

360° VIDEO CLOUD STREAMING & HTMLVIDEOELEMENT EXTENSIONS

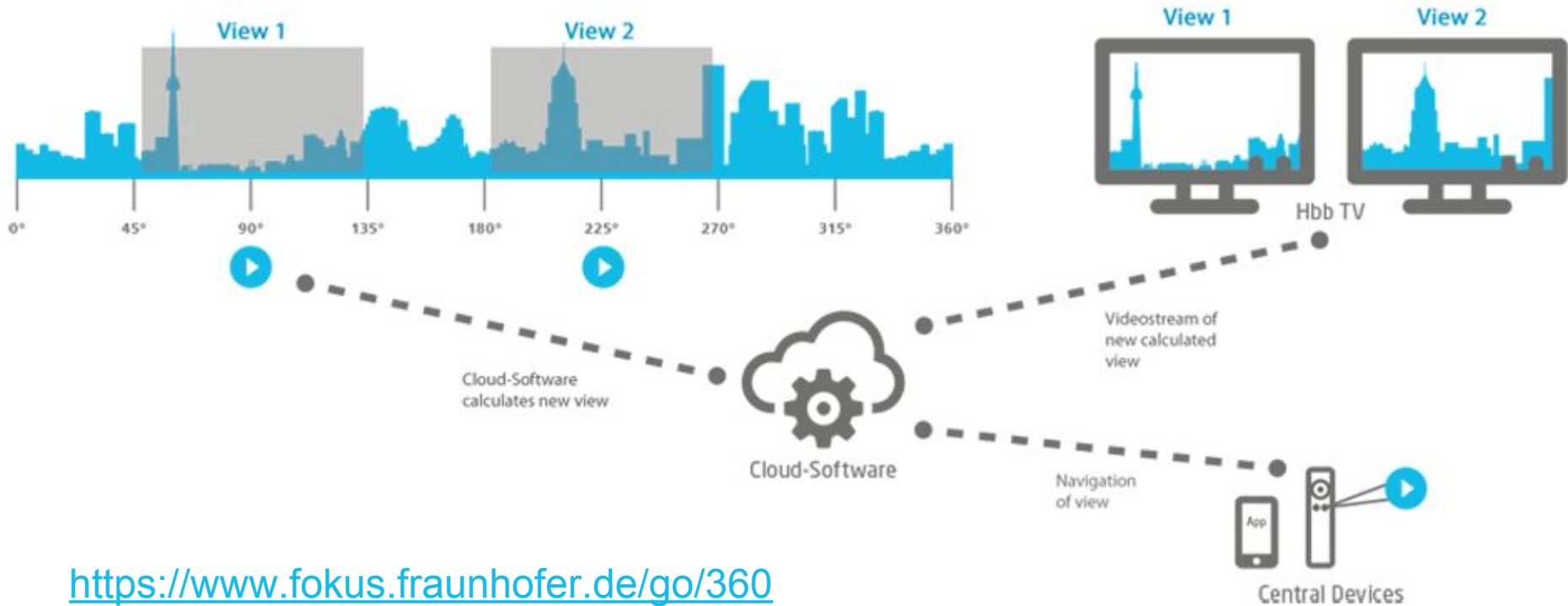


Louay Bassbouss | Fraunhofer FOKUS

W3C Workshop on Web & Virtual Reality, October 19-20, 2016; San Jose, CA, USA

360° video cloud streaming

360° Video on HbbTV devices



<https://www.fokus.fraunhofer.de/go/360>

360° STREAMING AND VIDEO PROCESSING OPTIONS

Server

Option1



Streaming

Client

user input

360° Processing



Video Playback

Option2



user input

360° Processing

Streaming



Video Playback

Option3a



360° Pre-Processing



user input

Prepare Video

Streaming



Video Playback

360° STREAMING AND VIDEO PROCESSING OPTIONS

Server

Option1



Streaming

Client

user input

360° Processing



Video Playback

Option2



user input

360° Processing

Streaming

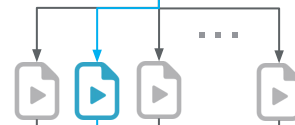


Video Playback

Option3b



360° Pre-Processing



Streaming

user input

Prepare Stream



Video Playback

ADVANTAGES AND DISADVANTAGES

	Option1	Option2	Option3a	Option3b
Additional Storage	No	No	Yes	Yes
360° Video Processing on Client	Yes	No	No	No
360° Video Processing on Server	No	Yes	No ¹	No ¹
Bandwidth	High	Low	Low	Low ²
Motion-to-Photon Delay	Low	Medium ³	Medium ³	Medium ⁴
CDN usage	Yes	No ⁵	No ⁵	Yes
Example Target Devices	Head Mounted Displays	Low Capability Devices e.g. HbbTV	Low Capability Devices e.g. HbbTV	Medium Capability Devices e.g. Chromecast
Interaction Types	<ul style="list-style-type: none"> - Motion Sensors - Touch/Mouse 	<ul style="list-style-type: none"> - TV RC - Keyboard - (Touch/Mouse) 	<ul style="list-style-type: none"> - TV RC - Keyboard - (Touch/Mouse) 	<ul style="list-style-type: none"> - TV RC - Keyboard - Touch/Mouse

Notes for previous slide

- 1) The original 360° video will be pre-processed and FOVs are stored in separate files. There will be an overlap between the FOVs this is why there is a need for more storage but on the other side no video processing is needed.
- 2) since only one FOV is streamed to the client, no additional bandwidth is needed comparing to traditional video streaming. But it is still possible to pre-cache neighboring FOVs e.g. in lower quality to enable fast switch between FOVs in this case additional bandwidth is needed
- 3) Motion to Photon delay depends on network latency and protocol used to stream a single FOV (and Buffering on the Player).
- 4) Motion to Photon Delay depends on the caching strategy of the player.
- 5) it is difficult to use CDN since a persistent connection between client and server is needed (there is a session for each client)

DEMONSTRATION (Option 3b)

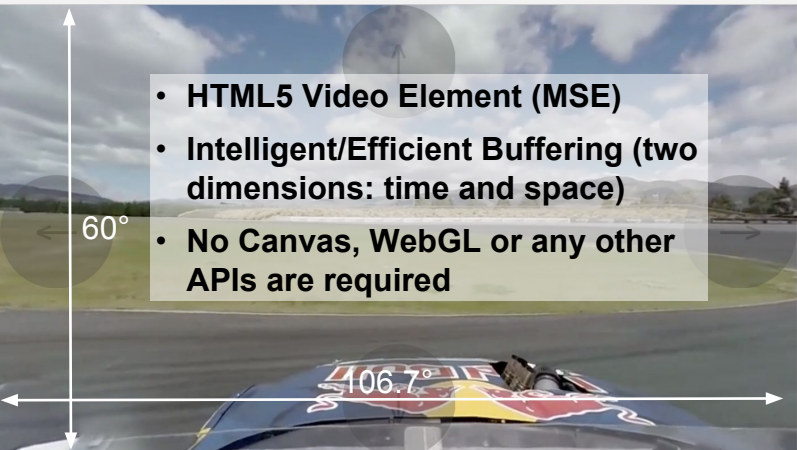
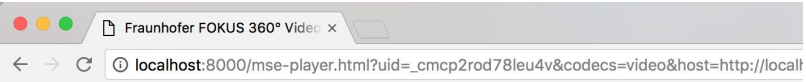


360°
Pre-Processing



4k origin 360° Video, 30fps, bitrate 40053 kb/s

HD view port, 30fps, bitrate 2435 kb/s, segment=333ms



- HTML5 Video Element (MSE)
- Intelligent/Efficient Buffering (two dimensions: time and space)
- No Canvas, WebGL or any other APIs are required



Prepare Stream
(Caching)

Demo video

The screenshot displays a video player interface. The main video area shows a first-person perspective from a car driving on a road. The control panel on the right includes buttons for 'Pause', 'Play', 'Fullscreen', and 'Toggle U-UI'. Below these are 'In-App Buffer' and 'Video Stats' sections. The 'Video Stats' section shows 'Current Time: 0:05', 'Duration: 7:33', 'D-R: 0.08', and 'Playback Rate: 1.00x (normal)'. The 'View Port' section shows 'P: 17°', 'Theta: 0°', 'Width: 1920°', and 'Height: 1080°'. At the bottom, a network packet capture table is visible, showing a list of packets with columns for 'No.', 'Time', 'Type', 'Length', 'Info', and 'Load in Bytes'. The table contains 10 rows of data, all with 'Type' as 'HTTP' and 'Length' as 2048.

No.	Time	Type	Length	Info	Load in Bytes
1	0.000000	HTTP	2048	GET / HTTP/1.1	0
2	0.000000	HTTP	2048	GET / HTTP/1.1	0
3	0.000000	HTTP	2048	GET / HTTP/1.1	0
4	0.000000	HTTP	2048	GET / HTTP/1.1	0
5	0.000000	HTTP	2048	GET / HTTP/1.1	0
6	0.000000	HTTP	2048	GET / HTTP/1.1	0
7	0.000000	HTTP	2048	GET / HTTP/1.1	0
8	0.000000	HTTP	2048	GET / HTTP/1.1	0
9	0.000000	HTTP	2048	GET / HTTP/1.1	0
10	0.000000	HTTP	2048	GET / HTTP/1.1	0

- Two types of players:
 - Native 360° Video Player
 - Using MSE → do we need extensions for the MSE API?
- Native 360° Video Player:
 - The HTMLVideoElement plays 360° video natively. Set `video.src={360_video_url}` (or use `<source element>`)
 - The HTMLVideoElement needs to get all the metadata in order to render the view correctly.
 - New functions to set and get the FOV are needed
 - New events on start, during and after changing the FOV are needed
 - (Maybe also functions and events for Zoom in/out.)
 - Example:
 - `video.setFOV(phi, theta, width, height)`
 - `video.onfovstart`, `video.onfovend`, ...

- MSE 360° Player
 - Allows to implement different player algorithms similar for DASH on top of MSE
 - Available viewports can be described in a manifest (e.g. DASH SRD fields)
 - At the start of the playback the currently selected viewport is buffered. When the user triggers a switch request for a different viewport, already buffered segments are removed/replaced by segments of the new viewport.
 - Challenge:
 - How to reduce delay by switching between two viewports?

