Live HTTP Streaming of Video and Subtitles within a Browser

Cyril Concolato Jean Le Feuvre
Telecom ParisTech
46, rue Barrault
75013 Paris, France
{cyril.concolato, jean.lefeuvre}@telecom-paristech.fr

ABSTRACT
Video streaming has become a very popular application on the web, with rich player interfaces and subtitle integration. Additionally, live streaming solutions are deployed based on HTTP Streaming technologies. However, the integration of video and subtitles in live streaming solutions still poses some problems. This paper describes a demonstration of live streaming of video and subtitle data using the MPEG-DASH technology and its synchronous playback in a web browser. It presents the formats, architecture and technical choices made for this demonstration and shows that it is feasible with upcoming browsers, paving the way for richer web video applications.

Categories and Subject Descriptors
H.5.1 [**Information Interface and Presentation**]: Multimedia Information Systems – *video*; H.5.4 [**Information Interface and Presentation**]: Hypertext/Hypermedia – *Architectures.*

General Terms
Performance, Experimentation, Standardization.

Keywords
HTML5, Streaming, Subtitling, Web technologies.

1. INTRODUCTION
Web-based video applications are now very popular thanks to the inclusion in the HTML5 standard of a video element, its support in all major browsers and the development of easy-to-use JavaScript libraries around HTML video Application Programming Interfaces. Additionally, solutions for video streaming over the Internet are being deployed using proprietary technologies such as Microsoft Smooth Streaming, Apple HTTP Live Streaming (HLS), Adobe HTTP Dynamic Streaming (HDS) or using standard technologies such as MPEG Dynamic Adaptive Streaming over HTTP (DASH) [1].

The combination of the HTML video environment and of HTTP streaming technologies opens the way for live and adaptive video streaming over the Internet, using the existing HTTP infrastructure of caches and Content Delivery Networks (CDN) known as Over The Top distribution (OTT), as an alternative or complement to traditional video broadcasting systems.

Subtitling or closed captioning has been a key component of video systems for a long time, providing a textual representation, possibly in a different language, of the audio track (subtitling) or providing a textual description, possibly extended with graphical elements, of the audio track associated with the video for the hearing impaired (captioning). While on the one hand there is a plethora of formats for subtitling and closed captioning, such as the Digital Video Broadcasting (DVB) subtitle formats used mainly in broadcast television, the Subtitle Rip (SRT) format used in conjunction with Internet video downloads, or the 3GPP Timed Text format for the mobile, none of these formats had had adoption on the Web. The recently developed WebVTT format [5], on the other hand, has been designed for the Internet, in particular with a tight integration with HTML, JavaScript and CSS. This format is also selected as a basis in Apple's adaptive streaming solution for subtitling [3].

The playback of subtitles together with video content in Web browsers has been demonstrated already by several sites[1]. However, the combined usage of subtitles and video content in a live scenario delivered using HTTP streaming technologies and played in a browser has not been demonstrated so far, in particular because of its underlying technical challenges. This paper proposes a demonstration of synchronous playback of video and subtitling content corresponding to a live event delivered over the Internet using HTTP streaming technologies and a web browser.

This paper first describes in Section 2 the demonstration formats and architecture, highlighting the challenges in such a demo, and the chosen approaches. Section 3 presents some results and discusses interesting aspects of the demo. Finally Section 4 concludes this paper and gives some ideas for future work.

2. DEMONSTRATION ARCHITECTURE
This section presents first the media formats used in the demonstrations and the specific aspects related to their adaptive streaming and browser usage; it then describes generic media segments aspects related to adaptive streaming and the specifics of the video and subtitle segments. Finally, it describes the usage of the MPEG-DASH Media Presentation Description (MPD) format [6] to represent delivery and synchronization information for the live streaming service.

MMSys'13, February 26-March 1, 2013, Oslo, Norway.

[1] http://www.html5rocks.com/en/tutorials/track/basics/

2.1 Media Formats

2.1.1 Unmultiplexed media segments

One of the main problems in building a dynamic adaptive streaming system is to properly handle the synchronization at the client side. Since adaptive streaming is used to switch media components according to their bandwidth, most systems use independent media component delivery. Each media, whether audio, video, subtitle or other, is packaged independently at the server side. The benefit of this design is that there is no need to duplicate media among files at the CDN side: the service provider only needs $N_{videos} + N_{audios} + N_{subtitles}$ files where a (partially) multiplexed approach would need up to $N_{videos} * N_{audios} * N_{subtitles}$ files, N being the number of versions of a media component. Therefore, the demonstration presented in this paper is based on the use of the unmultiplexed segment formats described below. In this demonstration, the video and subtitle segments have the same duration. This is not required and is just used for convenience.

2.1.1.1 Video segments

The video component in this demonstration is encoded using the AVC codec, and packaged as segments using the ISO Base Media File Format [4]. The video segments are made compatible with the Google Chrome support of the Media Source Extension draft specification [7]; in particular, each segment is composed of one and only one movie fragment, and each fragment starts with a random access point. These restrictions define a compatible subset of MPEG-DASH segment formats.

2.1.1.2 Subtitle segments

2.1.1.2.1 On segment duration

Audio-visual streams are by nature continuous media, with each coded frame lasting a short, usually constant time, without any gap between frames. Subtitles, and more generally many meta-data streams, differ in nature: each coded frame has an arbitrary duration and there may be some gaps or empty parts in the timeline; this makes the format less friendly for HTTP streaming environment, where media data is usually transferred by chunks of average constant duration. Using a constant duration for the segment avoids sending the timing of each segment, making the manifest file more compact and faster to download/update.

In the case of subtitling, it may happen that a subtitle may span over several target segment durations. In this case, the following approach may be used, depending on the media format:

- Copy the frame in each segment it overlaps, and signal in the media stream that some frames are copy of previous frames. This signaling informs players that have received the previous segment that they do not need to process this new frame, but just extend the duration of the previous one. This is the approach we follow for 3GPP timed text streams in ISO Base Media files.
- Split the subtitle frame in two or more frames to keep the segment duration constant on average; this is the approach we follow for WebVTT content.

The first approach provides a generic, media-agnostic solution to the detriment of bandwidth efficiency. The second approach seems to be the most appropriate one, as it avoids data duplication, and is usable with WebVTT content, but may not be well suited for all media types: it may not always be possible to split a frame in two or more frames.

Finally, we also use empty subtitle segments when no subtitle data is provided to cope with the fact that segments still need to be described in the manifest.

2.1.1.2.2 WebVTT segment

Because of its support in browsers, we chose the WebVTT format [5] to represent subtitling data. However, there is currently no standard way to define how a subtitle segment is made. MPEG is currently standardizing the storage of WebVTT content in the ISO Base Media file format, but it is neither finalized nor supported by browsers. Similarly, the WebM format has support for storing WebVTT tracks[2], but it is not yet supported by browsers. Additionally, in both cases ISO Base Media File Format and WebM, requiring in the production workflow an additional packaging of subtitles seems unneeded.

The latest[3] version of the documentation of the Apple HLS system describes the use of entire WebVTT files as segments. This approach is also what we chose, in particular because the concatenation of WebVTT files is gracefully handled by browsers: the header of the second file becomes misplaced and invalid in the concatenation, but it is simply ignored by the browser.

Hence, given the considerations explained above on subtitle segments durations, our WebVTT segments are simply WebVTT files where the cue timestamps are continuous, similarly to video frames. An example of WebVTT segment is given in Figure 1.

```
WEBVTT Segment 1


10
00:00:02.000 --> 00:00:02.200 line:0
This is cue 10 (start: 00:00:02.000 -- end:
00:00:02.200)
[...omitted in the paper for brevity ...]
18
00:00:03.600 --> 00:00:03.800 line:0
This is cue 18 (start: 00:00:03.600 -- end:
00:00:03.800)
```

Figure 1 - Example of WebVTT segment

In this example, the header indicates that it is a WebVTT file (with the WEBVTT signature) and indicates the segment number. This latter information is not used by the player and is just put in the file for debugging purposes. Each cue has an integer identifier giving its number in the session. This identifier is not used by the browser for playback timing. Again it is used only for debugging. Cues have their start and end times such that they don't overlap in time. This matches the non-overlapping segment approach in MPEG-DASH. Finally, each cue has a standard setting indicating that the text payload should be displayed on the first line from the top of the video. The text payload of the cue is composed in this demo of the cue identifier and the timing information.

[2] Storage of WebVTT content in WebM: http://wiki.webmproject.org/webm-metadata/temporal-metadata/webvtt-in-webm

[3] At the time of writing version 10.

2.1.2 Media Presentation Description

To describe the streaming service, we use the MPEG-DASH standard. In particular, we rely on MPD as given in Figure 2. We detail in this section some of the rationales for this MPD design.

2.1.2.1 Single, multiple, constant or updated MPD

For this demonstration, we are using two different tools to generate media data: one tool, MP4Box (see 2.2.1), to generate the video segments and the video MPD; and another tool to generate the subtitle segments and subtitle MPD. We then merge the two MPD to make a single MPD. This causes some problems related to timing issues that we solve in the next section (see 2.1.2.2). A possible alternative would be to use multiple MPD linked using the Xlink tools of DASH, but this is left for future work as the feature does not currently belong to any profile.

Given the way our MPD is produced in this demonstration, i.e. to avoid multiple MPD merges, we prefer constant MPD. However, since we need to describe live content, we chose to set the type attribute of the MPD to "dynamic", as specified in the MPEG-DASHs standard, but with a minUpdateTime attribute not specified, indicating that the MPD will not change. This is a convenient way to avoid MPD updates. Again, this should be improved for real deployment to cope with production errors, where MPD updates can be useful. This should be considered in future work.

2.1.2.2 MPD timing aspects

As a consequence of selecting unmultiplexed video and text components, the server/content producer has to make sure that each media component can be accurately positioned on the playback timeline at the client side. Timing of each media component is reconstructed through timestamps within a media container, but care needs to be taken to align the times in different containers. We have looked at two approaches to do that:

- Forcing all media components to use the same time origin for their timestamps. This may not be feasible or require repackaging of the different media containers, so we rejected this approach.
- Signaling to the client some offsets to apply to media timestamps in order to align the time origin of each media.

This second approach is for example used in HLS [3], with the X-TIMESTAMP-MAP metadata header placed in WebVTT segments, to map the timing of WebVTT cues to the audio/video timestamps in the MPEG-2 TS file. This is also the approach that we have selected, but using the presentationTimeOffset attribute specified in MPEG-DASH, as it does not create dependency of the subtitle segments on the media segments.

```
<MPD xmlns="urn:mpeg:DASH:schema:MPD:2011"
minBufferTime="PT10S" type="dynamic"
availabilityStartTime="2012-11-
16T16:32:46Z">
<ProgramInformation>
<Title>Live WebVTT/Video streaming</Title>
<Copyright>TelecomParisTech</Copyright>
</ProgramInformation>
<Period start="PT0S">
<AdaptationSet segmentAlignment="true">
<ContentComponent contentType="text"/>
<SegmentTemplate timescale="1000"
duration="1000" media="vtt\live-
segment$Number$.vtt" startNumber="0" />
<Representation mimeType="text/vtt"
bandwidth="100"/>
</AdaptationSet>
<AdaptationSet segmentAlignment="true">
<ContentComponent contentType="video"/>
<SegmentTemplate timescale="1000"
duration="1000" media="video\counter-
live$Number$.m4s" startNumber="0"
initialization="video\counter-
liveinit.mp4"/>
<Representation mimeType="video/mp4"
codecs="avc1.42c01e" width="640"
height="360" startWithSAP="1"
bandwidth="194835"/>
</AdaptationSet>
</Period>
</MPD>
```

Figure 2 - Example of DASH MPD for Live video and subtitle streaming

2.2 Software Components

The demonstration uses the following software components:

- The MP4Box tool from the GPAC project, revision 4226 [2], to produce the video segments and the MPD;
- A simple counter generator to produce WebVTT segments as described in 2.1.1.2;
- And Google Chrome (Canary version 24) to render the streamed content.

2.2.1 MP4Box usage

We use a specific option in MP4Box to generate live content called "dash context". This option enables running MP4Box several times while keeping some context information between different runs, so that the generation of media segments and MPD takes into account the previous runs. For instance, if the first run generated 10 segments of duration 2 seconds, the next run should produce segments whose decoding time stamps start at 20 seconds. MP4Box saves this contextual information in a text file provided in the command line with the "-dash-ctx <file>" option. This option allows producing a live event by calling MP4Box on a regular basis, to prepare the new video content for delivery using DASH. MP4Box also provides a mode in which it automatically calls the DASH segmentation on regular basis of the same input media files. These files cannot be updated in this mode but it provides a convenient way to generate a forever-lasting DASH live from a set of input media files.

An example of context is given in Figure 3. The file is composed of a general section [DASH], which holds general context information such as the first generation time of the MPD or base name for segment. Then, for each representation, a section documents the global properties of the representation, such as bandwidth or initialization segments, as well as the properties to reload at the next run, such as the next segment number to use or the next decoding time. Finally, a section documents the media start time of each segments in the active period.

```
[DASH]
SessionType=dynamic
TimeShiftBufferDepth=5000
GenerationTime=Tue Nov 13 12:01:41 2012
GenerationNTP=3561796901
SegmentTemplate=counter-live
MaxSegmentDuration=1.000000

[Representation_1]
Source=..\counter10s.mp4
Setup=yes
Bandwidth=302456
InitializationSegment=counter-liveinit.mp4
InitializationSegmentSize=856
TKID_1_NextDecodingTime=249000
NextSegmentIndex=11
NextFragmentIndex=11
CumulatedDuration=9.960000

[SegmentsStartTimes]
counter-live-1.m4s=0
...
counter-live-9.m4s=8
```

Figure 3 - Example of DASH context information used by MP4Box

At each call, MP4Box uses the segment timing information and the time shift buffer depth (5000 ms in the example) to remove all segments that are no longer in the time shifting buffer, based on the NTP of the last generation. This enables running the demonstration a long time without creating too many files.

2.2.2 Rendering with Google Chrome

To render the service described by the MPD, we use the Google Chrome browser. The browser loads an HTML page and executes some embedded JavaScript code implementing the following algorithm:

1. Fetch and parse the MPD, in a manner similar to what is done by the DASH-JS library [8];
2. Create an HTML <video> element to render the video component and an HTML <track> element as a child of the <video> element, to indicate to the browser the presence of a subtitle track, which should be synchronized with the video;
3. Create dummy <video> and <track> elements to enable the parsing of WebVTT segments (see step 4.f)
4. Start a loop and perform the following operations:
    a. Determine the current UTC time in the browser;
    b. Compare this time with the availability start time in the MPD, to determine the Media Presentation Time;
    c. Derive the video and WebVTT segment number from the Media Presentation Time
    d. Issue an HTTP request, using the XMLHTTPRequest object, to retrieve the video segment;
    e. Forward the media segment to the video decoding buffer using the Media Source Extension API [7] following the Youtube example DASH player[4];
    f. Change the src attribute of dummy <track> element to point to the new WebVTT segment, wait for the event indicating that the WebVTT file is loaded, and transfer the parsed cues to the real video element created in step 2.
    g. Wait until subsequent segments need to be fetched.

3. RESULTS AND DISCUSSION

3.1 Results

Figure 4 shows a snapshot of Google Chrome rendering the video and subtitle content. The text content displayed at the top of the video is the subtitle content. As it can be seen in the image, the text is synchronized with the video: the displayed video frame is the frame located at time 3.960s in the video while the text cue being rendered shows the range between 3.800s and 4.000s. In some initial tests, we saw that some text cues would be dropped. This was apparently due to a too high frame-rate for cues (200ms) since when the cue rate is set to 500ms, this problem disappears. While this may prove to be a problem to deliver some high frame rate graphical overlays, such as annotations; for subtitles, a cue rate of 500 ms should be sufficient.

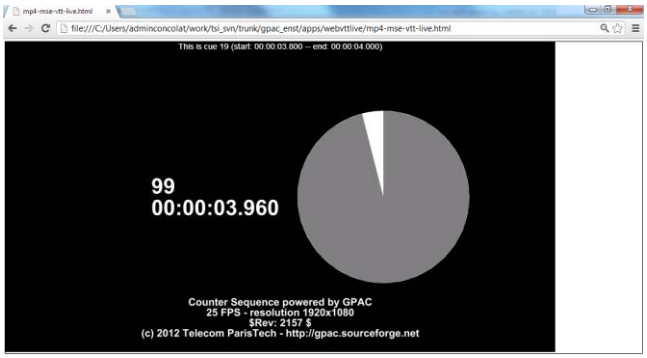

Figure 4 - Rendered result in Google Chrome

3.2 Discussions

We have tested the demonstrator on the long run (several hours). This test highlighted in particular the impact of non-accurate segment duration and generation. Indeed, a first version of the input video file was segmented every second but had a total duration of 9.96s (a 10s file missing 1 frame at 25 frames per second). In this case, although the rendering of the video and subtitle would stay synchronized, after some period of time, the fetching of the video segments, which is based on the 10s segment duration indicated in the MPD, would lead to the wrong video segment number and the video would stop. Similarly, we experienced some problems after long running sessions, due to a

[4] YouTube DASH prototype player, http://dash-mse-test.appspot.com/dash-player.html

drift between the segment generation and segment fetching algorithm. We plan to provide a better fetching algorithm in future work, which would determine the buffer occupancy when fetching media segments.

Another test was to load the HTML page (and thus start the playback) at different times after the start of the segment processing to verify that the browser could indeed start displaying the live content, i.e. the oldest in the time shift buffer, at any time. A first problem encountered was if the browser was started at time T (e.g. an hour) after the segment generation started. According to the HTML draft standard, upon receiving the first video frames, the browser inspects the timestamps. If it is non-zero, the browser will stall, expecting the previous video frames to be displayed. We've found two approaches to avoid this problem:

- The first one using the currentTime attribute of the HTML video element, to indicate to the browser to directly start playing the video T seconds into the media stream. This was successful but had the side effect of displaying a timeline where only the last small fraction was usable for seeking, as shown in Figure 5.

Figure 5 - Timeline during a video playback started a long time after the start of the video segment generation

- The second approach using the timestampOffset attribute of the Media Source Extensions draft standard. This tool instructs the media parser to shift all timestamps by a given value before feeding them to the media decoder. Thus, if the timestamp offset is equal to T, the media decoder will process the first frame with a new timestamp of 0 and will not stall. This means that the JavaScript layer needs to have access to the timestamp of the first packet. In our approach, we did not want the JavaScript code to parse the media data to determine this timestamp offset[5]. Instead, we used the segment duration given in the MPD, but this proved not really reliable when the segments were only approximately constant in duration. Finally, we relied on the presentationTimeOffset attribute given in the MPD[6].

We have also tested the playback in multiple clients, on multiple machines. The result in this case is that both browsers are synchronized, playing the same segment or two consecutive segments.

4. CONCLUSION

This paper presented the details of a demonstration of synchronized playback of live video and subtitle content in a web browser based on HTTP streaming technologies. The details included the selected media formats, software and content generation and playback approaches. The demonstration shows that it is feasible with Open-Source Software to stream and play live video with subtitles on the Internet over long running sessions. The paper discussed on the complex combination of the different technologies involved. Given that these technologies are based draft standards, it is expected that modifications will be made to these standards to simplify this combination especially to extend such demonstrations. Extensions, as part of future work, could be made to provide mechanisms for trick play of live content, or to support additional media tracks such as live annotated graphics.

[5] It could minimally parse the media data, i.e. the "sidx" box of the ISOBMFF, as done in the YouTube demo player, to determine the offset.

[6] For the timed-text track, we had to adjust manually the times of each cue since the Media Source Extension timestampOffset was not available for that track.

5. ACKNOWLEDGMENTS

This work was supported in part by the French funded project AUSTRAL (DGCIS FUI13).

6. REFERENCES

[1] Thomas Stockhammer. 2011. Dynamic adaptive streaming over HTTP --: standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems* (MMSys '11). ACM, New York, NY, USA, 133-144. DOI=10.1145/1943552.1943572 http://doi.acm.org/10.1145/1943552.1943572

[2] Jean Le Feuvre, Cyril Concolato, Jean-Claude Dufourd, Romain Bouqueau, and Jean-Claude Moissinac. 2011. Experimenting with multimedia advances using GPAC. In *Proceedings of the 19th ACM international conference on Multimedia* (MM '11). ACM, New York, NY, USA, 715-718. DOI=10.1145/2072298.2072427 http://doi.acm.org/10.1145/2072298.2072427

[3] R. Pantos, W. May, October 15, 2012. HTTP Live Streaming, http://tools.ietf.org/id/draft-pantos-http-live-streaming-10.txt

[4] ISO/IEC 14496-12:2012, Information Technology – Coding of audio-visual objects – Part 12: ISO Base Media File Format

[5] Ian Hickson, WebVTT, November 5, 2012. http://dev.w3.org/html5/webvtt/

[6] ISO/IEC 23009-1:2012, Information Technology – Dynamic Streaming over HTTP (DASH) – Part 1: Media Presentation description and segment formats, http://standards.iso.org/ittf/PubliclyAvailableStandards/c057623_ISO_IEC_23009-1_2012.zip

[7] A. Colwell, A. Bateman, M. Watson, Media Source Extensions, W3C Editor's draft 9 November 2012, http://dvcs.w3.org/hg/html-media/raw-file/tip/media-source/media-source.html

[8] Rainer, Benjamin; Lederer, Stefan; Muller, Christopher; Timmerer, Christian; , "A seamless Web integration of adaptive HTTP streaming," Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European , vol., no., pp.1519-1523, 27-31 Aug. 2012