INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11 MPEG2012/M26906
October 2012, Shanghai, China

Source **Telecom ParisTech**
Status **For consideration at the 102nd MPEG meeting**
Title **Demonstration of Live Streaming Video and Subtitles with DASH**
Author Cyril Concolato, Jean Le Feuvre

1 Introduction

This document describes a demonstration of live streaming of synchronized video and subtitles over the Internet. The key components of this demonstration are the use of

- The MP4 file format for the packaging of the live video stream into segments,
- The WebVTT subtitle format to represent live subtitle segments,
- MPEG DASH to represent a dynamic long-running streaming service and to enable synchronization of the video and subtitle streams.

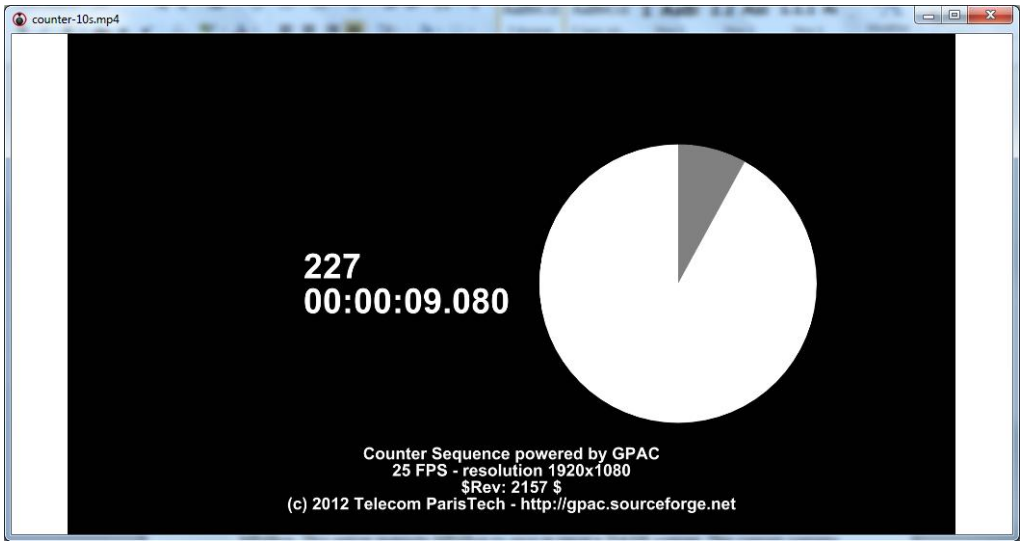
This demonstration is meant to show that it is possible to use MPEG-DASH and the MP4 file format to fulfill similar use cases to the latest version of Apple HTTP Live Streaming[1].

2 Implementation details

2.1 Video Segments

The video segments are generated on the fly using MP4Box from the GPAC Open Source project. In particular, a single video file (the counter sequence proposed as DASH conformance sequence) is segmented periodically using the -dash-ctx option of MP4Box. A snapshot of the video file is shown below.

[1] http://tools.ietf.org/html/draft-pantos-http-live-streaming-09



This dash-ctx option instructs MP4Box to save or reuse a DASH context. This context contains the necessary information to enable MP4Box to produce segments as if they were the next segments from a live stream. An example of context is given below:

```
[DASH]
InitializationSegment=counter-10s_dash_init.mp4
BitstreamSwitching=yes
InitializationSegmentSize=860
MaxSegmentDuration=1.000000
NextSegmentIndex=11
NextFragmentIndex=11
PeriodDuration=9.960000

[TrackID_1]
NextDecodingTime=249000
```

2.2 Subtitle Segments

A WebVTT segment is a file or part of a file as follows:

```
WEBVTT Segment 6

61
00:00:12.200 --> 00:00:12.400
This is cue 61 (start: 00:00:12.200 -- end: 00:00:12.400)

...

67
00:00:13.400 --> 00:00:13.600
This is cue 67 (start: 00:00:13.400 -- end: 00:00:13.600)

68
00:00:13.600 --> 00:00:13.800
This is cue 68 (start: 00:00:13.600 -- end: 00:00:13.800)
```

```
69
00:00:13.800 --> 00:00:14.000
This is cue 69 (start: 00:00:13.800 -- end: 00:00:14.000)

70
00:00:14.000 --> 00:00:14.200
This is cue 70 (start: 00:00:14.000 -- end: 00:00:14.200)
```

2.3 MPD

An example MPD of such service could like:

```
<MPD xmlns="urn:mpeg:DASH:schema:MPD:2011" minBufferTime="PT10S"
type="dynamic" availabilityStartTime="2012-10-10T20:05:43Z">
<ProgramInformation
moreInformationURL="http://concolato.wp.mines-
telecom.fr/2012/09/12/webvtt-streaming/">
<Title>Live WebVTT / Video streaming using DASH</Title>
<Copyright>TelecomParisTech(c)2012</Copyright>
</ProgramInformation>
<Period start="PT0S">
<AdaptationSet segmentAlignment="true">
<ContentComponent contentType="text"/>
<SegmentTemplate         timescale="1000"         duration="2000"
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="2000"
media="video\live-segment$Number$.vtt" startNumber="0" />
<Representation     mimeType="video/mp4"     codecs="avc1.42c01e"
width="640" height="360" startWithSAP="1" bandwidth="194835"/>
</Period>
</MPD>
```

Interestingly, this MPD does not specify a minUpdateTime attribute as all samples are known in advance.

2.4 Rendering

For the rendering of this content, we use the Chrome Canary browser (Version 24.0.1291.1). The browser renders an HTML document which contains some JavaScript code. This code does the following:

- Loads and parses the MPD,
- Creates a <video> element to render the video and a <track> element to render the subtitles,
- Fetches the video and subtitles segments using the XMLHttpRequest API (as demonstrated in DASH-JS[2]) ,

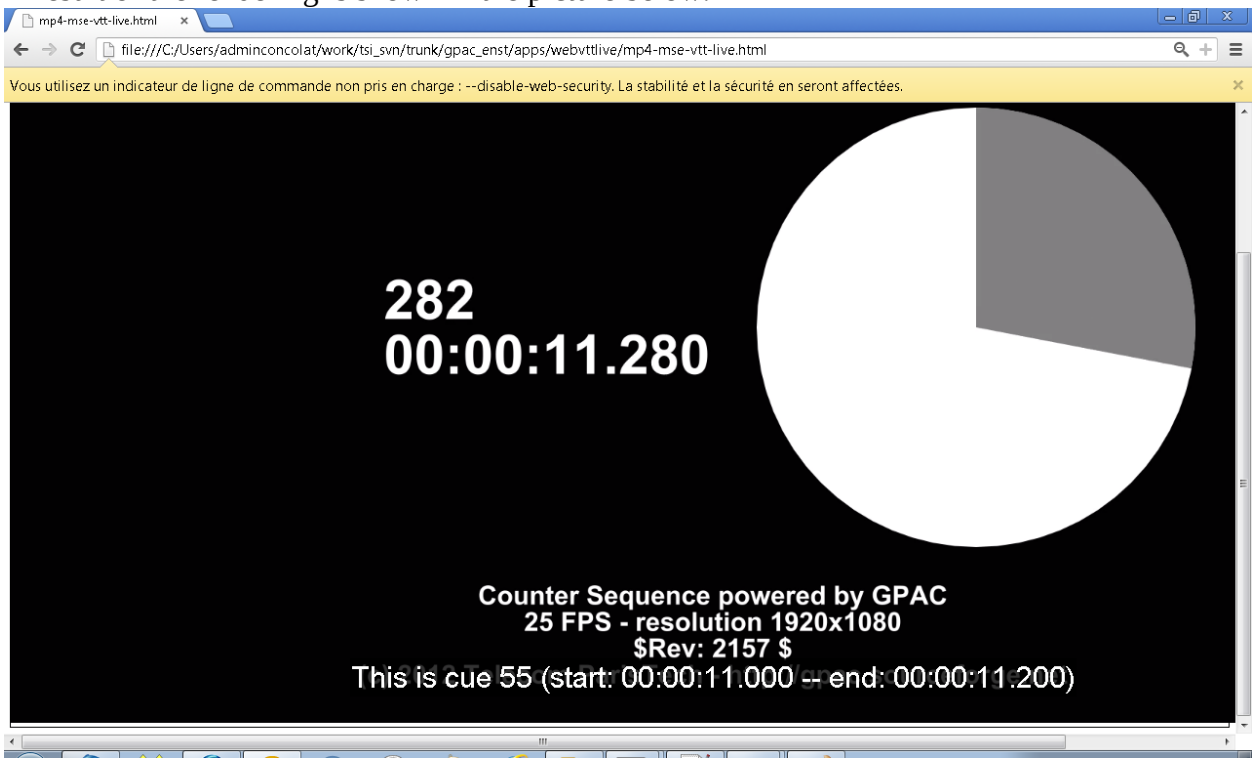
[2] http://www-itec.uni-klu.ac.at/dash/?page_id=746

- Feeds the video segments to the media decoder engine using the Media Source extension API (as demonstrated by YouTube MPEG DASH player[3]),
- Loads each subtitle segment separately with an additional dummy <video> element[4] and transfers the parsed cues to the real video element.

In this demonstration, synchronization is ensured by fetching synchronizable video and subtitle segments (based on the segment availability and duration and template) in order to receive them at a similar time, the rest of the synchronization between video frames and subtitle cues being handle by the video and track elements.
Note: For this demonstration, timestamps in WebVTT segments and video segments use the same timeline. This is a restriction which could be removed exploiting the MPD presentationTimeOffset attribute or the timestampOffset property of the MediaSource API.

3 Results

A result of the rendering is shown in the picture below.



As can be seen on the image, there is a slight mis-synchronization (by one frame). Additionally, in real time, the playback also suffers from some rendering problems, for instance some subtitles are not displayed.

However, despite these problems, this demonstration shows that it is possible to do live streaming of video and subtitles over the Internet using DASH, MP4, WebVTT and to render the content in coming browsers.

[3] http://dash-mse-test.appspot.com/dash-player.html
[4] This dummy element is needed because there is currently no JavaScrpit API to parse WebVTT text into text cues or to feed the track buffer of type WebVTT