Implementing the Media Fragments URI Specification

Davy Van Deursen
Ghent University - IBBT,
Ghent, Belgium
davy.vandeursen@ugent.be

Raphaël Troncy
EURECOM
Sophia Antipolis, France
raphael.troncy@eurecom.fr

Erik Mannens
Ghent University - IBBT,
Ghent, Belgium
erik.mannens@ugent.be

Silvia Pfeiffer
Vquence
Sydney, Australia
silviapfeiffer1@gmail.com

Yves Lafon
W3C
Sophia Antipolis, France
yves@w3.org

Rik Van de Walle
Ghent University - IBBT,
Ghent, Belgium
rik.vandewalle@ugent.be

ABSTRACT

In this paper, we describe two possibilities to implement the W3C Media Fragments URI specification which is currently being developed by the Media Fragments Working Group. The group's mission is to address media fragments on the Web using Uniform Resource Identifiers (URIs). We describe two scenarios to illustrate this implementation. More specifically, we show how User Agents (UA) will either be able to resolve media fragment URIs without help from the server, or will make use of a media fragments-aware server. Finally, we present some ongoing discussions and issues regarding the implementation of the Media Fragments specification.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information System**]: Audio, Video and Hypertext Interactive Systems; I.7.2 [**Document Preparation**]: Languages and systems, Markup languages, Multi/mixed media, Standards

General Terms

Languages, Standardization

Keywords

Media Fragments, Video Accessibility

1. INTRODUCTION

Media resources on the World Wide Web (WWW) used to be treated as 'foreign' objects, which could only be embedded using a plugin that is capable of decoding and interacting with the media resource. The HTML5 specification is a game changer and most of the popular browsers support already the new `<video>` element. However, specific media servers are generally still required to provide for server-side features such as direct access to time offsets into a video without the need to retrieve the entire resource. Support for such media fragment access varies between different media formats and inhibits standard means of dealing with such content on the Web [7].

The W3C Media Fragments Working Group[1] (MFWG) is part of W3C's Video in the Web activity whose goal is to make video a "first class citizen" on the Web. In this context, the mission of the MFWG is to address media fragments on the Web using Uniform Resource Identifiers (URIs) [1]. Following a requirement phase [9], three different axes have been identified for media fragments: temporal (i.e. a time range), spatial (i.e. a spatial region), and track (i.e. a track contained in the media resource). Furthermore, media fragments can be identified by name, which is a combination of the aforementioned three axes.

In this paper, we present a partial implementation of this specification. We illustrate this implementation using two scenarios. In scenario (a), Alice has received on her Facebook wall a status message containing a Media Fragment URI, that highlights 75 seconds of a video clip. In scenario (b), Alice creates a Web page embedding this video and adding a Spanish audio track (another resource), and sends it to José, her Spanish nephew. We first show the different possibilities to implement the Media Fragments specification based on these two scenarios (sections 2 and 3). We identify then a list of current implementation problems (in section 4). Finally, we give our conclusions and outline future work in Section 5.

2. PROCESSING MEDIA FRAGMENTS WITHIN THE USER AGENT

In scenario (a), Alice receives Facebook notifications on her smart phone. She wants to watch the 75 seconds of the movie using the media fragment URI posted on her wall:

`http://example.com/video.ogv#t=25,100`

In order to be able to play this part of the video without downloading the whole media resource, the time range needs to be mapped to one or more corresponding byte ranges.

Let's suppose Alice has a smart UA at her disposal, in the sense that it can map by itself a fragment identifier into a range request expressed in bytes. When Alice clicks on the Media Fragment URI link, the following steps occur:

1. The UA parses the media fragment identifier and maps the fragment to its corresponding byte ranges. Note that this mapping highly depends on the underlying media format. For example, a UA can collect the header information of MP4 files (containing tables giving a complete time and byte-offset mapping) by asking the first couple of bytes of the file [2]. Further,

[1] `http://www.w3.org/2008/WebVideo/Fragments/`

WWW2010, April 26-30, 2010, Raleigh, North Carolina.

Ogg files support temporal seeking over the network by applying a bisection search algorithm over the Ogg pages in the file [4].

2. Based on the found byte ranges, the UA sends one or more HTTP Range requests in terms of bytes to the server. Note that, in this scenario, an HTTP 1.1-compliant Web server supporting byte range requests is enough to serve media fragments.

3. The server responds with a 206 Partial Content response, containing the bytes corresponding to the requested media fragment. Finally, the UA starts playing the requested media fragment. The HTTP communication between the UA and the server is listed in Listing 1.

Listing 1: Accessing media fragments on an HTTP server.

```
# HTTP request
GET /video.ogv HTTP/1.1
Host: www.example.com
Accept: video/*
Range: bytes=19147-22880

# HTTP response
HTTP/1.1 206 Partial Content
Accept-Ranges: bytes
Content-Length: 3743
Content-Type: video/ogg
Content-Range: bytes 19147-22880/35614993

{binary data}
```

The advantage of processing media fragments within the UA is that media fragments can be served by a traditional HTTP Web server. On the other hand, UAs need to be aware of the syntax and semantics of media formats. Furthermore, whether media fragment to byte range mapping is possible or not within the UA, without having the full original media resource at its disposal, highly depends on the media format.

3. PROCESSING MEDIA FRAGMENTS WITH SERVER HELP

In scenario (b), Alice wants to share her experience with José, her Spanish nephew. However, she knows that her nephew is not fluent as she is in English. Therefore, she quickly sets up an HTML 5 web page containing just the video track of the movie fragment she saw:

```
http://example.com/video.ogv#t=25,100&track='video'
```

Alice adds also a Media Fragment URI to this Web page, pointing to the corresponding Spanish audio track, which is located in a different media resource:

```
http://example.com/video_es.ogv#t=25,100&track='audio'
```

Let's suppose that José has a normal UA that cannot map media fragments to bytes by itself, but understands also media fragments URI. When José starts to play the video, the following steps occur:

1. The UA parses the two media fragment identifiers and creates for each of them an HTTP Range request expressed in a different unit than bytes. More specifically, the Range header is expressed with time and/or track units, as illustrated in Listing 2 [8].

2. The server, which understands these other units, interprets the HTTP Range request and performs the mapping between media fragment and byte ranges. Based on this mapping, the server selects the proper bytes and wraps them within a HTTP 206 Partial Content response. Note that such a response also contains additional Content-Range headers describing the content range in terms of time and tracks (see Listing 2).

3. The UA needs to synchronize the two received media resources. This can be easily implemented using HTML5 and a bit of Javascript (an excerpt of this is shown in Listing 3) [5]. Synchronization is obtained by setting the *currentTime* properties right for each media element. Note that, as pointed out in [3], drift problems can occur since these different media elements are normally decoded in different threads. To fix such drifts, regular re-synchronisation points can be included during the playback.

Listing 2: Accessing media fragments on a Media Fragments-enabled HTTP server.

```
# HTTP request
GET /video_es.ogv HTTP/1.1
Host: www.example.com
Accept: video/*
Range: t:npt=25-100&track='audio'

# HTTP response
HTTP/1.1 206 Partial Content
Accept-Ranges: bytes, t, track
Content-Length: 3743
Content-Type: video/ogg
Content-Range: bytes 19147-22880/35614993
Content-Range: t:npt 24.85-100.34/653.791
Content-Range: track audio,subtitle/653.791

{binary data}
```

Listing 3: Excerpt of HTML5 and Javascript code for synchronizing media resources.

```
<div id="tracks">
  <video class="v" src="video.ogv#t=25,100&track='
       video'" type="video/ogg" />
  <audio class="a" src="video_es.ogv#t=25,100&track
       ='audio'" type="audio/ogg"/>
</div>
<script type="text/javascript">
  // move all current playback times to same time
  function playTime(time) {
    tracks = jQuery("div#tracks").children();
    for (i=0; i<tracks.length; i++) {
      tracks[i].currentTime = time;
    }
  }

  // play/pause toggle
  function playpause() {
    // resynch tracks ...
  }
</script>
```

An example of a server implementing the media fragment to bytes mapping is NinSuna[2]. NinSuna is a fully integrated platform for multimedia content selection and adaptation. Its design and the implementation thereof are based on Semantic Web technologies. Furthermore, a tight coupling exists between NinSuna's design and a model for describing

[2]http://ninsuna.elis.ugent.be/

structural, semantic, and scalability information of media resources. Media resources are ingested into the platform and mapped to this model. The adaptation and selection operations are based on this model and are thus independent on the underlying media formats, making NinSuna a format-independent media delivery platform [10]. The platform is currently able to perform track and time range selection and supports therefore the Media Fragment-specific HTTP request discussed above.

4. DISCUSSION

In this section, we would like to point out a number of discussion points related to the implementation of the Media Fragments specification.

Currently, it is not clear for all media fragment axes, how media fragments should be rendered and experienced by the end-user in a meaningful way. For instance, temporal fragments could be highlighted on the seekbar; spatial fragments could be emphasized by means of bounding boxes or they could be played back in colour while the background is played in grayscale. Finally, different tracks could be selected using dropdown boxes or buttons. Whether media fragments URIs are hidden from the end-user or not depends on the application.

Another point of discussion is how UAs are able to discover the available named and track fragments. More specifically, there is currently no standardized way to discover these names. One possibility is to use the Continuous Media Markup Language (CMML, [6]), which allows to annotate and index continuous media files. This way, a UA requests a CMML description of a media resource to obtain the available named and track fragments. Another possibility is to use the Media Multitrack API[3] developed within the HTML5 Working Group. This proposal is a JavaScript API for HTML5 media elements that allows Web authors to determine the data that is available from a media resource. It exposes the tracks that the resource contains, the type of data it is (e.g. audio/vorbis, text/srt, video/theora), the role this data plays (e.g. audio description, caption, sign language), and the actual language it is in (RFC3066 language code). It also enables control over the activation state of the track.

Finally, existing Web proxies used to cache and speed up the delivery of Web content. However, they have no means of caching Media Fragment URI Range requests as illustrated in section 3, since they only understand byte ranges. One way to solve this issue is to develop Web proxies that are aware of Media Fragment URIs. Another possibility is to implement a so-called two-ways handshake in which a first Range request expressed in a custom unit is issued from the UA, for which the server answers with just a header containing the correspondence of this unit into bytes and a specific header telling the UA to issue another Range request, this time expressed in bytes and can therefore be cached [8].

5. CONCLUSION AND FUTURE WORK

In this paper, we discussed how parts of the W3C Media Fragments 1.0 specification can be implemented. We discuss this implementation using two simple scenarios: one scenario where the UA is smart enough to resolve the media fragment on its own and one scenario where the UA gets help from a Media Fragments-aware Web server. Additionally, we identified a number of discussion points regarding the implementation of media fragments that need to be solved in the near future, such as how to render media fragments in UA, how to discover named and track fragments, and how to cache media fragments.

[3] `http://www.w3.org/WAI/PF/HTML/wiki/Media_MultitrackAPI`

6. ACKNOWLEDGMENTS

This paper was supported by the French Ministry of Industry (*Innovative Web* call) under contract 09.2.93.0966, "Collaborative Annotation for Video Accessibility" (ACAV), Ghent University, the Interdisciplinary Institute for Broadband Technology (IBBT), the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research-Flanders (FWO-Flanders), and the European Union.

7. REFERENCES

[1] M. Hausenblas, R. Troncy, Y. Raimond, and T. Bürger. Interlinking Multimedia: How to Apply Linked Data Principles to Multimedia Fragments. In *2^{nd} Workshop on Linked Data on the Web (LDOW'09)*, Madrid, Spain, 2009.

[2] ISO/IEC. Information technology – Coding of Audio, Picture, Multimedia and Hypermedia Information – Part 14: MP4 file format. ISO/IEC 14496-14:2003, December 2003.

[3] S. Pfeiffer. `http://blog.gingertech.net/2010/02/12/audio-track-accessibility-for-html5/`.

[4] S. Pfeiffer. The Ogg Encapsulation Format Version 0. RFC 3533, `http://www.ietf.org/rfc/rfc3533.txt`, 2003.

[5] S. Pfeiffer and C. Parker. Accessibility for the HTML5 `<video>` element. In *6^{th} International cross-disciplinary conference on Web accessibility (W4A'09)*, pages 98–100, Madrid, Spain, 2009.

[6] S. Pfeiffer, C. Parker, and A. Pang. The Continuous Media Markup Language (CMML), Version 2.1. `http://www.annodex.net/TR/draft-pfeiffer-cmml-03.html`, 2006.

[7] R. Troncy, L. Hardman, J. van Ossenbruggen, and M. Hausenblas. Identifying Spatial and Temporal Media Fragments on the Web. W3C Video on the Web Workshop, 2007.

[8] R. Troncy and E. Mannens, editors. *Media Fragments URI 1.0*. W3C Working Draft. World Wide Web Consortium, December 2009.

[9] R. Troncy and E. Mannens, editors. *Use cases and requirements for Media Fragments*. W3C Working Draft. World Wide Web Consortium, November 2009.

[10] D. van Deursen, W. V. Lancker, W. D. Neve, T. Paridaens, E. Mannens, and R. V. de Walle. NinSuna: a Fully Integrated Platform for Format-independent Multimedia Content Adaptation and Delivery based on Semantic Web Technologies. *Multimedia Tools and Applications – Special Issue on Data Semantics for Multimedia Systems*, 46(2-3):371–398, January 2010.