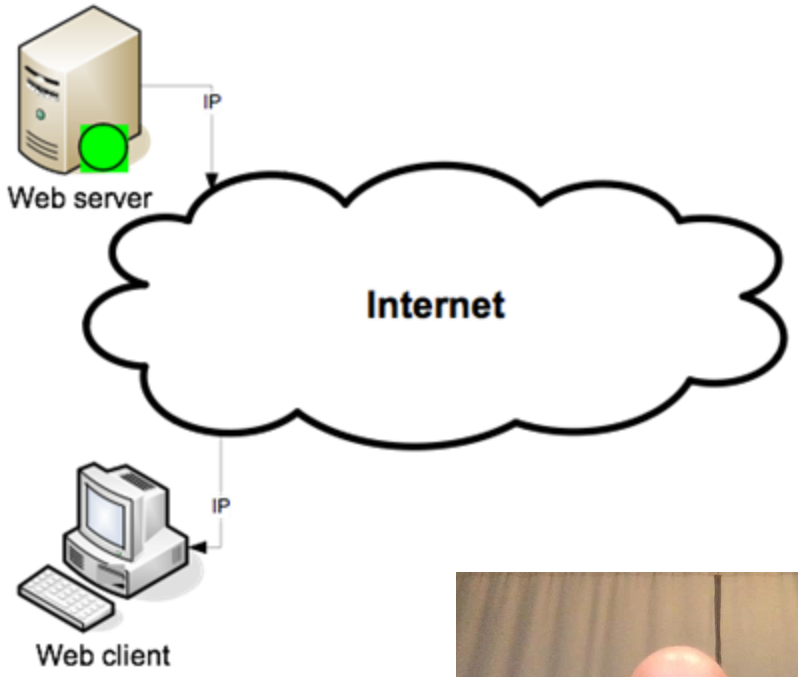
Connection-Oriented Streaming of Multimedia Content

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
AGH University of Science and Technology

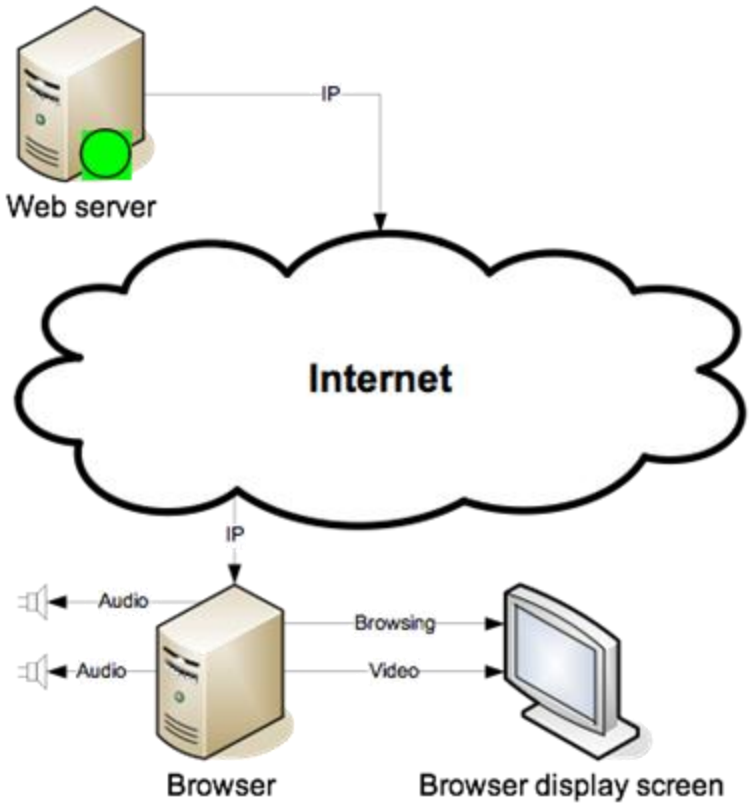
Mikołaj Leszczuk (AGH UST), Rafał Myszka (Akamai),
Jakub Nawała (AGH UST)

Why Connectionless Streaming Is Bad?

- Running out of **IP** addresses
- Network Address Translation (**NAT**)
- No **RTP/UDP** easily possible then… ☹
- But **NAT** usually OK with **Web** traffic! ☺

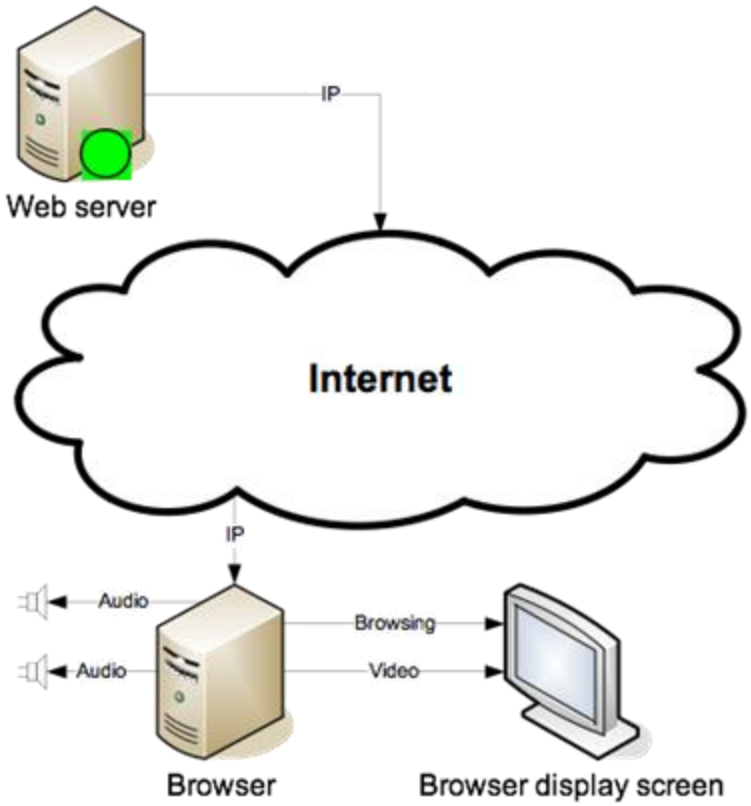


Why Connection-Oriented Streaming Is Good?



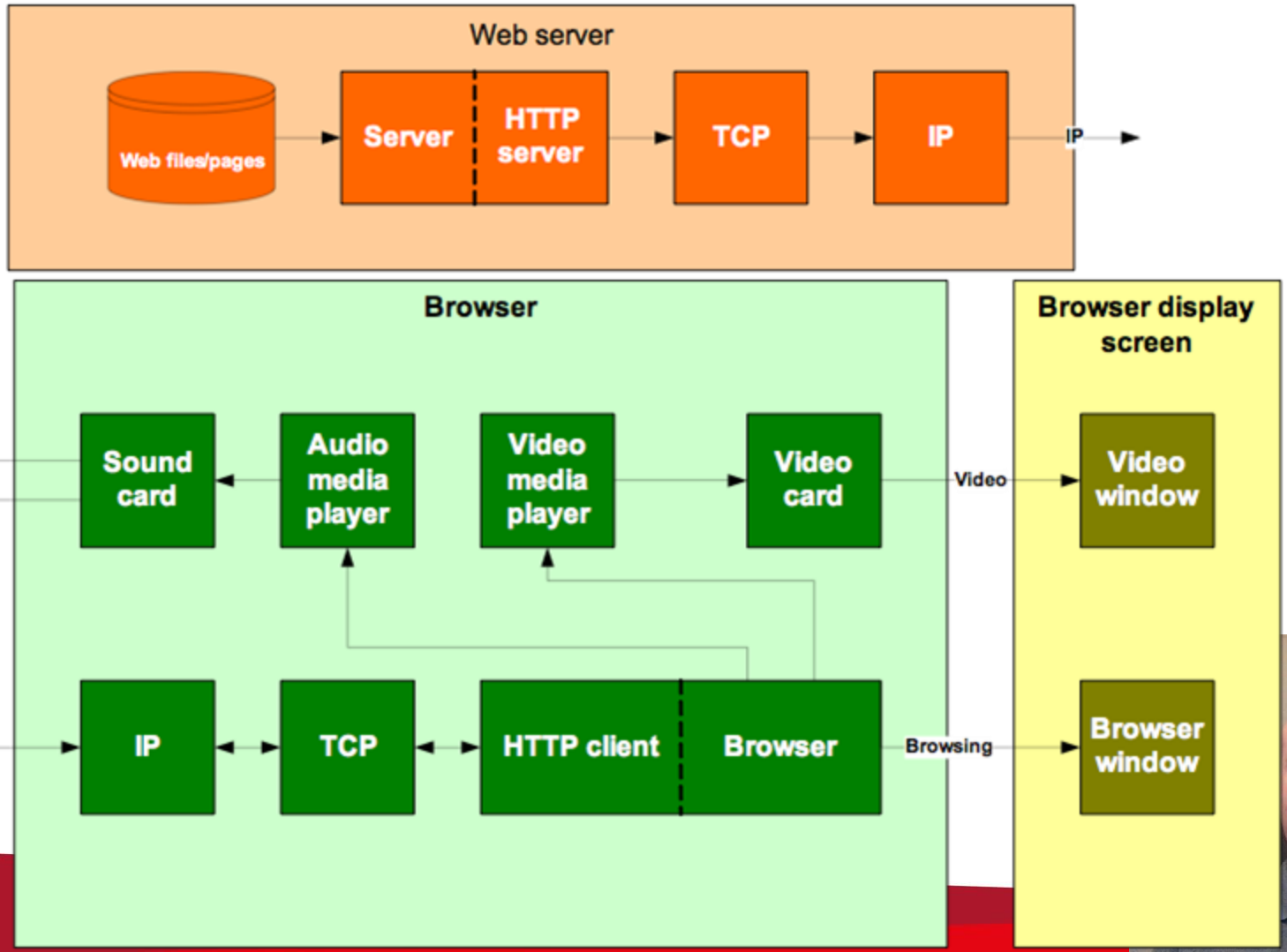
- **Web** means **HTTP/TCP/IP**
- Let's encapsulate streaming as **Web**
- Progressive Media Download (**PMD**)
- Called as "streaming" even if technically not streaming
- What are technical details?

What is **PMD**?

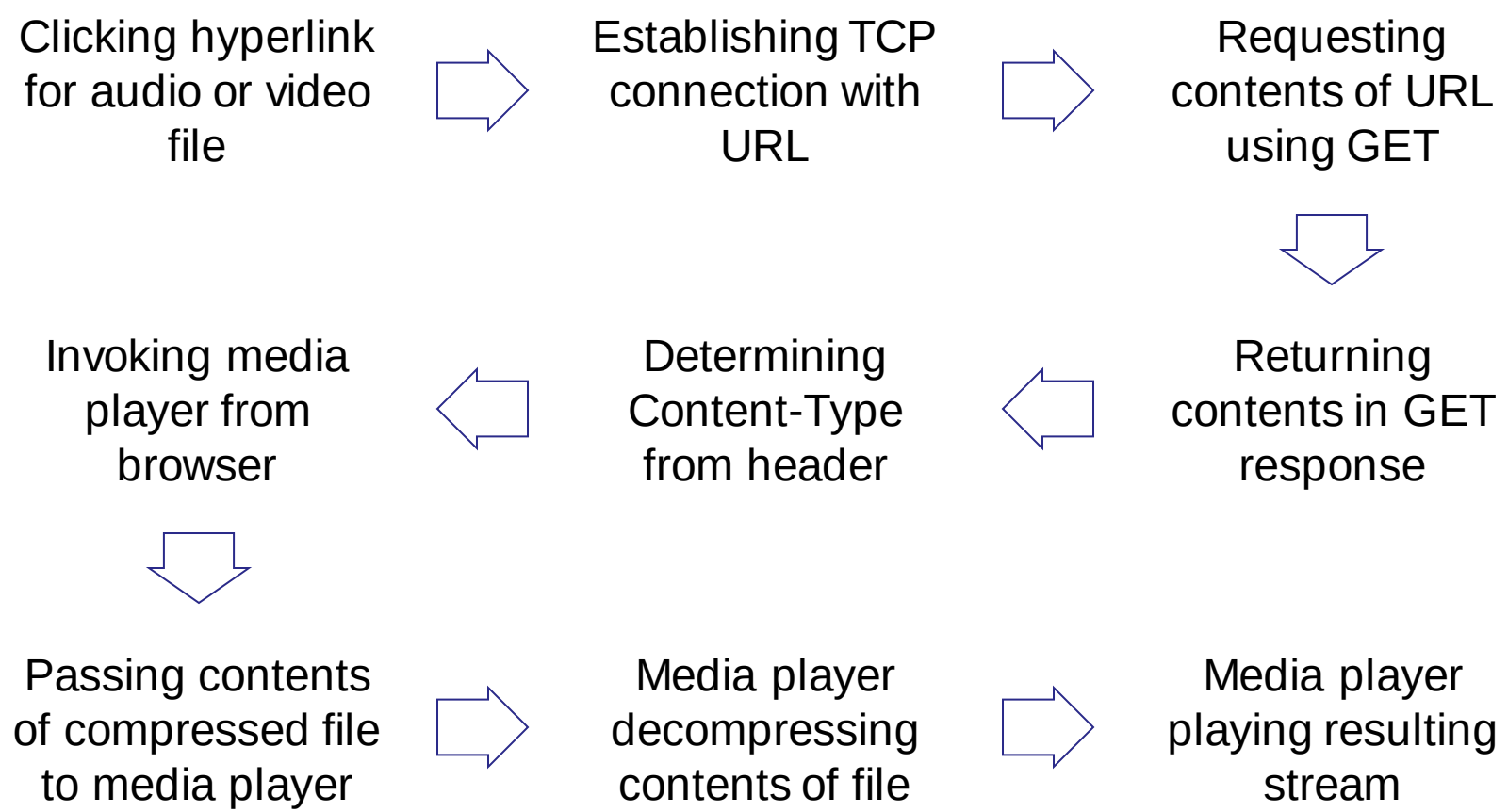


- How do files are accessed?
- Can plain **Web** server be used?
- Is **A/V** accessed the same way?

Components of **PMD**



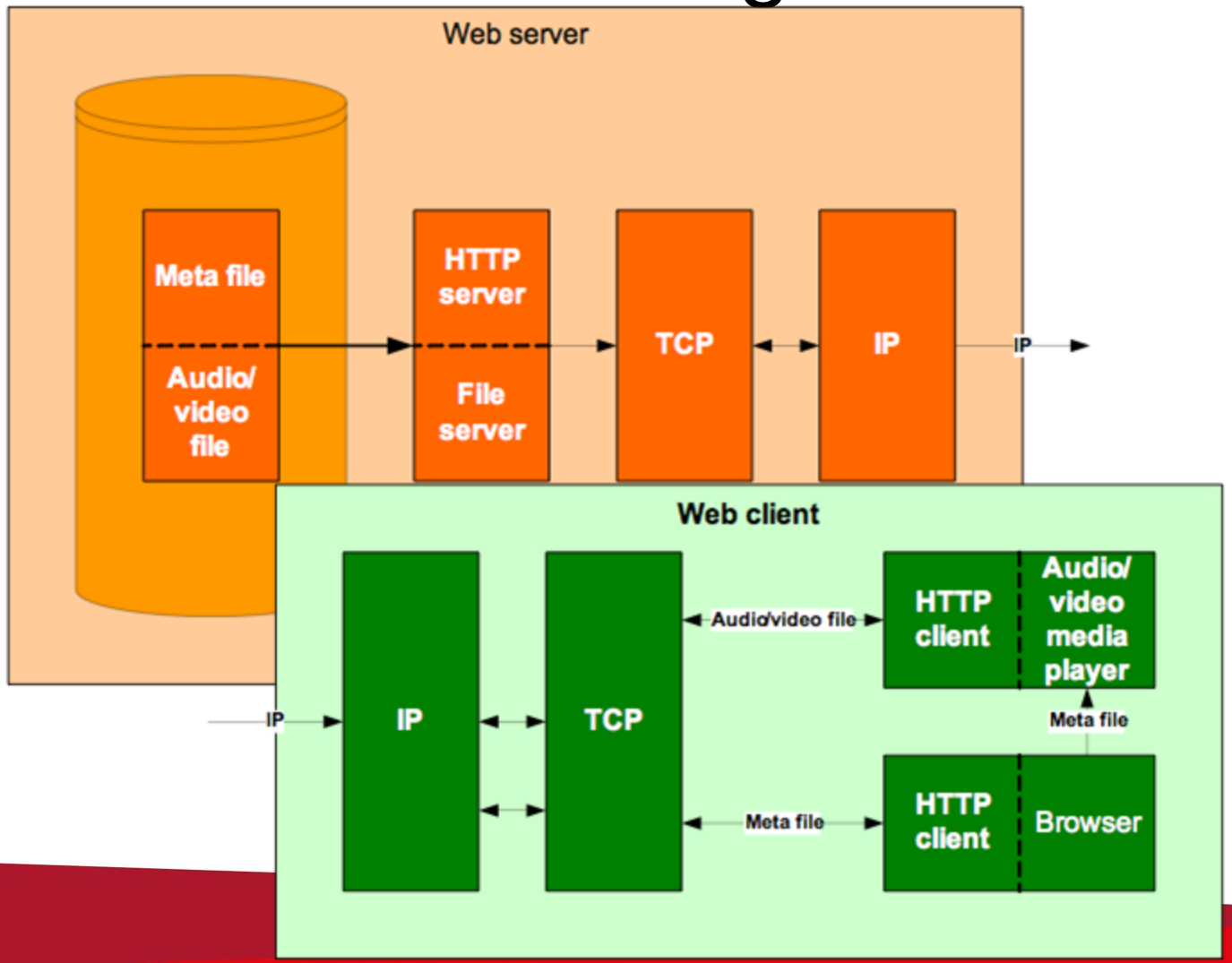
HTTP Process for **PMD**



Problem. And Solution

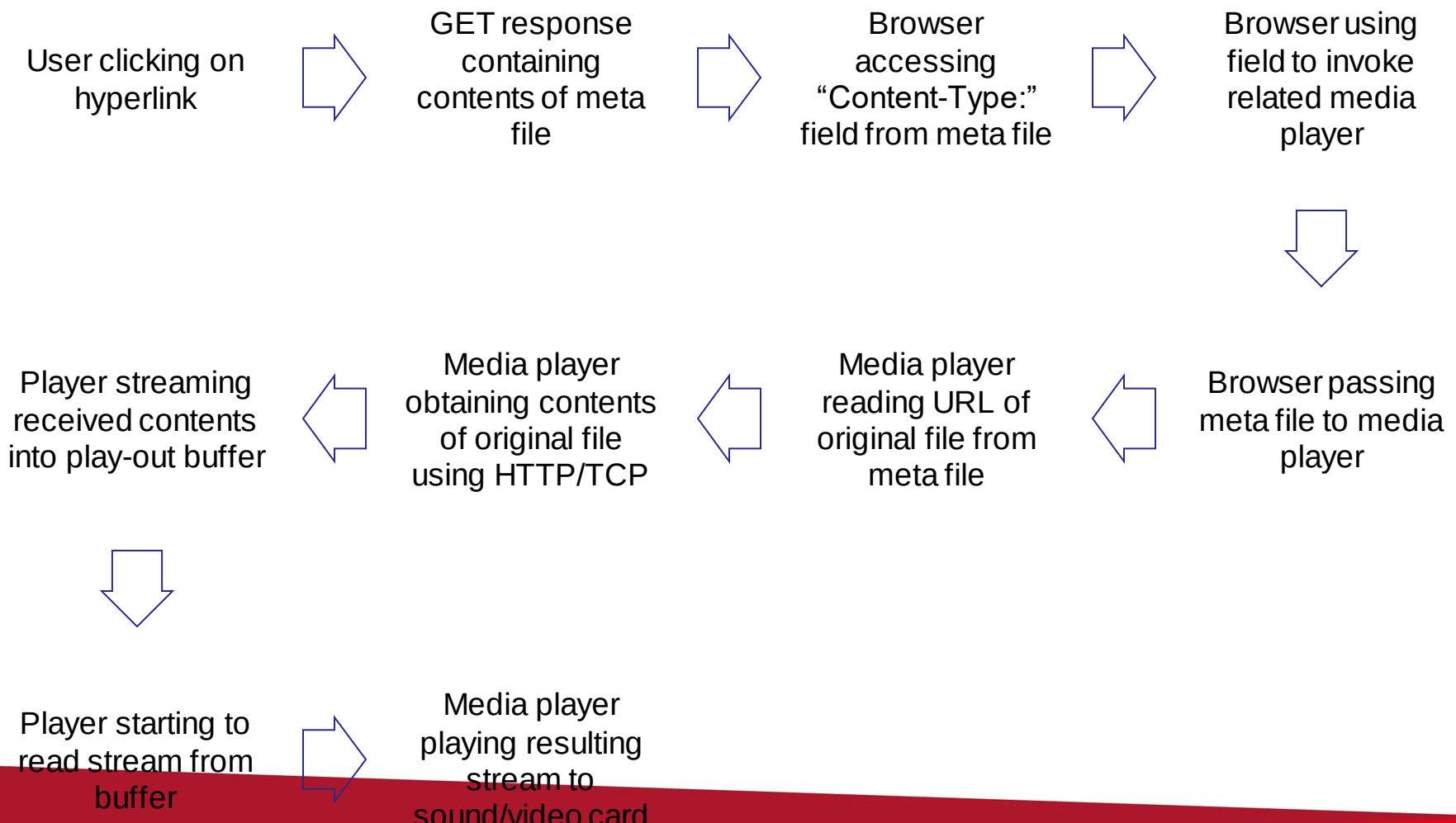
- First: browser downloading entire media file
- Only then: passing it to media player
- **Long delay** if significant file size
- What if media player downloaded media file by itself?
- **Introducing “meta file”**

Components of **PMD** using Meta File



HTTP

Process for **PMD** using Meta File



Header Information Location

End of file (AVI format)

- **First download, then play**
- **Playback start by**: total download time
- **Min bandwidth**: no (full pre-buffering)

Beginning of file (streaming formats)

- **Simultaneous download and play**
- **Playback start by**: bandwidth statistics
- **Min bandwidth**: yes (smooth playback)

PMD Players

Browser

» Opening media file **URL** by browser

» Caching internally by browser

» Media file played directly by browser

Plug-in

» Redirecting media file **URL** to plug-in

» Caching externally by plug-in

» Media file played indirectly by plug-in

Pros and Cons of **PMD**

Pros

Clients can start the playback before the whole file gets downloaded

Part of the existing infrastructure

No configuration required

Cons

No dynamic flow control

» Media stops to play when playback rate exceeds download rate

No interactive streaming

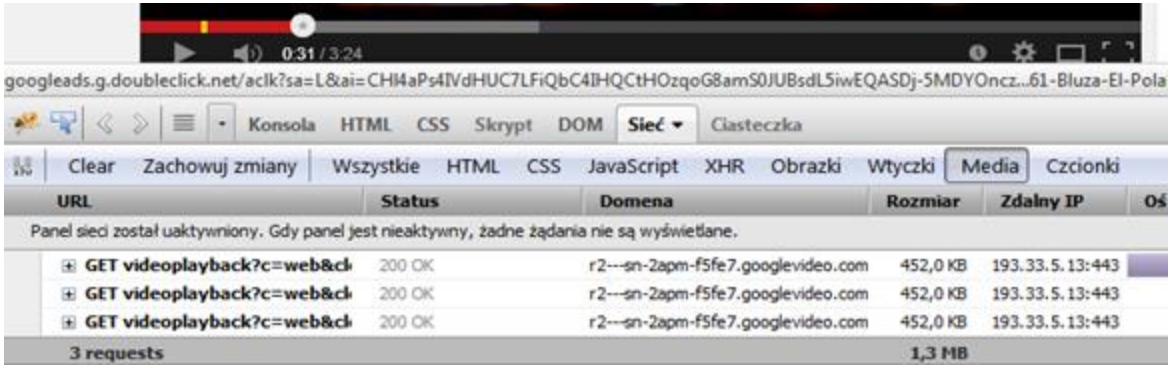
No support for multicast

Large overhead

What is Chunked **PMD**?

Progress Bar

Iteratively growing list of fragments (chunks) requested for download

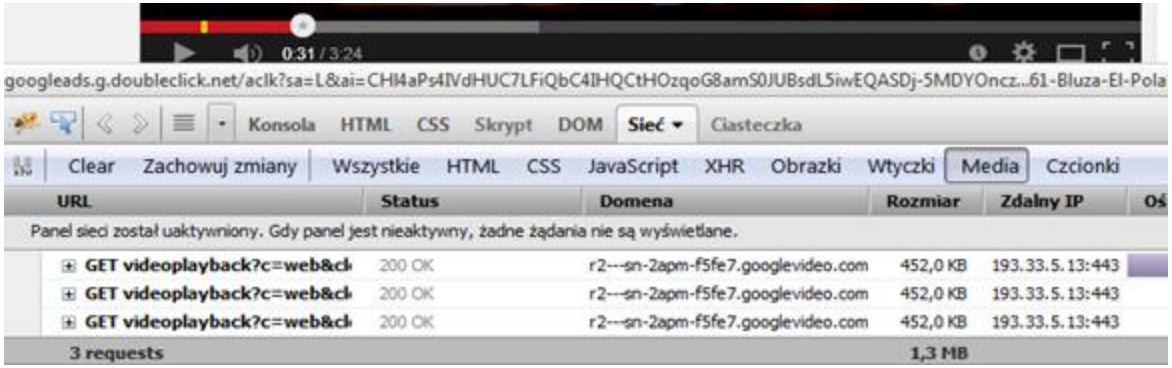


- Not requesting full content in **HTTP** request anymore
- Breaking content into sequence of small **HTTP-based** file segments
- Each segment containing short interval of playback time of content

What is Chunked **PMD**?

Progress Bar

Iteratively growing list of fragments (chunks) requested for download

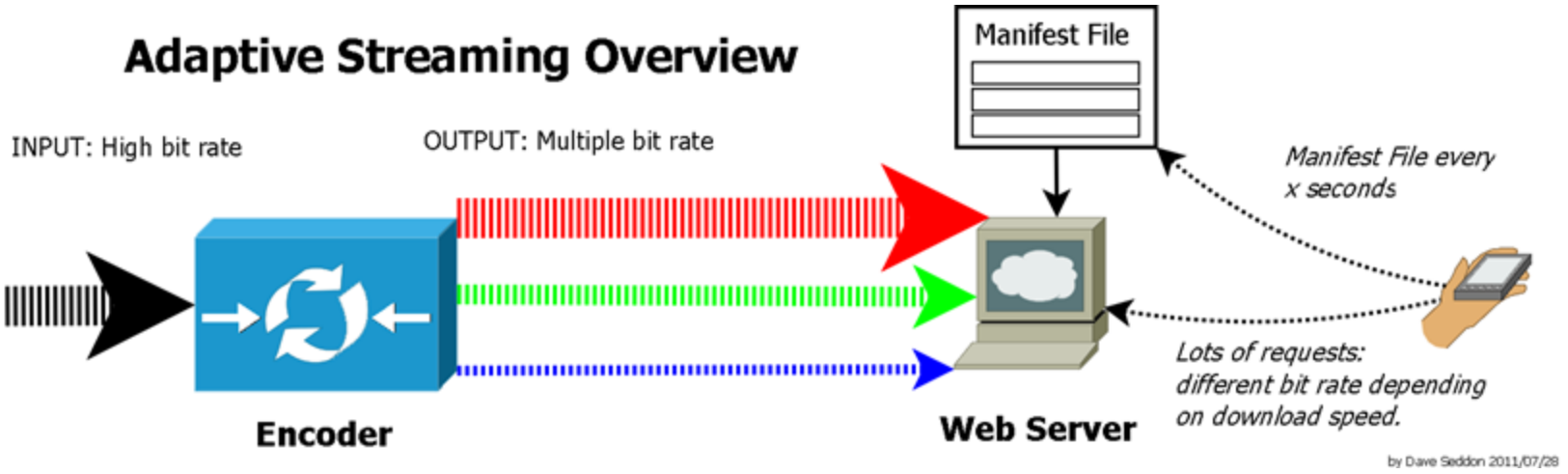


- Technology requesting (e.g.) each **Group of Pictures (GOP)** in a separate request
- Playback begins once chunk downloaded
- **Examples**: **YouTube**, **Vimeo**

Dynamic Adaptive Streaming over **HTTP**

- Continuous feedback from clients
  - Based on bandwidth and CPU usage
- Content made available at a variety of bit-rates
  - Each stream divided into short length (2 - 10 seconds) fragments
- Clients having the extensive control
  - Smooth changes of quality
  - Subtitles language change

Dynamic Adaptive Streaming over **HTTP** cont.



Manifest File

- File containing **metadata** for group of accompanying files being part of
  - Set, or
  - Coherent unit
- Term from cargo shipping procedure, where **ship manifest** listing:
  - Crew of vessel, and/or
  - Cargo of vessel
- In **adaptive bitrate streaming**, client downloading manifest (playlist) file describing:
  - Available stream segments, and
  - Their respective bit rates

Manifest
Files for HTTP Streaming

https://www.menti.com/348xpctthx

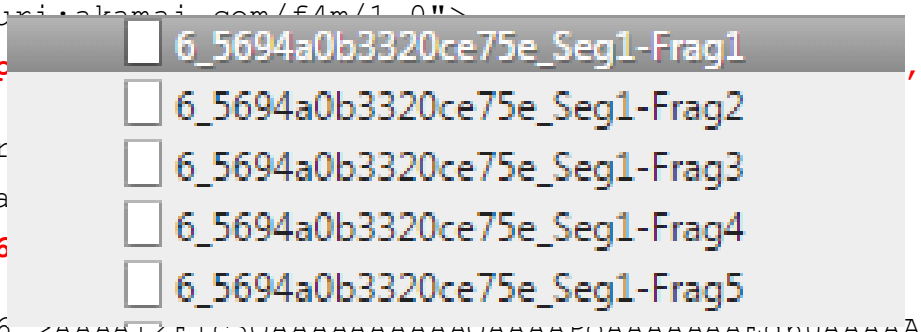
Solutions for Dynamic Adaptive Streaming over **HTTP**

- Industry solutions (similar to each-other) – mystery of terms:
  - **HTTP Dynamic Streaming (HDS)** by **Adobe**
  - **HTTP Live Streaming (HLS)** by **Apple**
  - **HTTP Silverlight/Smooth Streaming (HSS)** by **Microsoft**
- International standard known as **MPEG-DASH**

HDS

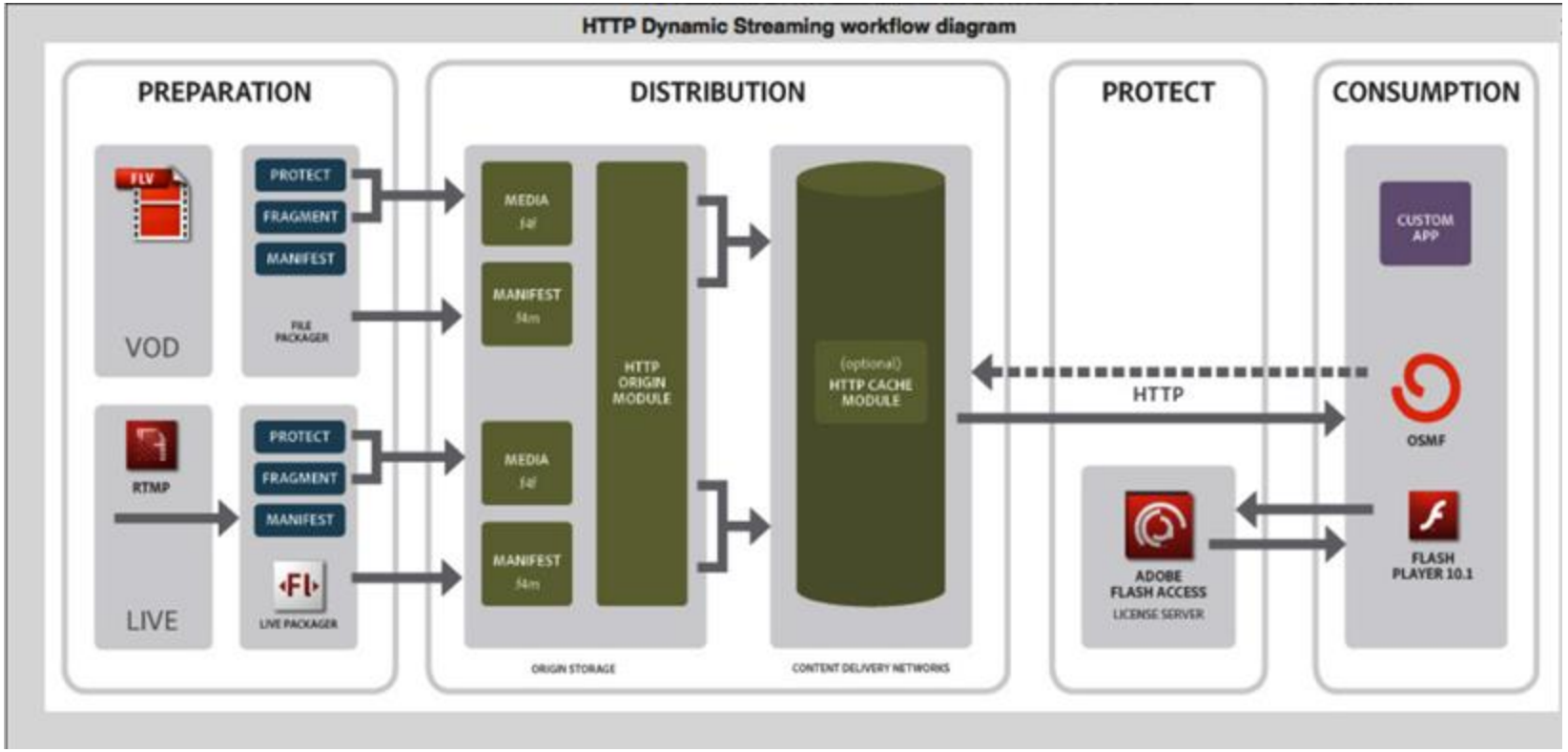
- Supports both: live and on-demand deliveries
- **Multi-Bit-Rate** (**MBR**) support
- Accepting **RTMP** feed (over port **1935**) as input signal from encoder
- Allows for parallel publishing to backup location – **redundancy** (**backup stream**)

Syntax of **manifest.f4m** file – Encoded Suite of Details about Order of Fragments (**.f4**)




```
<?xml version="1.0" encoding="UTF-8"?>

<manifest xmlns="http://ns.adobe.com/f4m/1.0"
xmlns:akamai="uri:akamai.com/f4m/1.0">

<id>/multi/comp                                                    ,2500,4000,9000,k.mp4
.csmil_0</id>

  <streamType>r

  <akamai:strea

  <duration>306

<bootstrapInfo
id="bootstrap_6 >AAAA12F1C3QAAAAAAAAAAQAAAAP0AAAAAAAEq00AAAAAAAAAAAAAAAAAAAQAAABlh
c3J0AAAAAAAAAAABAAAAAQAAADMBAAAARmFmcnQAAAAAAAAD6AAAAADAAAAAQAAAAAAAAAAAAAXcAAAA
DMAAAAAAST4AAAF80AAAAAAAAAAAAAAAAAAAAAA==</bootstrapInfo>

<media bitrate="9326" url="6_5694a0b3320ce75e_" bootstrapInfoId="bootstrap_6">
<metadata>AgAKb25NZXRhRGF0YQgAAAAMAAhkdXJhdGlvbgBAcyF87ZFocwAFd2lkdGgAQJ4AAAAAAAA
ABmhlaWdodABAkOAAAAAAAAANdmlkZW9kYXRhcmF0ZQBAwheSP3JsVAAJZnJhbWVyYXRlAEA998/p06bg
AAx2aWRlb2NvZGVjaWQAQBwAAAAAAAADWF1ZGlvZGF0YXJhdGUAQE/8AJQpGtMAD2F1ZGlvc2FtcGxlc
mF0ZQBA53AAAAAAAAAPYXVkaW9zYW1wbGVzaXplAEAwAAAAAAAAAZzdGVyZW8BAQAMYXVkaW9jb2RlY2
lkAEAkAAAAAAAAAhmaWxlc2l6ZQBBtUVpJAAAAAAACQ==</metadata>
```

HDS Architecture Overview



HLS

Dedicated mostly for users of **iOS** and **macOS** (**Mac**)

» But other vendors implemented their clients based on the open specification (documented in RFC 8216)

Supports live and on-demand streaming

Multi Bit-Rate (**MBR**) support

Allows for parallel publishing to backup location – **redundancy** (**backup stream**)

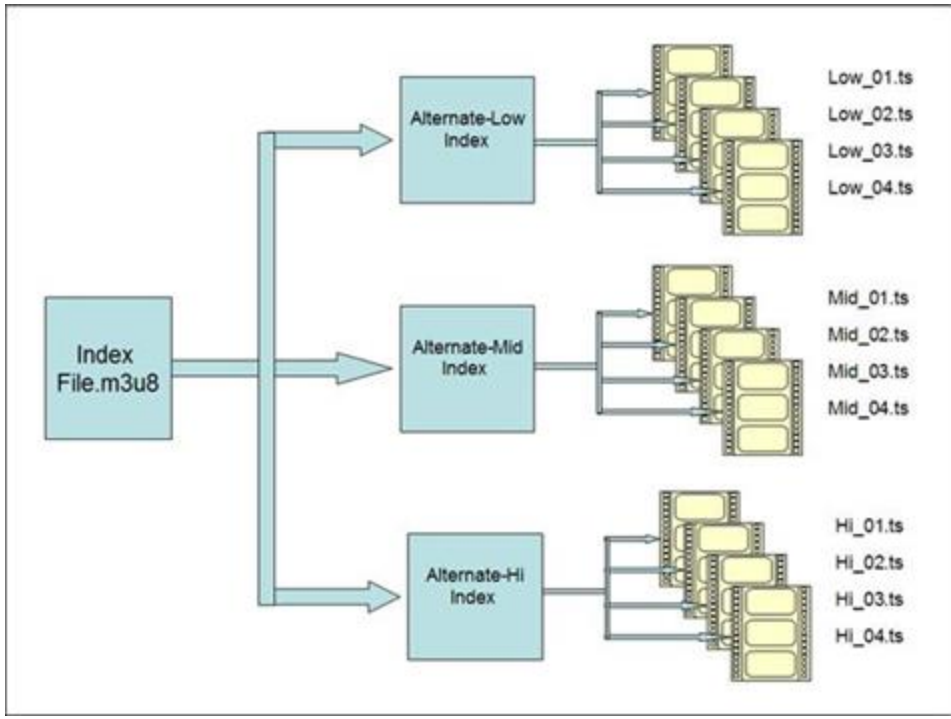
Exemplary URL: https://bitdash-a.akamaihd.net/content/sintel/hls/playlist.m3u8

Example of **master.m3u8** File – But Where Is Reference to Segments (**.ts**)?!

```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=548000,RESOLUTION=320x240,CODECS="avc1.66.30, mp4a.40.34"
http://livetest_rm-lh.akamaihd.net/i/stream_1@143961/index_500_av-p.m3u8?sd=10&rebase=on
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=548000,RESOLUTION=320x240,CODECS="avc1.66.30, mp4a.40.34"
http://livetest_rm-lh.akamaihd.net/i/stream_1@143961/index_500_av-b.m3u8?sd=10&rebase=on
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1048000,RESOLUTION=320x240,CODECS="avc1.66.30, mp4a.40.34"
http://livetest_rm-lh.akamaihd.net/i/stream_1@143961/index_1000_av-p.m3u8?sd=10&rebase=on
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1048000,RESOLUTION=320x240,CODECS="avc1.66.30, mp4a.40.34"
http://livetest_rm-lh.akamaihd.net/i/stream_1@143961/index_1000_av-b.m3u8?sd=10&rebase=on
```

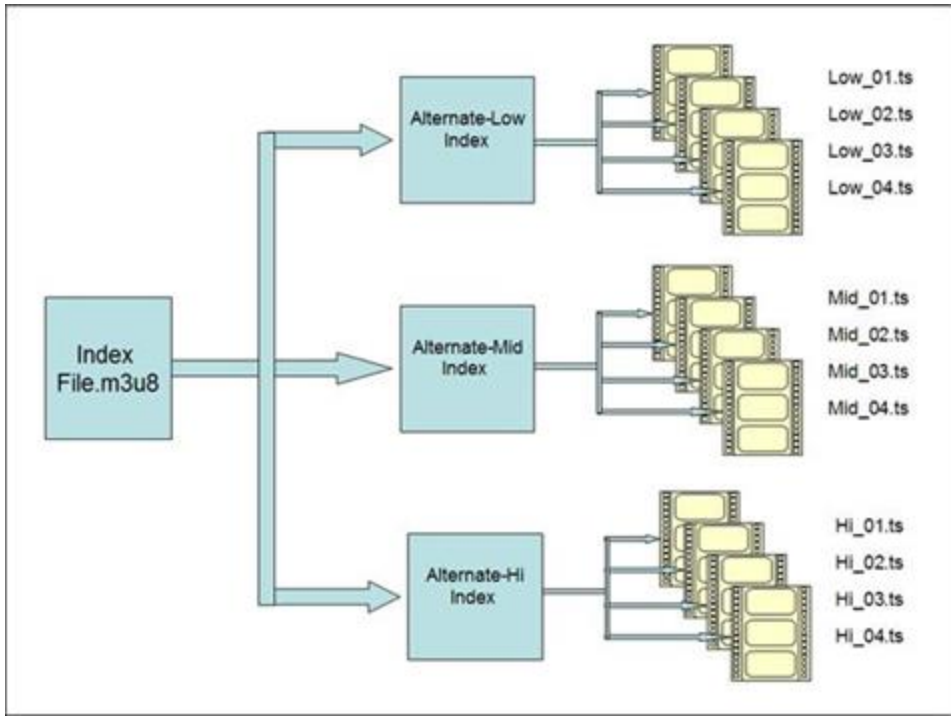
...One Level Deeper (Hierarchical Tree)

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-ALLOW-CACHE:YES
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:75973
#EXTINF:3.249,
http://livetest_rm-lh.akamaihd.net/i/stream_1@143961/segment75973_500_av-p.ts
#EXTINF:10.000,
http://livetest_rm-lh.akamaihd.net/i/stream_1@143961/segment75974_500_av-p.ts
#EXTINF:10.000,
http://livetest_rm-lh.akamaihd.net/i/stream_1@143961/segment75975_500_av-p.ts
```



...One Level Deeper (Hierarchical Tree)

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-ALLOW-CACHE:YES
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:75973
#EXTINF:3.249,
http://livetest_rm-lh.akamaihd.net/i/stream_1@143961/segment75973_500_av-p.ts
#EXTINF:10.000,
http://livetest_rm-lh.akamaihd.net/i/stream_1@143961/segment75974_500_av-p.ts
#EXTINF:10.000,
http://livetest_rm-lh.akamaihd.net/i/stream_1@143961/segment75975_500_av-p.ts
```



How to Convert “Typical” **.flv** or **.mp4** Video into **.m3u8**?

» Very easily ☺

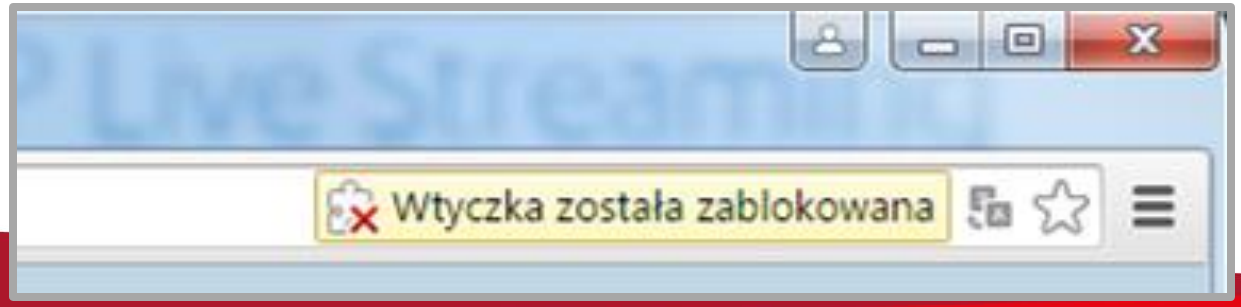
» Using **FFMPEG**:

```
ffmpeg -i webcam.flv -vcodec
h264 -hls_list_size 0
-segment_list webcam.m3u8
-segment_format ts
test_webcam.m3u8
```

HSS

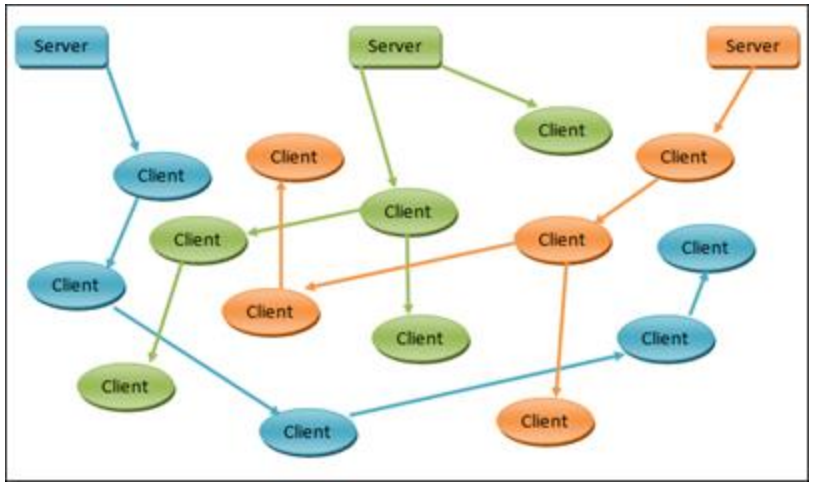
- Similar (tree-like) hierarchy of files:
  - Client manifest (**.ismc**)
  - Video fragment (**.ismv**)
  - Audio fragment (**.isma**)
- **No HSS support in browsers since 2015!**

Microsoft



Peer-to-Peer (P2P)
Networks Serving Video Streams

- Used in connection with other streaming methods: **RTP** and **PMD**
- Increased geographical coverage
- Better bandwidth utilisation
- Saving costs to broadcaster
- Long latency times (even up to 90 seconds!)



Attribution: **Soumyasch** at **English language Wikipedia**

Recapitulation

PMD

- Multimedia content stored on a local machine
- No rate control
- Traditionally, the whole file is transmitted (even if it is not necessary)
- No need to keep many versions of the same file (on the server side)
- Uses existing HTTP architecture

Streaming

- No content stored on a local machine
- Adaptive rate control
- Only the chunk being watched is transmitted
- Built-in support for fast-forwarding
- Uses existing HTTP architecture

Reference

Pantos, R., & May, W. (2017). HTTP live streaming (No. RFC 8216). [Online] https://tools.ietf.org/html/rfc8216

The manifest (playlist) files for HTTP streaming can be built as…

https://www.mentimeter.com/s/39651b0f05220895431b3db039bf40e3/25de82744365