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 examples of implementations of the Media Fragments URI specification which is currently being developed by the W3C Media Fragments Working Group. The group's mission is to create standard addressing schemes for media fragments on the Web using Uniform Resource Identifiers (URIs). We describe two scenarios to illustrate the implementations. 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, Hyperlinks, Web, URI, HTTP

Keywords

Media Fragments, Video Accessibility, Video URL, media delivery, media servers

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 already support the new `<video>` and `<audio>` elements. However, to make media a "first class citizen" on the Web, it needs to be as easily linkable as HTML pages. Only when we can link into media resources, we will really be able to share the important parts of a media resource. Only when we are able to dynamically choose the tracks that are enabled in a media resource, we will really be able to cater for accessibility needs for media resources. Only when we are able to navigate through media resources based on semantics rather than random guesswork, we will really be able to master the full complexity of rich media [4, 5].

The mission of the W3C Media Fragments Working Group[1] (MFWG), which is part of W3C's Video in the Web activity, is to address media fragments on the Web using Uniform Resource Identifiers (URIs) [1]. Following a requirement phase [7], 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) [6]. Furthermore, media fragments can be identified by name, which is a semantic replacement for addressing any range along 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), José, who is living in Argentina, wants to watch a live stream coming from the United Nations in New York. Further, the corresponding Media Fragment URI is adjusted according to the fact that José only speaks Spanish and is only interested in a specific discussion. We first show the different possibilities to implement the Media Fragments specification based on these two scenarios (sections 2 and 3). We then identify 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 a Facebook notification on her smart phone for a movie in which a friend has pointed out a specific scene. She wants to watch the 75 seconds long scene with English audio track using the media fragment URI posted on her wall:

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

To be able to play this part of the video, without downloading the whole media resource, requires a UA that can interpret the URI, determine that it relates to a media resource, and request only the appropriate data for download. This is of particular importance since Alice is watching the video on a smart phone where the bandwidth cost for the full movie would kill her budget.

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

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

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}
```

Let's suppose Alice has a smart UA at her disposal that understands the Media Fragment URI and can resolve the temporal fragment identifier through a range request expressed in byte. 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 range(s). Alice's case requires that the UA understands how time is mapped to byte offsets for the underlying media resource format. Assuming we are using an HTML5 Ogg capable browser, then the browser will first set up the media resource at `http://example.com/video.ogv` for decoding, which means downloading the first couple of bytes of the file to collect the resource header information and set up the decoding pipelines for the contained tracks. Then, the UA would stop downloading the resource from the beginning, and resolve the fragment specification, which is possible now that it has confirmed the MIME type of the media resource. The process for MP4 files is similar with MP4 headers containing tables that provide a complete mapping of time to byte-offsets [2]. Recent Ogg files also have such an index, but for old Ogg files, temporal seeking is still possible by applying a bisection search algorithm over the Ogg pages in the file over the network until the correct byte ranges are retrieved [3].

2. Based on the found time-to-byte mapping, the UA sends one or more HTTP Range requests for the relevant 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. Apache is one such example server.

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

The advantage of processing media fragments within the UA is that media fragments can be served by a traditional HTTP Web server. It only requires an extension to the UA software for parsing the Media Fragment URI syntax. This UA extension needs to be aware of the syntax and semantics of media formats, but a UA will already need to understand media formats that it wants to decode and play back.

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

```
# HTTP request
GET /live_video.ogv HTTP/1.1
Host: www.example.com
Accept: video/*
Range: t:clock=2010-03-26T11:00:00Z-&track=video,
     audio_es


# HTTP response
HTTP/1.1 206 Partial Content
Acccept-Ranges: bytes, t, track
Content-Length: 15000
Content-Type: video/ogg
Content-Type: multipart/byteranges;boundary=BOUNDARY
Content-Range-Mapping: t:clock 2010-03-26T10:59:58Z
     -*/2010-03-26T09:00:00Z-*&track video,audio_es

--BOUNDARY
Content-Type: video/ogg
Content-Range: bytes 2000-13000/*
{binary data}
--BOUNDARY
Content-Type: video/ogg
Content-Range: bytes 24000-28000/*
{binary data}
```

Whether media fragment to byte range mapping is possible or not within the UA without having the full original media resource available, highly depends on the media format.

3. PROCESSING MEDIA FRAGMENTS WITH SERVER HELP

In scenario (b), José wants to watch a live stream coming from the United Nations in New York. Since he only speaks Spanish and is only interested in a specific discussion, he creates the following Media Fragment URI:

`http://example.com/live_video.ogv#t=clock:`
`2010-03-26T11:00:00Z&track=video,audio_es`

José has a UA that understands and parses media fragments, but prefers to ask a clever server to deliver the appropriate bytes rather than resolving the byte range mapping itself. When José starts to play the video, the following steps occur:

1. The UA parses the media fragment identifier and creates 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 [6].

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-Mapping headers describing the content range in terms of time and tracks. Finally, the UA receives, decodes, and starts playing back the requested media fragment.

An example of a server implementing the media fragment to bytes mapping is NinSuna[2]. NinSuna is a fully integrated

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

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 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 of the underlying media formats, making NinSuna a format-independent media delivery platform [8]. The platform is able to perform track and time range selection and supports 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 transport bar; 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 fragment 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 Rich Open multitrack media Exposition format (ROE[3]), which allows to express the track composition of a media resource, the names for the tracks and a restricted amount of metadata such as language, role and content-type, which can further help select tracks. Another possibility is to use the Media Multitrack API[4] 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 are 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 an indirection in which a first Range request expressed in a custom unit is issued from the UA, for which the server answers with just a HEAD reply 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 [6].

[3] `http://wiki.xiph.org/ROE`

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

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.

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), the European Union, and Mozilla Corporation.

7. REFERENCES

[1] M. Hausenblas, R. Troncy, Y. Raimond, and T. Bürger. Interlinking Multimedia: How to Apply Linked Data Principles to Multimedia Fragments. In *2nd 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. RFC 3533: "The Ogg Encapsulation Format Version 0," Available on `http://www.ietf.org/rfc/rfc3533.txt`.

[4] S. Pfeiffer. Architecture of a Video Web - Experience with Annodex. W3C Video on the Web Workshop, 2007.

[5] 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.

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

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

[8] D. Van Deursen, W. Van Lancker, W. De Neve, T. Paridaens, E. Mannens, and R. Van 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.