Meeting live broadcast requirements – the latest on DASH Low Latency

DVB World Online Presentation – March 2020

Will Law
Chief Architect, Akamai
Typical Broadcast Latency
6 seconds
45+ seconds
30 seconds
20 seconds
10 seconds
8 seconds

LOW LATENCY RANGE

ULTRA LOW LATENCY RANGE

SUB-SECOND

LEGACY LATENCY RANGE

BUSINESS USE CASES

VOICE
LIVE AUCTIONS
GAMBLING,

BROADCAST LATENCY – DTT, DTH,

Live Sports and e-Sports

SOCIAL MEDIA – SMS/IP

OTT STREAMING WITH HLS & DASH

CHUNKED CMAF segments 1s-6s

WebRTC

DISTRIBUTION TECHNOLOGY

10s segments
6s segments
2s segments
1s segments

Latency Achievable at Scale Via Mainstream CDN Streaming Technologies
CMAF Low Latency Chunks

Example: CMAF Fragment containing a Coded Video Sequence of 20 samples

[Image credit deep inside MPEG somewhere, I suspect Kilroy Hughes]
Why does chunking reduce latency?

LIVE Encoder producing 4s segments
HAS default

Lowest conventional latency

Chunked segments (1000ms per chunk)

Latest chunks & play from start

Latest chunks & decode forward

Defer start
Low latency timeline within DASH IF

- A Live Task Force was established under the IOP working group in late 2017.

- This task force began work Change Request (CR) on ‘Low latency Modes for DASH’, building on work done under the ATSC-DASH project.

- The Live Task issued a request https://github.com/Dash-Industry-Forum/dash.js/issues/1474 to the dash.js project in Dec 2017 to develop low latency streaming capabilities. Goal is to ensure that the end-to-end latency of the streams will be comparable to broadcast latency ~ 3.5 seconds of end-to-end latency.

- First version of the ULL behavior demonstrated in April 2018.

- Low latency guidelines available for community review June 2019.

Player issues: Bandwidth estimation problem

- One consequence of segment data being delivered as fast as it is produced is that the segment download time is approximately equal to the segment duration.
- Conventional throughput estimation algorithms will always produce the answer that throughput equals bandwidth and hence the player will never switch up.
- We need other ideas for estimating bandwidth under these conditions.
- Ali Begen will address this problem and its solution in a subsequent presentation in this webinar.
Player issues: Catch up and rate adjustment

- Once started, a live player cannot adjust its latency ...
- Unless its speeds or slows down its playback rate
- This turns out to be a very useful tool for a player which is dealing with very low buffers and variability in throughput.
Player issues: synchronization

- Combining an external time source with aggressive catch-up and a latency target gives you synchronization.

- Variations caused by CPU capability of devices and delay to receive time source response.

- In practice, can maintain sync to within +/- 3 frames between laptops, desktops and mobile devices.
DASH IF Guidelines on Low Latency

- Published March 27 2020
- Now available at https://dashif.org/guidelines/
- A deep technical resource on how to conduct low latency streaming with DASH.
What will you find in the low latency guidelines?

- Scenario and architecture
- Low latency service offering
- Chunked and on-chunked low latency adaption sets
- Delta with DVB Low latency DASH
- Informative low latency client guidelines for operation
- ABR Encoder, Encryption and MP4/CMAF Packaging
- Encoding and CMAF Chunk Duration Recommendations
- Latency calculations and use of Producer Reference Time
- Resynchronization Points
- Service Configuration Parameters for Low Latency
- Example FFmpeg Configuration
- Example live MPDs
- MPD Generator and Packager Operation for Low-Latency Content
Noteworthy items from DASH IF Guidelines

• Two operational modes are permitted
  – Simple live offering is used by applying @duration signaling and $Number$ based templating
  – Main live offering with the SegmentTimeline as either $Number$ or $Time$ is supported by the proposed updates in MPEG-DASH fourth edition.

• At least one ServiceDescription element shall be present as follows:
  – A Latency element - specifying min, target and max latencies
  – A PlaybackSpeed element – min, max playback speeds for latency correction.

• At least one UTCTiming element with millisecond precision.

• A low latency adaption set can be built in two ways
  – Using non-chunked segments whose duration < 30% of target latency
  – Using chunked segments
Resync elements

- Resynchronization Points are defined in the 4th Edition of MPEG DASH ISO/IEC 23009-1:2020/Amd.1
- Their purpose is to define the location of intermediate resynchronization points within media segments.

- Benefits are
  - Fast random access while maintaining low latency
  - Quick Resynchronization after buffer underruns
  - In-Segment down-switching in case buffer draining is observed
  - Understanding the applied chunk size and duration and hence support for the rate adaptation.

Service offering: DASH-LL streams with 4s CMAF segments with 2s GOP, 500ms chunks.
Resync elements – how to find them

• By providing a **binary map** for each Resynchronization Point in a Resynchronization Index Segment for each Media Segment.
  • This is most easily used for Segments that are fully available on the network.
  • So not very useful for live, especially low latency

• By **signalling in the mpd** the existence of Resynchronization Points in a Media Segment. This permits the player to parse the segment to locate the Resynchronization Point.
Resync elements – example 1

<Resync type="0" dT="500000" dImin="0.03125" dImax="0.09375"/>

<SegmentTemplate timescale="1000000" duration="8000000"
availabilityTimeOffset="7.500" availabilityTimeComplete="false"
initialization="init-stream$RepresentationID$.m4s" media="chunk-stream$RepresentationID$-$Number%05d$.m4s" startNumber="1"/>

<Representation id="0" mimeType="video/mp4" codecs="avc1.640016"
bandwidth="500000" width="960" height="400" sar="1:1" qualityRanking="5"/>

Indicates that

• The point is of type 0 – a CMAF chunk without any promise for specific random-access capabilities.
• The maximum chunk duration is 500ms (500,000/1,000,000)
• The minimum distance in bytes between two Resynchronization Points is 15,625B (0.03125x500,000)
• The maximum distance in bytes between two Resynchronization Points is 46,875B (0.09375x500,000)
Resync elements – example 2

<Representation id="2" mimeType="video/mp4" codecs="avcl.640016" bandwidth="300000" width="720" height="300" sar="1:1" qualityRanking="1">
  <Resync type="2" dT="1000000" dImin="0.1" dImax="0.15" marker="TRUE"/>
</Representation>

Indicates that

- The point is of type 2 – a CMAF chunk that can be used for fast access or switching
- The maximum time delta between these points is 1s (1,000,000/1,000,000)
- The minimum distance in bytes between two Resynchronization Points is 30,000B (0.1x300,000)
- The maximum distance in bytes between two Resynchronization Points is 45,000B (0.15x300,000)
- As the @marker flag is set to true, a DASH client may search for the resync point using a box-parsing algorithm.
ETSI has published **TS 103 285 V1.3.1**, a new version of DVB’s *MPEG-DASH profile for transport of ISO BMFF based DVB services over IP based networks*. The main changes are the inclusion a low latency mode and HDR dynamic mapping.

Difference between DVB and DASH-IF for low latency DASH

- DVB-DASH in clause 11.18.2 recommends the use of @duration together with $Number$, whereas DASH-IF does not make any of such recommendations.

- DVB DASH does not mention the Resync element, but the presence of the element would not break a DVB-DASH client.

- DVB DASH signals Adaptation Sets that are for low latency purposes with an EssentialProperty or SupplementalProperty descriptor with the @schemeIdUri attribute of "urn:dvb:dash:lowlatency:critical:2019" and the @value attribute set to "true". If the SupplementalProperty descriptor is used, then this does not impact regular low-latency DASH clients.
What is the dash.js client?

- A open-source MSE/EME player library, written in Javascript, which serves as a robust DASH player for commercial applications as well as a reference player implementation for the DASH Industry Forum

- It is an open-source project on Github - https://github.com/Dash-Industry-Forum/dash.js/
New APIs to support low latency playback

- `mediaplayer.setLowLatencyEnabled()` – activates low latency behaviors around start-up, bandwidth estimation and buffer management and ABR

- `mediaplayer.setLiveDelay()` – sets the target latency in seconds

- `Mediaplayer.setCatchUpPlaybackRate()` – sets the maximum rate at which the player will adjust its playback speed in order to modify the latency without interrupting stream playback.

- `mediaPLayer.setLowLatencyMinDrift()` – sets the allowable variance from the target delay before the playback rate adjustment kicks in.
Public example
ffmpeg, Akamai CDN, dash.js player

Thank you!