GPAC Licensing

Broadcasting low-latency OTT content with ATSC3 and DVB-mABR

About me (Romain Bouqueau)

Open-source & standard contributor

2005-2009 Allegro DVT (now ATEME)

- Video encoder quality
- Performances

2009-2012 Telecom ParisTech (academic lab)

- R&D engineer: mux/streaming, OSS, standards, …

2012-… Founder at Motion Spell / GPAC Licensing & various startups

https://www.linkedin.com/in/rbouqueau/

Our long journey with low latency OTT

Overhead and performance of low latency live streaming using MPEG-DASH

Publisher: IEEE | Cite This | PDF

GPAC Licensing PRODUCT S

By Romain Bouqueau Jul 9, 2014 13 comments

Experts Opinions, GPAC Licensing, GPAC Open-Source, Technology

Diving into ultra-low latency for live using MPEG-DASH

Low latency with HTTP-streaming (segment-based) technologies is a challenge. In particular MPEG-DASH is gaining adoption among international industry consortiums such as DVB with ~~DVB DASH~~ or ETSI with HbbTV 2.0 (not published yet), with a growing focus on live use-cases. For such applications, latency is a concern.

In this article, we will show you how **the GPAC team studied the impact on the overhead of HD streaming with very low latency using MPEG-DASH** and demonstrated that overhead on the transport side is negligible, in the order of 1%. At 1% overhead, we could demonstrate a 240ms latency. With such a low latency, interactive or bidirectional applications such as video conferencing or live streaming with voting become possible.

Of course such low latency can only be safely reproduced in local network conditions. Yet it shows that the latency is not due to the MPEG-DASH technology but rather to the network conditions. It also shows that a few technical choices can dramatically reduce the latency.

All the tools used for this demonstration are available as free software. Feel free to try and contact us if you have any questions.

Sources of latency

Overhead and Performance of Low Latency Live Streaming using MPEG-DASH

Nassima Bouzakaria, Cyril Concolato, Jean Le Feuvre
Institut Mines-Télécom; Telecom ParisTech; CNRS LTCI
46, rue Barrault, 75013 Paris, France
{nassima.bouzakaria, cyril.concolato, jean.lefeuvre}@telecom-paristech.fr

***Abstract*— HTTP Streaming is a recent topic in multimedia communications with on-going standardization activities, especially with the MPEG DASH standard which covers on demand and live services. One of the main issues in live services deployment is the reduction of the overall latency. Low or very low latency streaming is still a challenge. In this paper, we push the use of DASH to its limits with regards to latency, down to fragments being only one frame, and evaluate the overhead introduced by that approach and the combination of: low latency video coding techniques, in particular Gradual Decoding Refresh; low latency HTTP streaming, in particular using chunked-transfer encoding; and associated ISOBMF packaging. We experiment DASH streaming using these techniques in local networks to measure the actual end-to-end latency, as low as 240 milliseconds, for an encoding and packaging overhead in the order of 13% for HD sequences and thus validate the feasibility of very low latency DASH live streaming in local networks.**

Keywords—HTTP Streaming; Live Streaming; Low Latency; MPEG-DASH; Video Encoding; Overhead.

I. Introduction

HTTP Streaming technologies have been introduced recently to deliver multimedia streams, taking into account the constraints of today's networks. They try to overcome the deployment issues of other protocols such as RTP/RTCP/RTSP in environments where firewalls, Network Address Translation (NAT) or UDP traffic filtering are used. HTTP Streaming offers similar features to RTP/RTCP/RTSP streaming as it can be used for on demand or live services, and can be adaptive to network bandwidth fluctuations. However, some key factors are differentiating: HTTP streaming leverages existing HTTP infrastructures (proxies, caches, content delivery networks) to make an efficient use of the network when targeting a large number of clients, in a way similar to multicast streaming but without the deployment issues; relies on a client-centric approach to perform network adaptation, similar to using multiple multicast RTP streams and letting the client decide; and eases content repurposing from live to on-demand and vice-versa. An important standard in the field of HTTP streaming is the MPEG Dynamic Adaptive Streaming over HTTP (DASH) standard [5].

While packet-based streaming solutions, e.g. using RTP, can achieve latency, in the order of frames, HTTP streaming solutions such as DASH are not used today for such very low latency streaming. The major reason for that is that HTTP streaming relies on a segmentation process, whereby encoded media frames are aggregated into segments (in the DASH terminology) used as: a download unit in HTTP requests; a buffering unit to smooth network bandwidth variations; an indication of the boundaries to perform seamless switching between streams encoded with different bitrates. For these purposes, segments typically start with a Random Access Point (e.g. an IDR frame in the AVC coding format), and last for a few seconds. Such encoding and segmentation therefore introduce a delay which is not acceptable for low latency streaming, and in particular for live.

Traditionally, very low latency streaming is required for interactive or bidirectional applications such as video conferencing or live streaming with voting. Despite the benefits of HTTP streaming explained earlier, DASH is not initially adapted for such low latency. In this paper, we would like to investigate how to achieve very low latency, i.e. latency similar the one achievable with RTP, but using DASH, to benefit from its advantages in particular, in scenarios such as interactive streaming or hybrid delivery. In the hybrid delivery scenario, where DASH streaming over broadband network is combined with a broadcast service, the latency of the DASH system should be lower than the broadcast, and if no additional buffer is introduced in the broadcast (at the client side or at the encoder side), this means that the DASH system should achieve very low latency. Even in local area networks, where the network jitter is small but where bandwidth can vary (for example when shared between users), adaptive HTTP streaming can still be useful and could benefit from very low latency, e.g. in a local broadcast of a live event.

In this paper, we push the use of DASH to its limits to evaluate different aspects related to latency. We consider the use of DASH over HTTP 1.1 where "chunked-transfer encoding" is used and rely on specific parts of segments, called fragments, being downloaded before entire segments are ready. Additionally, to have meaningful low latency in DASH, we use low latency video coding tools, in particular the Gradual Decoding Refresh feature of the AVC standard. In this paper, we will evaluate the usefulness of this approach both in terms of latency and overhead.

Section II of this paper will present the state of the art in HTTP live streaming. Section III will describe our approach and propose an evaluation of the introduced overhead. Section IV will describe some experiments made to validate the approach and Section V will conclude the paper and propose future work.

What is latency?

Latency is the time elapsed between two events, which in a perfect world, would be simultaneous.

In a live video streaming scenario, the latency is typically the time elapsed between the capture of a video frame, and this frame being displayed on the end-user terminal.

We all have a feeling about what latency is:

Musicians playing with digital audio workstations are used to the delay between the pressing of a key and the sound coming out of from the speakers. In this case, 200ms latency is considered very high, and 10ms latency is considered acceptable.

Watchers of the soccer world cup know they don't all experience the same delay between the moment of goal is scored on the field, the moment they hear their neighboors scream with joy, and the moment they finally see the actual action on their screen.

In live TV streaming, a latency of 1-2 seconds is considered OK. OTT streaming like HLS or MPEG-DASH typically introduce 10s latency, or more.
This isn't generally considered an issue, as it has most of the time no impact
on the viewer's experience (as there's no interaction between the user and the source). A latency of some video frames (~200 ms) is considered pretty good. Some video applications, like WiGig, require very low video latencies (~2ms), which is less than the duration of one video frame.

In general, we prefer latency to be as small as possible.
What is causing this latency, and what is causing it to go up or down?

In live video streaming, there are three sources of latency, which indirectly determine the overall resulting latency.

Processing latency

Also known as "the computers are not fast enough", processing latency is the time spent waiting for any processor to finish its work. The longer the task, the higher the latency. This kind of latency goes to zero when the processing power (CPU frequency, memory bandwidth, etc.) goes to infinity.

An infinitly fast computer would introduce zero processing latency. In the absence of such a computer, the software engineer can reduce this latency by optimizing the code so there's less processing work to do.

Processing latency generally isn't a problem in live video streaming setups:

Video encoders and decoders are designed to be fast enough to cope with at least 25 frames per second. Given that video compression requires encoders/decoders to mostly operate sequentially, this implies that the average frame processing time must be less than 40 milliseconds.

Communication latency

Also known as "the packets are not travelling fast enough" (or even "the distance is too big"), communication latency is about signal propagation over distances.

Electrical or optical signals going through wires take some time to go from
one point to another (the speed of light being the ultimate limit). This is partly what you're measuring when you ping google.com. Technically, a big part of the ping value comes from the processing latency from routers between you and the server.

The over-the-top live streaming engineer generally has zero control on this kind of latency. We generally consider it as a black box ensuring mostly-reliable transmission between two endpoints.

Communication latency is often confused with data throughput rate, also known as "bandwidth". Depending on the transmission protocol, a latency might indeed play an important role in the resulting throughput.

Algorithmic latency

Algorithmic latency is at the same time the most difficult to understand, and the most interesting. It's generally the biggest cause of overall latency, and also, the one we have the most control over. It's the direct

Problem statement

OTT: 1 user = 1 streaming session

For popular live events:

- Expensive delivery
- QoS/QoE may suffer

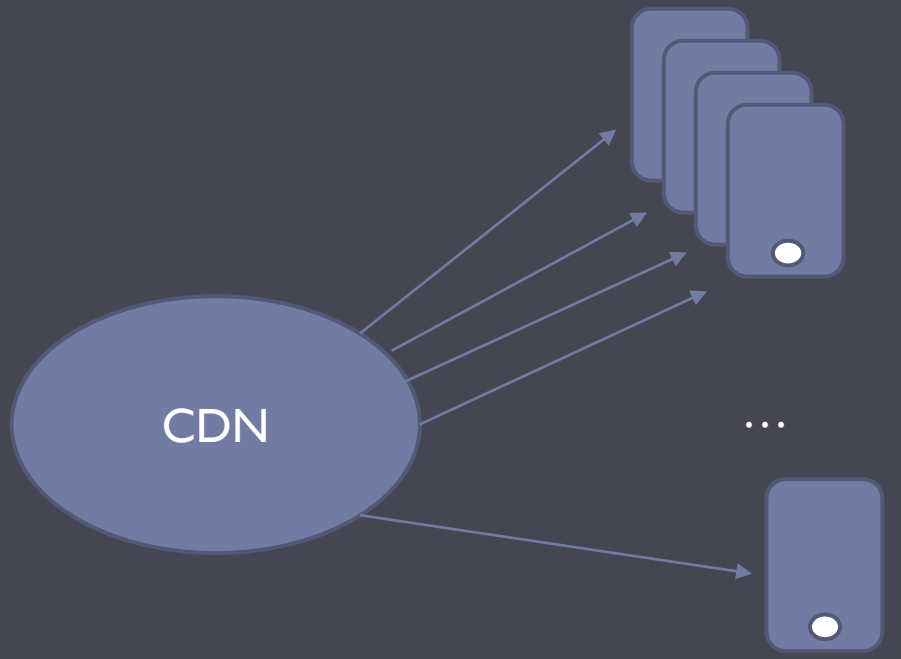
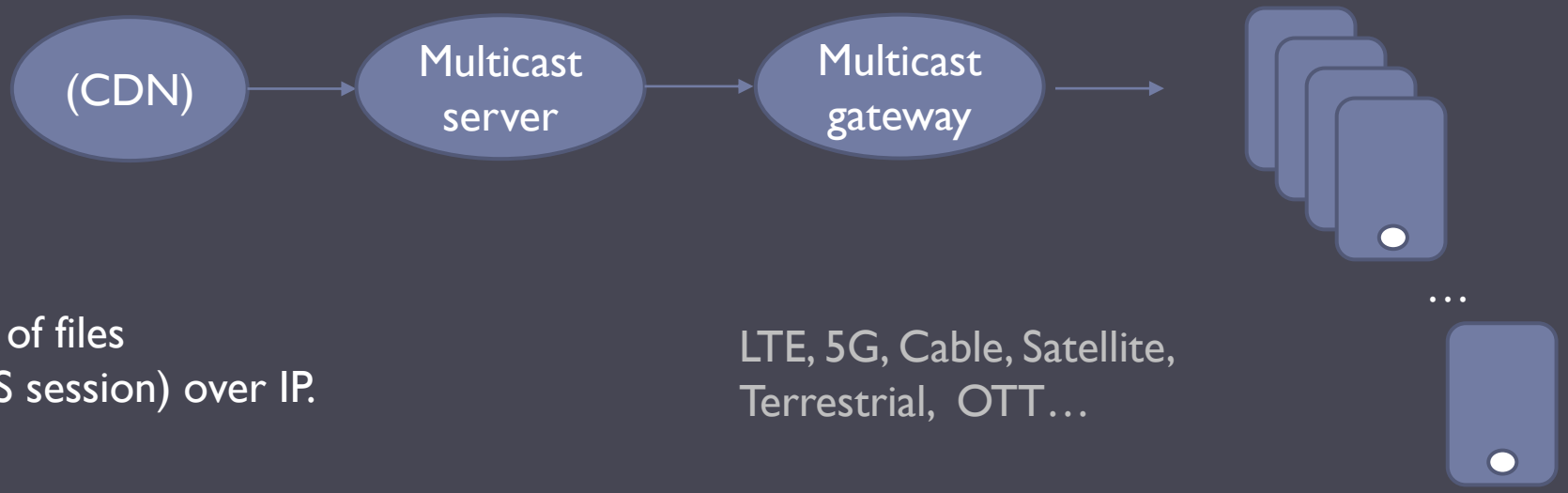


photo by master1305 - www.freepik.coma

Solution



Transport a local filesystem (e.g. set of files representing your live DASH or HLS session) over IP.

LTE, 5G, Cable, Satellite, Terrestrial, OTT…

Several option, ROUTE (inherits from FLUTE) being the most consensual one:

- Open specification
- Adopted by ATSC3 and DVB-mABR
- Replaces HTTP/TCP by ROUTE/UDP

photo by master1305 - www.freepik.coma

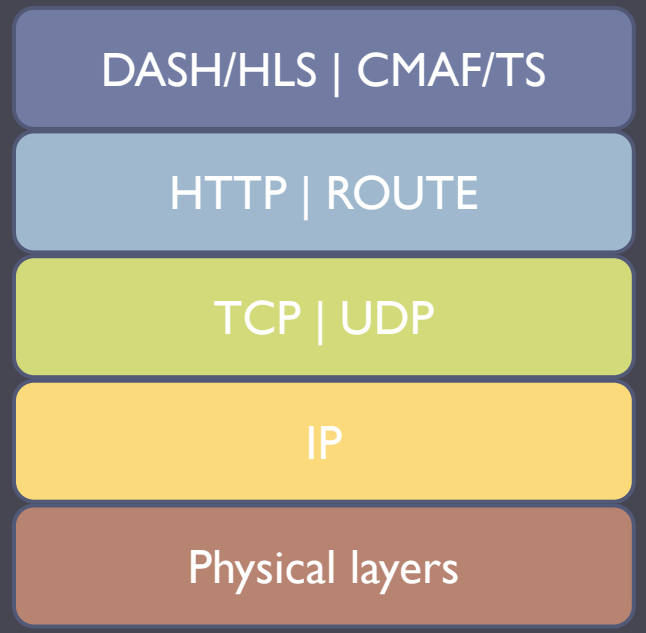
Pros/cons

Pros:

- Agnotic of transported files
- Hybrid broadcast/broadband (5G)
- Leverages the upper layers
  - Encryption
  - Personalized ads
  - …
  - Low latency

Cons:

- Security (IP)
- Live: for file-based sessions, need to update the session
- Carousel: server scheduling complexity
- Recovery mechanisms complexity
- Manifest manipulation:
  - low latency



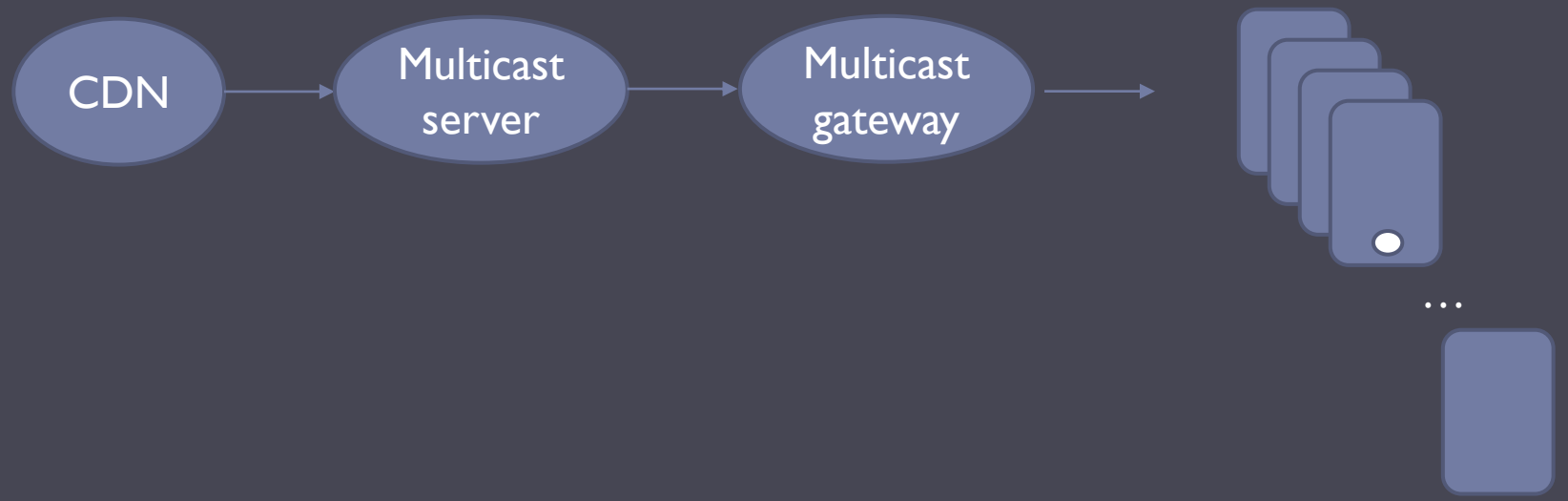
ROUTE: low latency specifics

ATSC3:

- Agnostic: the streamer or the player should compensate for the distribution latency
  - MPD@suggestedPresentationDelay

DVB mABR:

- Manifest manipulation in the Gateway
- Carousel scheduling



Experimenting with the GPAC open-source framework (http://gpac.io)

ROUTE:

- ROUTE: works with DASH/HLS/MPEG2-TS/…
- ROUTE: ATSC and DVB flavours

Server (low latency):

- `gpac -i URL dashin:filemode @ -o route://IP:ADDR:llmode`

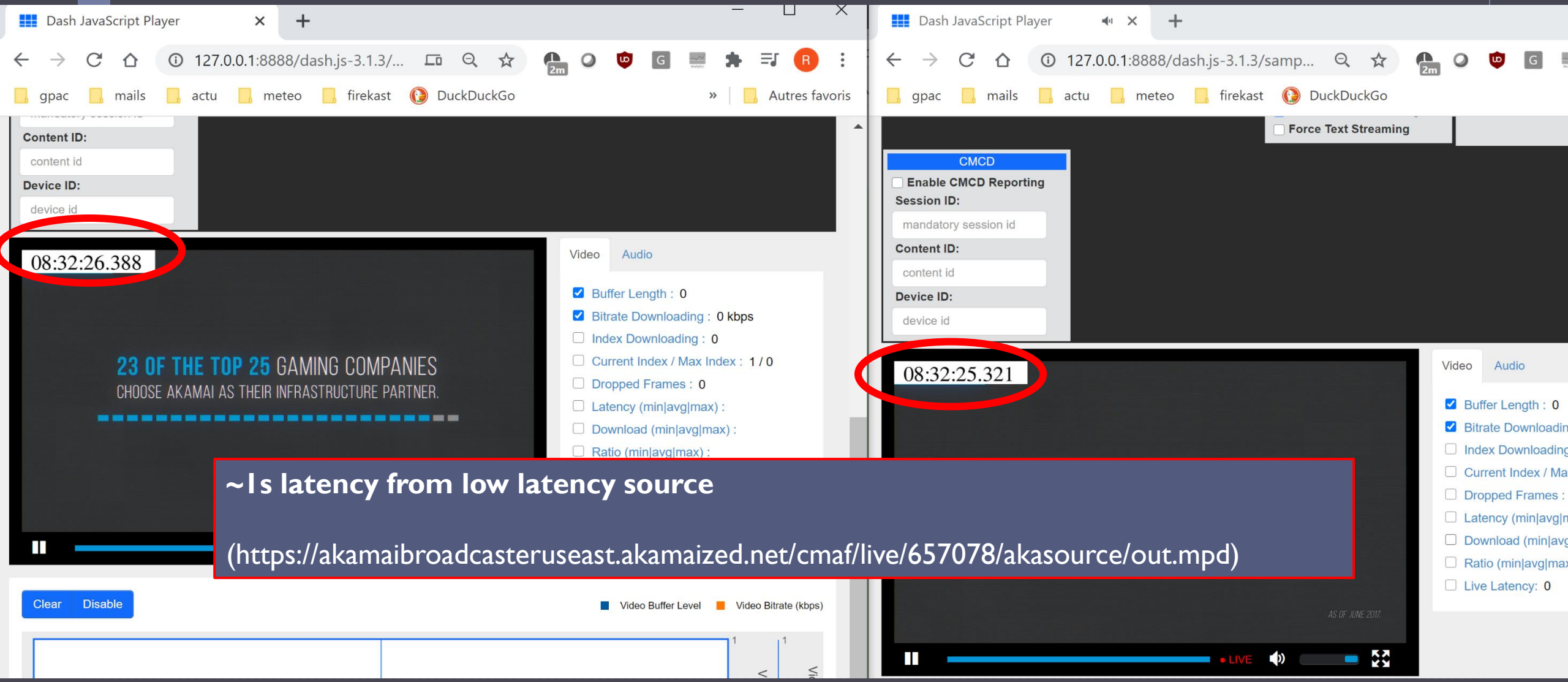
Gateway (HTTP, TCP port 10000):

- `gpac -i route://IP:ADDR/:max_segs=4 dashin:filemode @ httpout:port=10000 --rdirs=temp --reqlog=* --cors`

Playback:

- `gpac -play route://IP:ADDR`
- dash.js (using the HTTP gateway)
  - next slide

Transmission latency: Source ---------- Reception



~1s latency from low latency source

(https://akamaibroadcasteruseast.akamaized.net/cmaf/live/657078/akasource/out.mpd)

More GPAC command-lines

Gateway (repush to CDN)

- `gpac -i route://225.1.1.0:6000 dashin:filemode @ -o http://127.0.0.1:8080/live.mpd --hmode=push`

Gateway (repush as TS)

- `gpac -i route://225.1.1.0:6000/ -o udp://225.1.1.10:1234/:ext=ts`

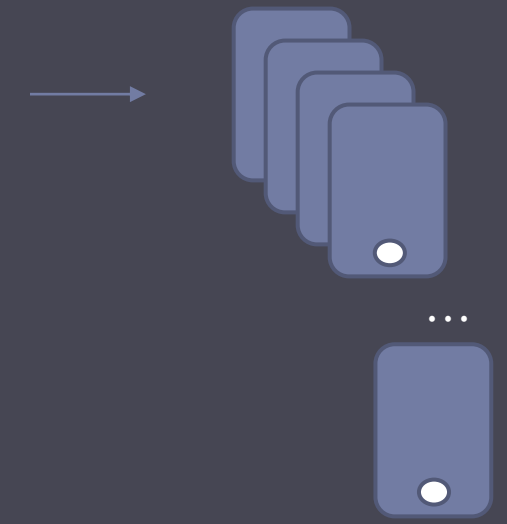
Dump session:

- `gpac -i route://225.1.1.0:6000:odir=dump_route`

More at https://github.com/gpac/gpac/wiki/route

Going further with hybrid low latency

- Low latency in broadcast mode is safer
  - Not going through unmanaged networks
- Hybrid broadcast-broadband interesting use-cases, all offering their technical challenges related to low latency:
  - SHVC (ATSC3) to augment from HD to UHDv2
  - Personalized ads
  - Linear personalized content
  - Interactivity



Thank you!

romain.bouqueau@gpac-licensing.com