DASH could be used with nearly any connection as shown on Figure 3 where a mobile device is connected through a 3G network and a PC is connected through a 1Gbit connection. The actual streaming/adaptation logic is on the client side and, thus, the client decides which segments to download, the buffer size, and when the switch to another representation shall occur.

4.1 DASH Data Model

The DASH data model is hierarchical starting with the MPD. Each *MPD* could contain one or more Periods. The purpose of a *Period* is to separate the content, e.g., for ad insertion or for changing the camera angle in a live football game. Each Period could then contain one or more *Groups* which enables the grouping of different *Representations* that logically belong together. For example, representations with the same codec, language, resolution, etc. could be within the same group. This mechanism allows the client to eliminate a range of representations that does not fulfill its requirements. A period could also contain a *Subset* which enables the restriction of combinations of groups and expresses the intention of the creator of the MPD. Of course, the MPD also supports trivial cases just containing one or more representations of multiplexed content.

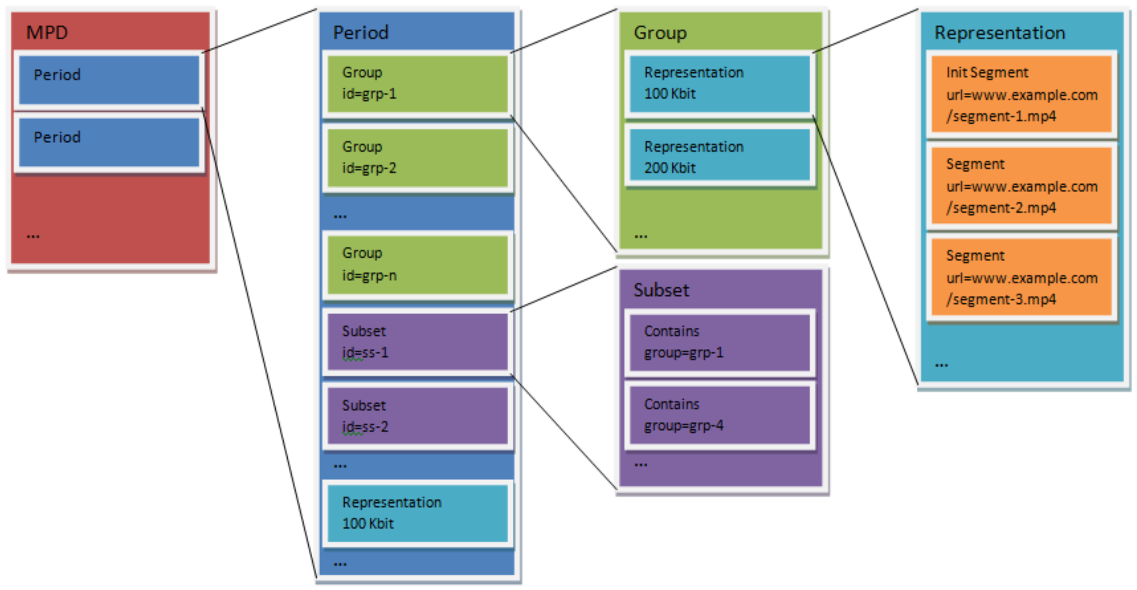

Figure 4. DASH Data Model.

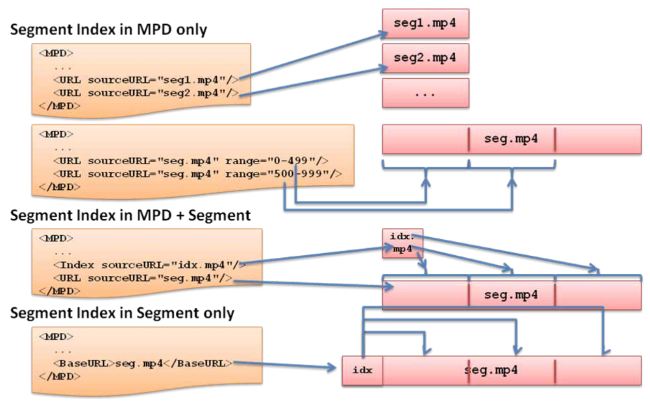
Segment Index in MPD only
<MPD>
...
<URL sourceURL="seg1.mp4"/>
<URL sourceURL="seg2.mp4"/>
</MPD>
seg1.mp4
seg2.mp4
...
<MPD>
...
<URL sourceURL="seg.mp4" range="0-499"/>
<URL sourceURL="seg.mp4" range="500-999"/>
</MPD>
seg.mp4
Segment Index in MPD + Segment
<MPD>
...
<Index sourceURL="idx.mp4"/>
<URL sourceURL="seg.mp4"/>
</MPD>
idx.mp4
seg.mp4
Segment Index in Segment only
<MPD>
...
<BaseURL>seg.mp4</BaseURL>
</MPD>
idx
seg.mp4

Figure 5. Segment Indexing.

The representations contain the actual *Segments*. Typically, the first segment of a representation is an initialization segment that contains only meta-information for the decoding process. The other segments contain the actual data (audio, video, text, etc.) and could be accessed from the client through the given URL.

4.2 Segment Indexing

DASH employs a very flexible segment indexing mechanism as depicted in Figure 5. In particular, there are three different methods how segments could be indexed. The first one is the MPD only mechanism where the index will be signaled within the MPD and not in the underlying data. However, there are two ways how to do that within the MPD. The first approach provides multiple segments on the server and the second approach provides only one file (i.e., one big segment). For the second one the MPD describes the byte ranges of the different segments. The second solution is a hybrid one where a part of the index is signaled in the MPD and the other one in the underlying data. This means that the MPD describes the location of a separate index file that contains the byte ranges of the segments. With this solution there are just two files located on the server. Finally, it is also possibly to include the whole index in the underlying data, i.e., at the beginning of file. All of these solutions could be used for on-demand use cases but only the first one is preferred for live sessions.

4.3 Deployment Remarks

As mentioned before, DASH is very flexible regarding the data model, indexing, segment duration, etc. which is intentionally as on-demand and live use case scenarios do have different requirements. For example, the segment duration may be as long as possible for on-demand sessions because segments are costly and unnecessary, byte range requests have much more caching efficiency, and on the server fewer files are needed. Typically there will be only one file on the server for each bitrate. However, for live sessions this is not a suitable approach as the session needs to be divided in multiple segments (i.e., the whole duration of the session is not known in advance). For live sessions also a high encoding performance is needed because several bitrates, languages, etc. must be encoded and packetized into segments in real time.

5. CONCLUSION AND FUTURE WORK

This paper describes an implementation of the emerging Dynamic Adaptive Streaming over HTTP (DASH) standard which is currently developed within MPEG and 3GPP. Our implementation is based on VLC and fully integrated into its structure as a plugin. Furthermore, our implementation provides a powerful extension mechanism with respect to adaptation logics and profiles. That is, it should be very easy to implement various adaptation logics and profiles. Future versions of the plugin will provide an update to the latest version of the standard (i.e., a lot of changes have been adopted recently, e.g., Group has changed to AdaptationSet), add support for persistent HTTP connections in order to reduce the overhead of HTTP streaming (e.g., compared to RTP), and seeking within a DASH stream. Finally, we will investigate adding support for live streaming. The plugin is available as open source at the DASH website of the Alpen-Adria-Universität Klagenfurt [11] which also provides a documentation describing how to build and use the plugin. Furthermore, test files and videos of the plugin are provided at this Web site.

6. ACKNOWLEDGMENTS

This work was supported in part by the EC in the context of the ALICANTE project (FP7-ICT-248652)

7. REFERENCES

[1] H. Schwarz, D. Marpe, T. Wiegland, “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, Sep. 2007, pp. 1103-1120.

[2] B. Wang, J. Kurose, P. Shenoy, and D. Towsley, “Multimedia Streaming via TCP: An Analytic Performance Study”, *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 4, no. 2, May 2008, pp. 16:1-16:22.

[3] 3GPP TS 26.234, “Transparent end-to-end packet switched streaming service (PSS)”, Protocols and codecs, 2010.

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

[5] Adobe HTTP Dynamic Streaming, http://www.adobe.com/products/httpdynamicstreaming/ (last access: May, 2011).

[6] R. Pantos, W. May, “HTTP Live Streaming”, IETF draft, March. 2011, http://tools.ietf.org/html/draft-pantos-http-live-streaming-06 (last access: May, 2011).

[7] C. Müller, C. Timmerer, “A Test-Bed for the Dynamic Adaptive Streaming over HTTP featuring Session Mobility”, *ACM Multimedia Systems*, San Jose, California, USA, Feb. 2011, pp. 271-276.

[8] ISO/IEC CD 23001-6. 2010. Information technology – MPEG systems technologies – Part 6: Dynamic adaptive streaming over HTTP (DASH), Guangzhou, China, Oct. 2010.

[9] DASH BasicCM profile, http://www-itec.uni-klu.ac.at/dash/?page_id=10#SupportedProfile (last access: May, 2011).

[10] E. Freeman, E. Freeman, K. Sierra, B. Bates, *Head First Design Patterns*, O’Reilly Media, Oct. 2004.

[11] DASH at Alpen-Adria-Universität Klagenfurt, http://www-itec.uni-klu.ac.at/dash (last access: Jul. 2011).