A WEB-BASED MEDIA SYNCHRONIZATION FRAMEWORK FOR MPEG-DASH

Volker Jung, Stefan Pham, Stefan Kaiser

Fraunhofer Institute for Open Communication Systems FOKUS
{volker.jung,stefan.pham,stefan.kaiser}@fokus.fraunhofer.de

ABSTRACT

The increasing amount of browser-enabled connected devices per household opens up new opportunities for multi-screen media consumption. As HTML5 is maturing, Web technology offers tools that enable features such as video synchronization across different stationary and portable devices. In this paper the Fraunhofer FOKUS Web-based media synchronization framework is presented in two scenarios: "Video Wall" and "Silent TV". Both scenarios leverage MPEG-DASH, an HTTP adaptive streaming technology, for more advanced scenarios.

Index Terms— Video synchronization, MPEG-DASH, WebSockets, WebRTC

1. INTRODUCTION

Nowadays many households have multiple multimedia-capable devices that are connected to the Internet, e.g. PCs Smart TVs, smartphones and tablets. What most of them have in common, is a Web browser. With HTML5 the HTMLMediaElement [1] has been introduced and facilitates media playback inside HTML5 browsers without the need for proprietary plugins, and hereby provides a consistent API for media playback. This paper describes the FOKUS media synchronization framework (WMSF), which allows HTML5-enabled browsers to synchronize play states and events across devices in a synchronization group. The term synchronization group is used to describe a group of connected clients. The framework takes advantage of WebRTC (Real-Time Communications) [2] for peer-to-peer (P2P) communication between browsers that support WebRTC, but also implements a Web-based fallback solution, e.g. WebSockets. These are the basic features of the framework:

- Join a synchronization group at its current playposition
- Selection and synchronization of the same or different audio/video
- Playback control from any device

In the next two chapters we will describe the concept and components of WMSF. In chapter 4 the two use cases "Video Wall" and "Silent TV" are presented and what advantages they gain by using MPEG-DASH as their media format.

2. PEER COMMUNICATION

In order to implement inter-browser synchronization, communication between browsers must be established beforehand. As the use of WebRTC for P2P communication is encouraged, but is not supported in all browsers, a Web-based fallback solution must be provided. Non-WebRTC inter-browser communication requires an additional Web server. For this purpose we use node.js as the Web server with the Socket.IO module, which chooses the best possible connection type, e.g. HTTP long polling or WebSockets. In case that WebRTC clients and non-WebRTC clients are connected to the same synchronization group, a communication channel has to be established between both groups, which is handled by this additional server, which we call the *communication broker*.

The physical connected network structure builds a star architecture in case that only WebRTC clients, or non-WebRTC clients, are connected to a synchronization group. Non-WebRTC clients form a star around the *communication broker* and WebRTC clients form a star around one chosen WebRTC client, from now on called *wrtc-master*. If both types of clients are present, a connection has to be established between the *communication broker* and the *wrtc-master*, to obtain a fully connected network. This allows non-WebRTC clients to join the synchronization group as well.

In case of connection loss, e.g. when a browser closes or crashes or just loses the Internet connection, all remaining clients are notified of the loss. Should the *wrtc-master* lose the connection, the *communication broker* choses a new *wrtc-master* among the remaining WebRTC clients.

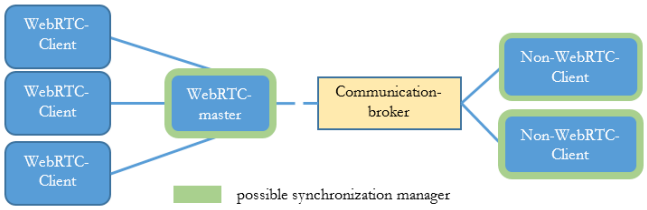

Figure 1 WebRTC & Non-WebRTC clients connected via the communication broker

3. PLAYBACK SYNCHRONIZATION

As soon as a connection is established, browsers are capable of communicating playback event states of the HTMLMediaElement. The actual synchronization has to be handled next. Ishibashi, Tsuji and Tasaka [3] proposed a scheme where a *synchronization manager* is introduced. In this scheme, clients unicast their current play state to the *synchronization manager*, which in essence, computes and "broadcasts" corrected playout timings in control packages. By using this scheme, it is not necessary to design the underlying network as a full-meshed network structure.

To synchronize media elements, one of the clients is chosen to be the *synchronization manager* (see Figure 1). Subsequently, all clients synchronize an internal virtual clock to the time of the *synchronization manager*. Once the manager decides, that all clients should start their playback, it "broadcasts" a *startPlayback(time)* message. The clients are then responsible to start their playback at the specified time. Unfortunately, the HTMLMediaElement API is not equally implemented on all browsers and devices. Since not all functions can be expected to work, a smallest common denominator of playback events has been chosen as synchronization events. These events are: *play*, *pause* and *seek*. Additionally the manager only sends a *startPlayback* message, when all clients have communicated that they are able to start playback (e.g. buffer is filled and media API is initialized). When a client is not able to resume its playback because of buffer-underruns, all clients pause and wait until every client is ready again.

4. USE CASES

We have selected two use cases to showcase and validate functionalities of WMSF. Furthermore, we are streaming the media as MPEG-DASH to enable more innovative and challenging scenarios. MPEG-DASH is a recently specified standard for HTTP Adaptive Streaming and is supported in most Web Browsers by using the Media Source Extension API.

2.1. Video Wall

The "Video Wall" serves as a technology demonstration for WMSF. The basic idea is to prepare a tiled video, where an independent screen represents each tile. An independent screen here means a phone, tablet or TV, which runs an HTMLMediaElement-enabled browser and when playing back media, is synchronized with other screens. When positioning the tiles next to each other in the right order, one large screen displays the video. With the help of WMSF play states (e.g. pause/play, seeking) are synchronized across all tiles of the "Video Wall" as well. For our experimentation the "Video Wall" is made up of four screens (2x2 matrix) or nine screens (3x3 matrix).

Besides serving as a technology demonstration, the "Video Wall" is also a test bed for MPEG-DASH streaming. One of the challenges when dealing with HTTP adaptive streaming technology is the client-side bit-rate switching algorithm. For example, playback interruptions should be avoided, the maximum possible video quality should be achieved and the amount of bit-rate switches should be minimized. A balance between these goals needs to be achieved for a proper algorithm. By manipulating the available bandwidth for the devices in the "Video Wall", even more complex bit-rate switching behavior can be observed. This is because each client decides independently when to switch bit-rates. Basically the clients are competing against each other for bandwidth. For the purpose of applying varying bandwidth limitations on the fly we use dummynet, a network emulation tool.

2.2. Silent TV

The second demonstration consists of a TV screen and multiple second screen devices, e.g. phone or tablet, which are connected via WMSF. One of the features in the ISOBMFF profiles of MPEG-DASH is that the media content is not muxed, meaning that for a simple DASH stream, there is separate video and audio track. This allows content providers to offer separate audio tracks (e.g. different audio languages), which are then muxed with the video "just in time" when played back by a DASH client. In the demonstration we leverage this feature to have the video track only displayed on the (silent) TV and the audio track(s) to be played back synchronously on the second screen. MPEG-DASH allows streams to be audio- or video-only. Imagine a scenario where two people would like to watch a movie together, but cannot decide which language to select. Using a headphone plugged into the second-screen, both can enjoy the movie in their preferred language. In combination with WMSF, a seamless user experience can be guaranteed. Moreover, the second screens serve as remote controls, since all play states are synchronized across devices.

3. REFERENCES

[1] W3C. (2009, August 25). HTML5 [Online]. Available: http://www.w3.org/TR/2009/WD-html5-20090825/video.html#htmlmediaelement

[2] A. Bergkvist, D. C. Burnett, C. Jennings. (2014, March 21). WebRTC 1.0: Real-time Communication Between Browsers [Online]. Available: http://dev.w3.org/2011/webrtc/editor/webrtc.html

[3] Y. Ishibashi and S. Tasaka, "A group synchronization mechanism for live media in multicast communications," Global Telecommunications Conference, 1997. GLOBECOM '97., IEEE (Volume:2), pp. 746 - 752, March 1997.