Implementing the Media Fragments URI Specification: Media Fragments Firefox Extension

Jakub Sendor <jakub.sendor@eurecom.fr>

Media Fragments

- Interesting scene from a movie
- Region of an image containing a specified person
- Portion of music (e.g. fancy guitar solo)
- Audio track from a video file

Media Fragments use cases

- Bookmarking parts of audio/video file
- Mash-ups
- Annotating media fragments
- Conserving bandwidth

Fragments Dimensions

- **Temporal: begin, end time**
- **Spatial: region of the video in terms of pixels/percents**
- **Track: track name**
- **Named: unique fragment id**

Media Fragments URIs

- **Using URI query part:**

```
http://www.example.org/video.ogv?t=60,100
```

- **Using URI fragment part:**

```
http://www.example.org/video.ogv#t=60,100
```

- **Mixing both:**

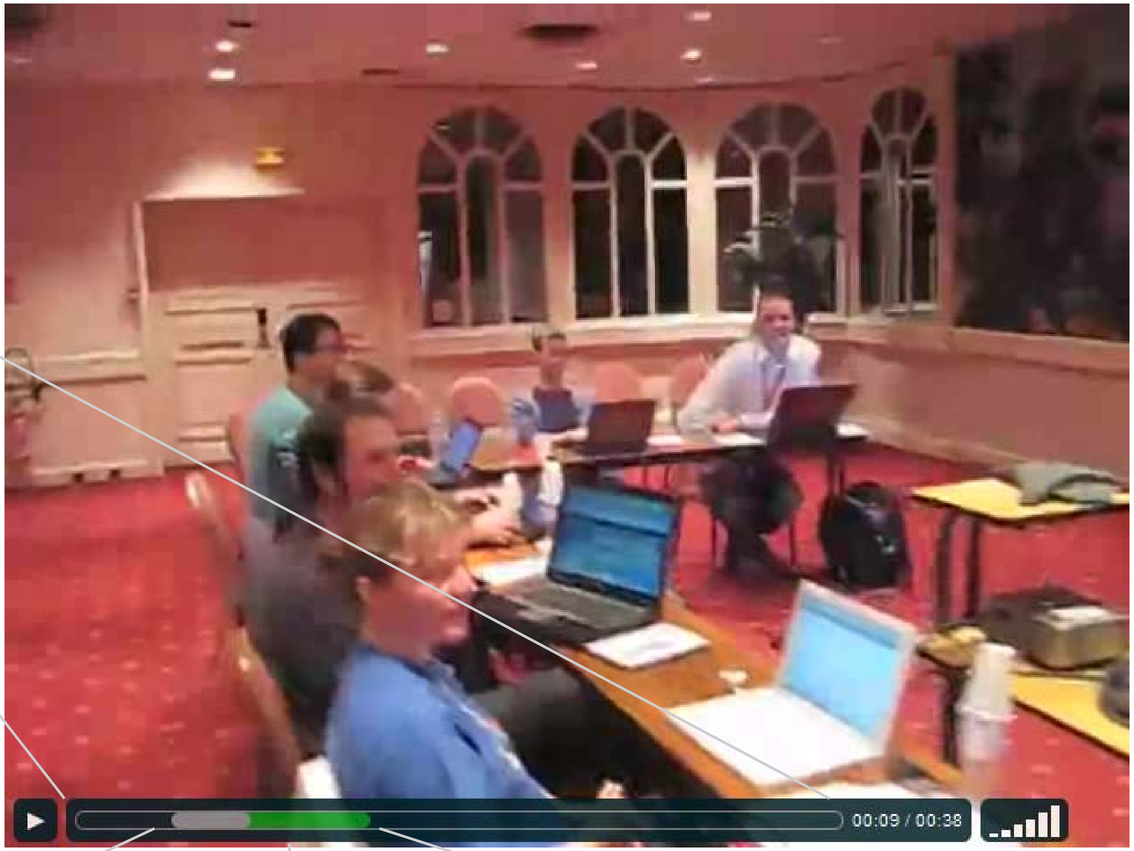
```
http://www.example.org/video.ogv?t=60,100#t=10,15
```

Media Fragments Resolution

- **For the URI query part:**
  - media file processed only on server side
  - client receives a new video file
- **For the URI fragment part:**
  - client analyzes URI fragment
  - request to the server is enhanced with proper Range header value
  - server sends corresponding byte ranges to the client

Media Fragments Extension (temporal)

Original resource length



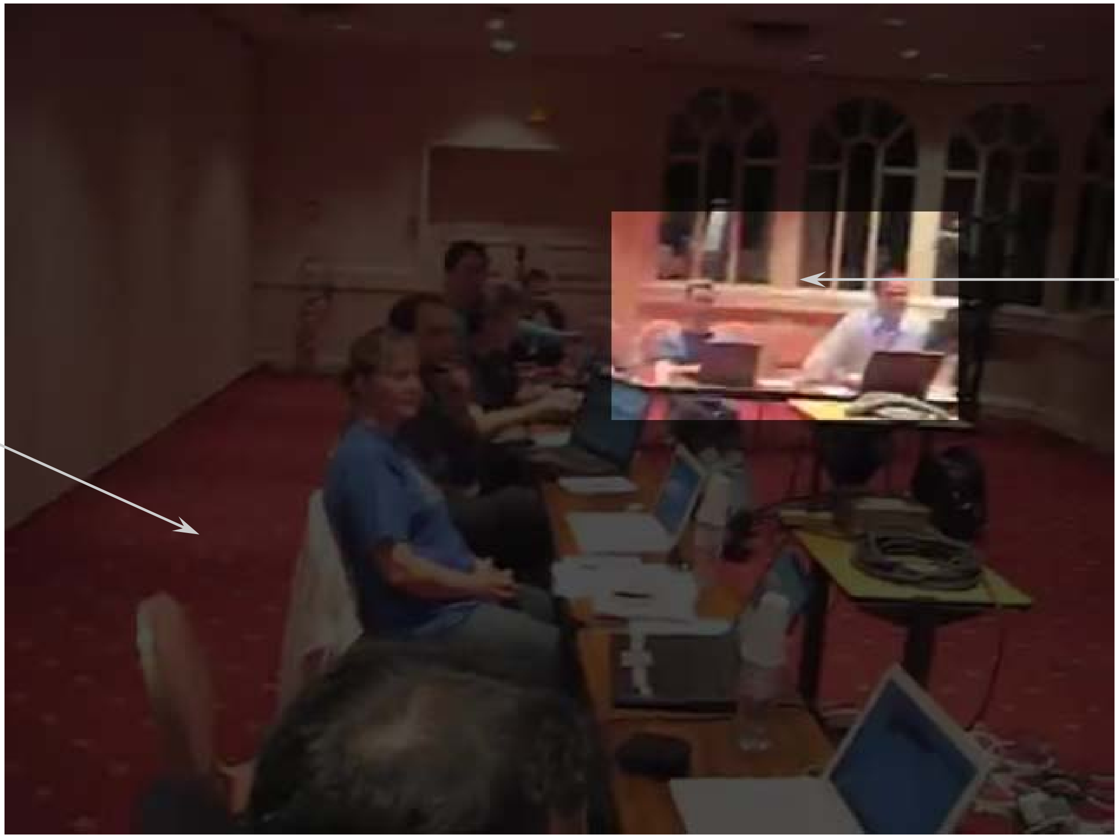
Fragment beginning

Playback progress

Fragment end

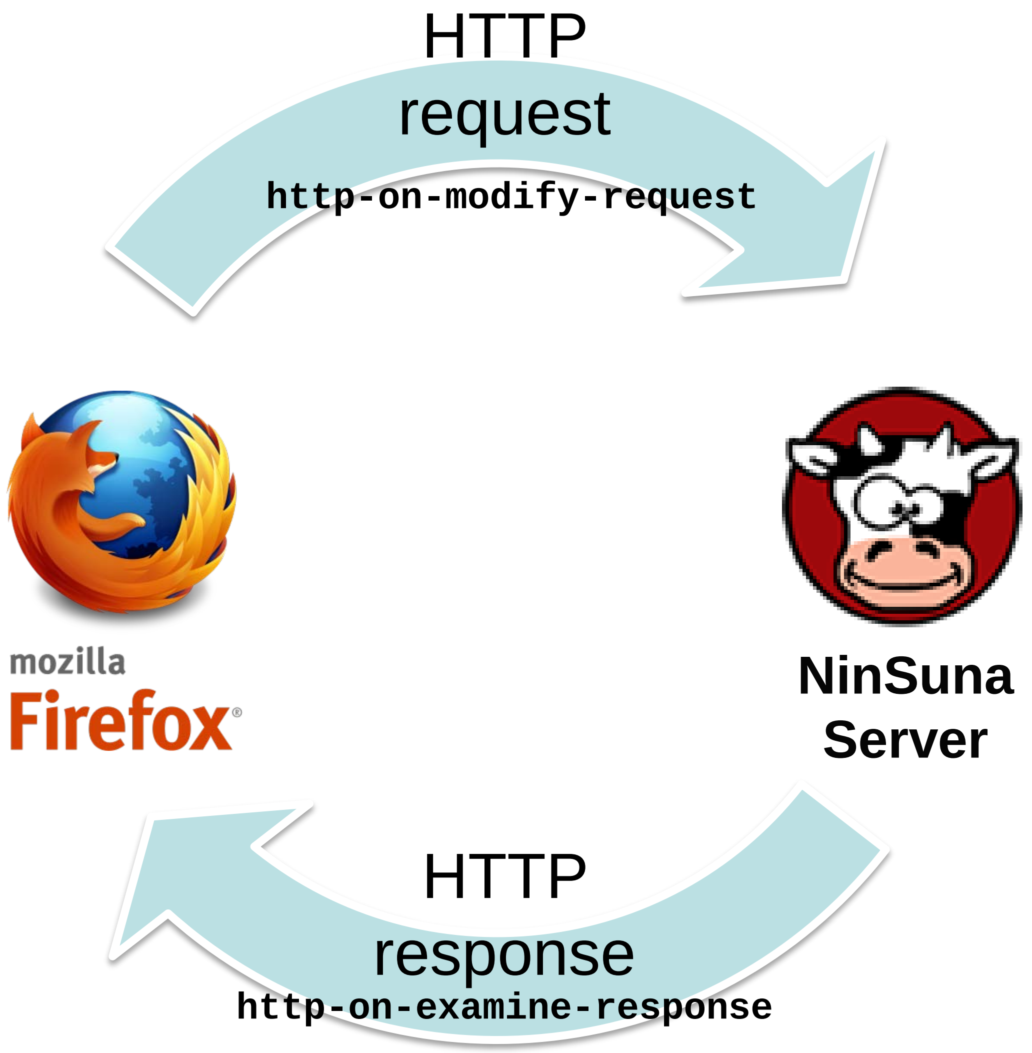
Media Fragments Extension (spatial)

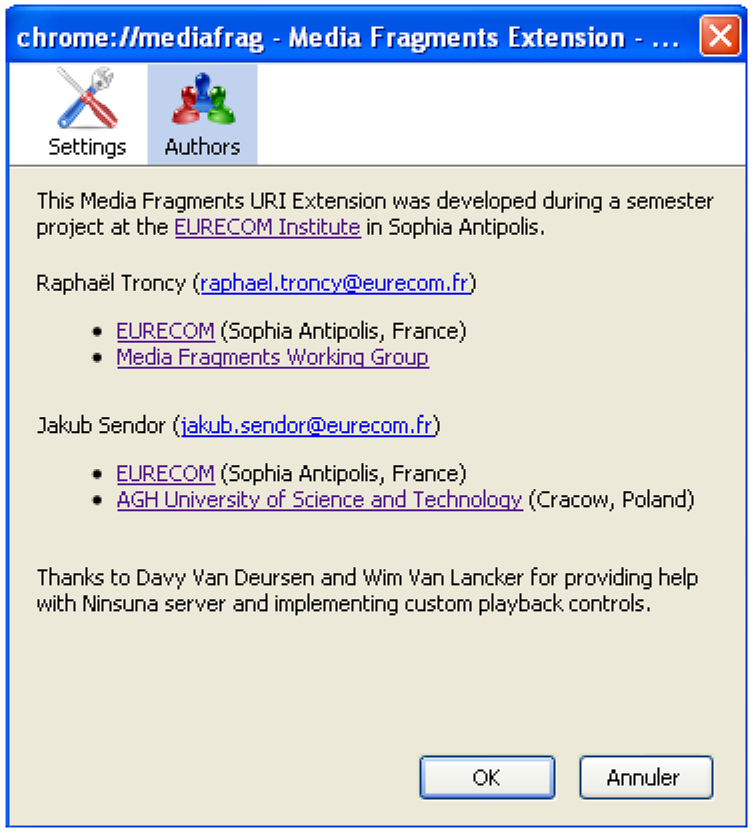
semi-opaque overlay



highlighted fragment

Observing HTTP Traffic





This Media Fragments URI Extension was developed during a semester project at the EURECOM Institute in Sophia Antipolis.

Raphaël Troncy (raphael.troncy@eurecom.fr)

- EURECOM (Sophia Antipolis, France)
- Media Fragments Working Group

Jakub Sendor (jakub.sendor@eurecom.fr)

- EURECOM (Sophia Antipolis, France)
- AGH University of Science and Technology (Cracow, Poland)

Thanks to Davy Van Deursen and Wim Van Lancker for providing help with Ninsuna server and implementing custom playback controls.

Examining HTTP Traffic

- **HTTP request:**
  - retrieving URI
  - parsing key=values pairs from fragment part
  - setting Range header
- **HTTP response:**
  - checking Content-Type
    and Content-Range-Mapping headers values
  - attaching custom playback controls to page
  - creating spatial dimension overlay (if specified)

Example: requesting time fragment

- **A web developer specifies a video source with a temporal fragment URI:**

```
http://ninsuna.elis.ugent.be/DownloadServlet/mfwg/fragf2f.ogv#t=5,15
```

- **key=value pair is analyzed, fragment begin and end time are matched**

```
t=5,15
```

- **Media Fragments Extension analyses the fragment part, retrieves beginning and end time and sets proper Range header value:**

```
Range: t:npt=5-15
```

Example: requesting time fragment

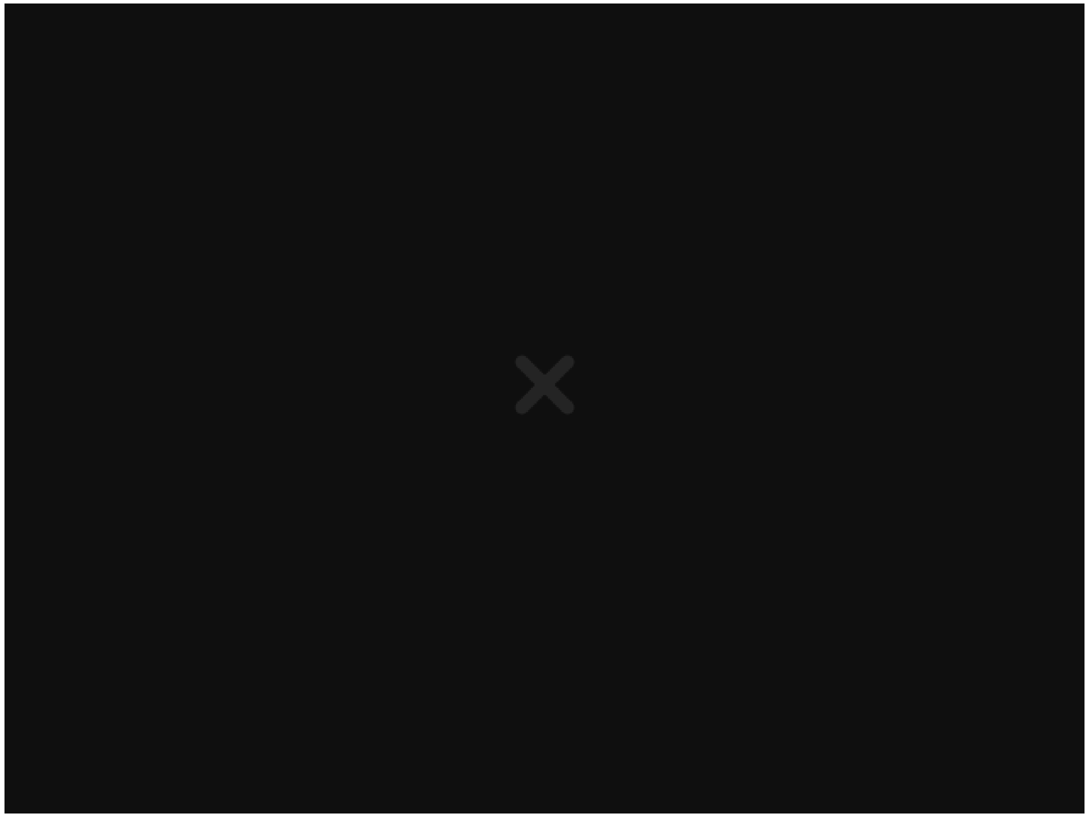
- **The NinSuna server responds with the 206 Partial Content response and Content-Range-Mapping header showing the mapped time ranges and media fragment in the message payload:**

```
HTTP/1.1 206 Partial Content

Content-Type: video/ogg
Accept-Ranges: bytes, t, track, id
Content-Range: bytes 629578-1690588/4055466
Content-Range-Mapping:
{t:npt 4.8-14.8/0-38.33}={bytes 629578-690588/4055466}
```

Example: requesting time fragment

- ... and it won’t work!
- Because the player does not have the media file header



Example: requesting time fragment

- **Player needs to be initialized, thus bare fragment is not playable**
- **To fix this, we add *;include-setup* to the Range header value:**

```
Range: t:npt=5-15;include-setup
```

Example: requesting time fragment

- The response from the server is slightly different:

```
HTTP/1.1 206 Partial Content

Content-Type: multipart/byteranges;boundary=End
Content-Range-Mapping:
 {t:npt 4.8-14.8/0-38.33;include-setup}
={bytes 0-5998,629578-1690588/4055466}
...

--End
Content-Type: video/ogg
Content-Range: bytes 0-5997/4055466

{binary data}
--End
Content-Type: video/ogg
Content-Range: bytes 629578-1690588/4055466

{binary data}
--End
--End--
```

Example: requesting time fragment

- **This is why we need to attach a Stream Listener to the HTTP channel:**

```
HTTP/1.1 206 Partial Content

Content-Type:
multipart/byteranges;boundary=End
...

--End
Content-Type: video/ogg
Content-Range: bytes 0-5997/4055466

{binary data}
--End
Content-Type: video/ogg
Content-Range: bytes 629578-
1690588/4055466

{binary data}
--End
--End--
```

```
HTTP/1.1 206 Partial Content

Content-Type: video/ogg
...

{binary data}
{binary data}
```

Spatial fragments

- Without additional information send to server
- Overlay is created by appending four additional DIVs to the webpage
- They are styled to create the impression of semi-opaque layer over the video element

Current Issues

- **Modifying a video element that is not embedded into a webpage is not possible**
  - Fragment will be requested and retrieved properly
  - Attaching custom playback controls is not possible
- **Track dimension:**
  - Fragments are not consecutive bytes ranges
  - Firefox is not digesting this
- **Media Fragments Non-Aware servers:**
  - Some servers that do not understand the Range header send sometimes error
  - To be implemented: resending request without Range header and adjusting playback controls on client side

Future Work

- **Aim to cover 100% of specification:**
  - error behaviors and graceful degradation
- **Adding more functionality to the custom playback controls:**
  - seeking outside fragment
  - “on the fly” update of spatial dimension overlay
- **Media Fragments proxy support:**
  - all videos will be available as fragments!
- **Publishing first stable version:**
  - Mozilla Add-ons, automatic updates