DASH Eventing and HTML5

Giri Mandyam
February, 2018
mandyam@qti.qualcomm.com
Introduction

• DASH events are timed actions that are meant to take place in conjunction with media playout

• DASH events may be conveyed via
  1. The MPD (Media Presentation Description) Document
     • The EventStream fragment
  2. The ‘emsg’ box in the ISO BMFF media track
     • In-band event

• Neither mechanism is well-suited for HTML5
  ◦ MPD eventing requires the application (HTML/JS) to execute the events as per media playback time
     • Application may not have time-accurate insight into media playback status
  ◦ emsg eventing support is non-existent in current browsers
     • Even if browsers propagated emsg data into JS, the event handler in the application can become a bottleneck
How does HTML5 handle DASH events today?

• HTML5 chose the route of text track cues
• Text track cues are basic elements in web-friendly text track formats (e.g. TTML/WebVTT)
• As per the HTML5 specification, a text track cue consists of (not comprehensive)
  ◦ An identifier
  ◦ An arbitrary string.
  ◦ A start time
  ◦ An end time
  ◦ A pause-on-exit flag
  ◦ Some additional format-specific data
  ◦ The data of the cue, including rules for rendering the cue in isolation.
How does HTML5 handle DASH events today? (cont.)

- Text track cues could be video chapters, captions, or even metadata (which would correspond to events)
  - Cues that can be rendered directly (e.g. captions) would not require special JS handling
- Example WebVTT with caption (voice span) and metadata

```
WEBVTT

00:11.000 --> 00:13.000
<v Roger Bingham>We are in New York City

00:13.000 --> 00:15.000
METADATA HERE; NO TAGS

00:15.000 --> 00:16.000
<v Roger Bingham>We're actually at the Lucern Hotel, just down the street

00:16.000 --> 00:18.000
<v Roger Bingham>from the American Museum of Natural History
```
HTML5 Handling of Text Track Cues

• `<video>` tag allows for track-specific event handlers
  ◦ `oncuechange` event handler provides all current cues to JS

• DataCues have also been defined, which allow for binary payload
  ◦ Browser vendor support is currently non-existent, as feature is controversial
    • [https://groups.google.com/a/chromium.org/forum/#!topic/blink-dev/U06zrT2N-Xk](https://groups.google.com/a/chromium.org/forum/#!topic/blink-dev/U06zrT2N-Xk)

• HbbTV has identified problems with short duration cues that pose problems for this mechanism for DASH event handling
  ◦ See [http://lists.w3.org/Archives/Public/public-inbandtracks/2013Dec/0004.html](http://lists.w3.org/Archives/Public/public-inbandtracks/2013Dec/0004.html)
ATSC 3.0 Approach

• The ATSC runtime environment model supports two approaches to rendering DASH streaming services
  ◦ Application Media Player (AMP)
  ◦ Receiver Media Player (RMP)

• The AMP is assumed to be standard HTML/JS (particularly libraries such as Dash.js) included in a downloadable application provided by a broadcaster
  ◦ The application would be “bound” to the over-the-air service
  ◦ Suitable for certain types of devices (e.g. smartphones, tablets) that can render DASH media but would not have an integrated ATSC receiver
  ◦ Meant to execute in standard browser context

• The RMP would be available in devices where ATSC reception and rendering take place (e.g. smart TV’s)
  ◦ RMP exposes controls via a WebSocket interface
ATSC 3.0 Event Handling

- **AMP**
  - Given that AMP runs in browser context, access to in-band event (‘emsg’) is not supported
    - Transcoding in-band events to text track cues at the transmitter side is an option to overcome this
  - MPD-carried EventStream can be parsed/handled directly

- **RMP**
  - Extracts both in-band and EventStream from incoming DASH media
  - Conveys event data to application via a WebSocket control interface

Both methods incur latency in event handling

- **AMP**
  - Dependent on timely extraction of text track cues by browser implementation
  - Application logic should be properly designed to handle event data as soon as it is encountered

- **RMP**
  - Implementation must extract event data from incoming media stream and relay to application in a timely manner
  - Application logic should also be efficient in handling incoming event data
Event Retrieval in ATSC 3.0
From A/344 “ATSC 3.0 Interactive Content”
Emerging Approach

• MPEG recently commenced work on carriage of web resources via ISO BMFF File Container

• Use of dedicated track data for interactivity (HTML and JS) would allow for rendering of media-timed events with or without application handler

• High-level processing flow:
• Direct rendering of media-timed interactivity now possible with media players with integrated web runtime engines

• Several approaches being considered in MPEG

• Streaming service example:
  ◦ Service includes baseline HTML document with appropriate updateable sections
    • Delivered within an Initialization Segment
  ◦ Ensuing track data will update sections of this document (e.g. <div>’s)