Player Optimizations

Tips and tricks for HTTP adaptive streaming players to improve stability, start-up time, quality and latency.

Will Law

Chief Architect

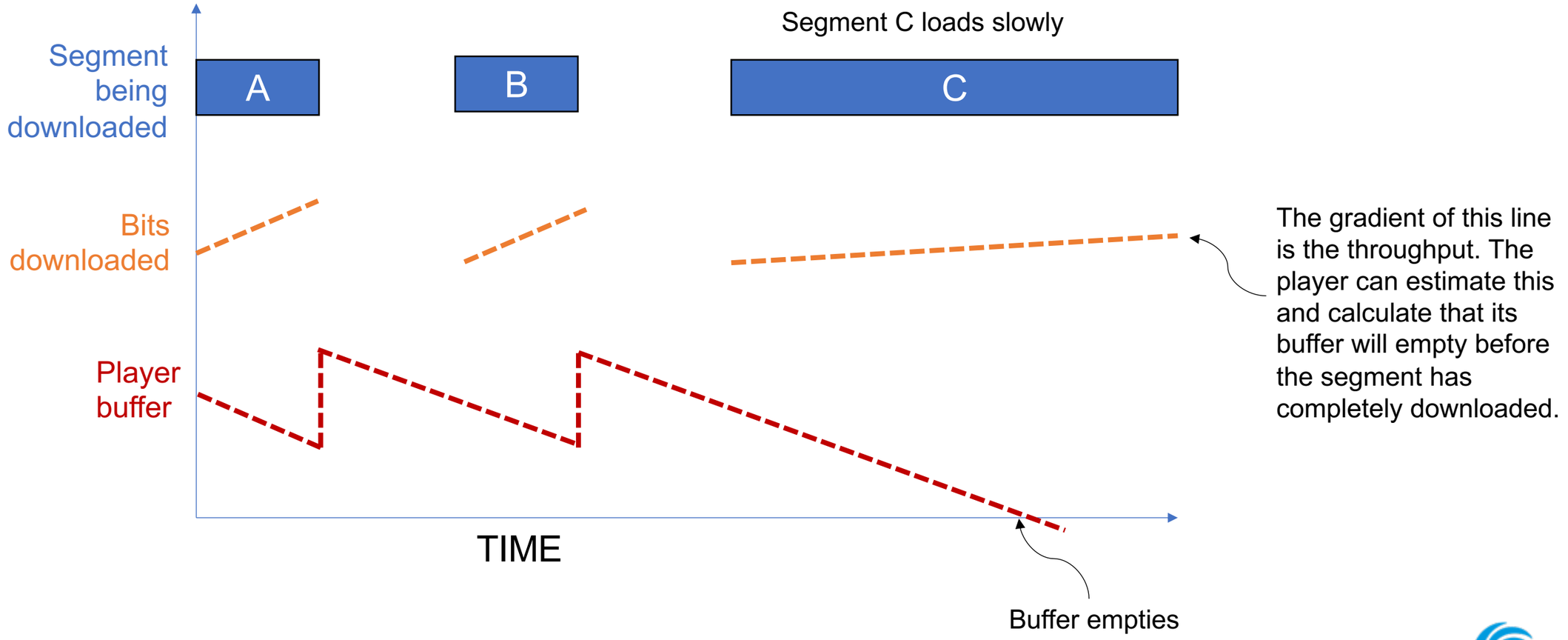
July 2019

Akamai

Experience the Edge

1. Segment Abandonment

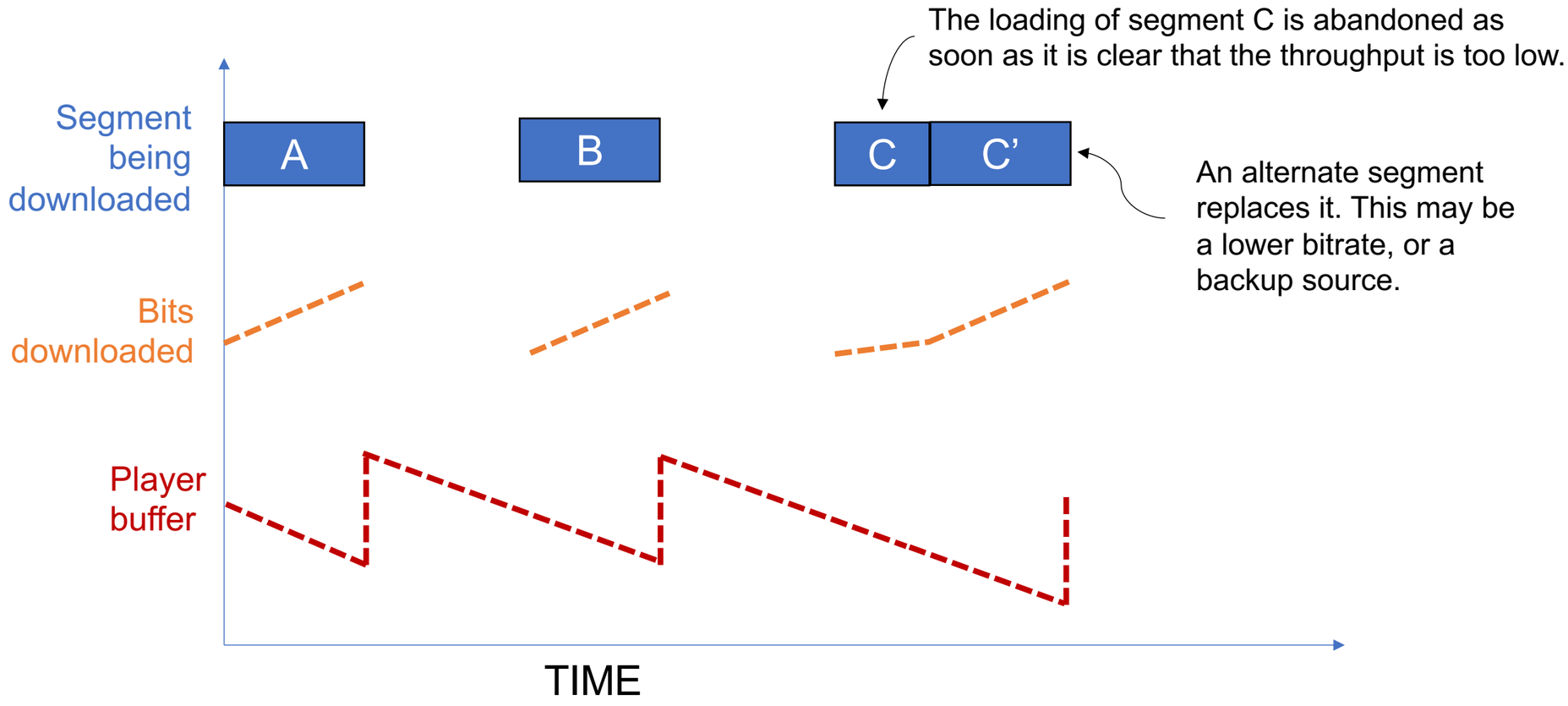
Many players ride slow segments in to the ground …



Akamai

Segment Abandonment – contd.

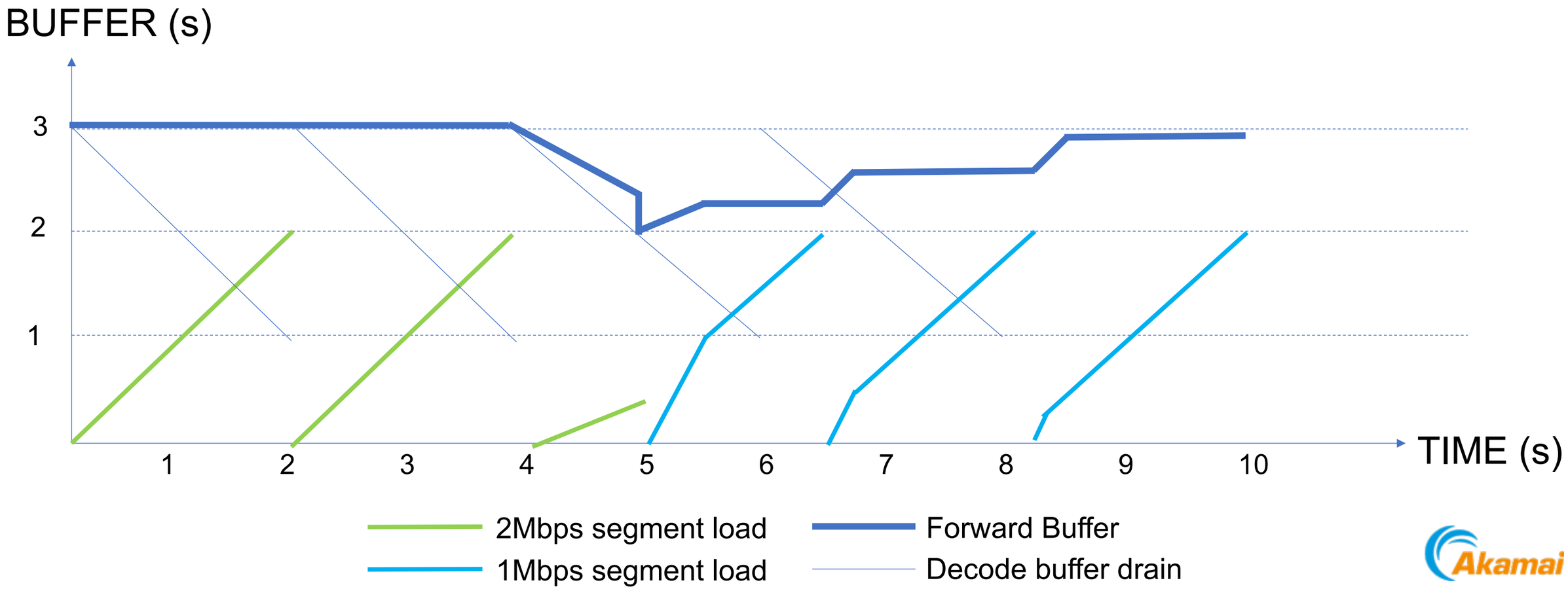
Many players ride slow segments in to the ground …



Akamai

2. ABR switching for low latency with chunked transfer

To enable a player to switch, the forward buffer must be greater than the segment duration. For example, 3s buffer with 2s segments. This would allow 3.5s of E2E latency. Player has at most 1s to decide if it should abandon the segment and switch down.



Akamai

3. Place init files in manifests to remove one RTT from startup

bbb_30fps.mpd	200	http/1.1	xhr	XHRLoader.js:78	3.5 KB	23 ms
bbb_30fps_3840x2160_12000k_0.m4v	200	http/1.1	xhr	XHRLoader.js:78	1.2 KB	162 ms
bbb_a64k_0.m4a	200	http/1.1	xhr	XHRLoader.js:78	1.1 KB	255 ms
bbb_30fps_3840x2160_12000k_1.m4v	200	http/1.1	xhr	XHRLoader.js:78	2.2 MB	857 ms
bbb_a64k_1.m4a	200	http/1.1	xhr	XHRLoader.js:78	33.4 KB	36 ms

You can represent the Initialization segment directly in the MPD using a data: URL, per RFC 2397. This simplifies your stream setup and removes two requests before media can be played.

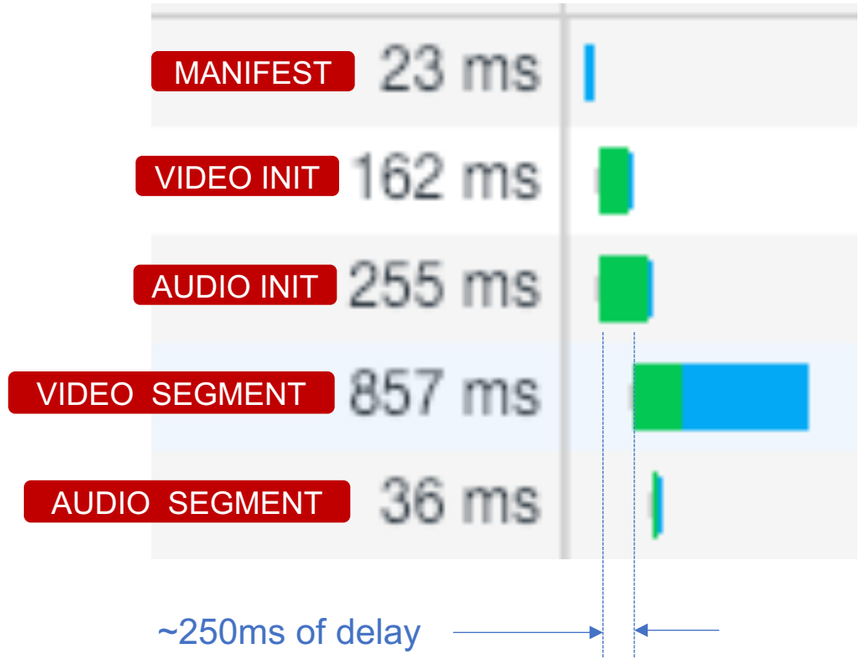
```
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 DASH-MPD.xsd"  mediaPresentationDuration="PT3256S"
minBufferTime="PT1.500000S" profiles="urn:mpeg:dash:profile:isoff-live:2011">
	<Period>
		<AdaptationSet id="1" segmentAlignment="true" sar="1:1" mimeType="video/mp4" startWithSAP="2" maxPlayoutRate="1">
			<InbandEventStream schemeIdUri="tag:rdmedia.bbc.co.uk,2014:events/ballposition" value="1"/>
			<Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
			<BaseURL>avc3-events/</BaseURL>
			<SegmentTemplate media="$RepresentationID$/$Number%06d$.m4s" timescale="1000" duration="3840" startNumber="1"/>
			<Representation id="960x540p50" bandwidth="2814440" width="960" height="540" frameRate="50/0" codecs="avc3.64001f" maxPlayoutRate="1" scanType="progressive">
				<SegmentTemplate initialization="data:video/mp4;base64,AAAAHGZ0eXBpc282AAAAAWF2YzFpc29tZGFzaAAAAldtb292...AAAAAAAAAAAAAAAQc3RjbwAAAAAAAAAA"/>
			</Representation>
			<Representation id="192x108p6_25" bandwidth="31368" width="192" height="108" frameRate="25/0" codecs="avc3.42c015" maxPlayoutRate="1" scanType="progressive">
				<SegmentTemplate initialization="data:video/mp4;base64,AAAAHGZ0eXBpc282AAAAAWF2YzFpc29tZGFzaAAAAlNtb292...AAAAAAAAABBzdHNjAAAAAAAAAAAAAAAUc3RzegAAAAAAAAAAAAAAAAAAAAABBzdGNvAAAAAAAAAAA="/>
			</Representation>
		</AdaptationSet>
	</Period>
</MPD>
```

Akamai

4. Load init segments in parallel to media segments

bbb_30fps.mpd	200	http/1.1	xhr	XHRLoader.js:78	3.5 KB	23 ms
bbb_30fps_3840x2160_12000k_0.m4v	200	http/1.1	xhr	XHRLoader.js:78	1.2 KB	162 ms
bbb_a64k_0.m4a	200	http/1.1	xhr	XHRLoader.js:78	1.1 KB	255 ms
bbb_30fps_3840x2160_12000k_1.m4v	200	http/1.1	xhr	XHRLoader.js:78	2.2 MB	857 ms
bbb_a64k_1.m4a	200	http/1.1	xhr	XHRLoader.js:78	33.4 KB	36 ms

- Many players defer loading of the media segments until the init segments have been retrieved.
- This slows startup.
- Request init segments and media segments in parallel.



Akamai

5. Use request headers or query args to send useful information to CDNs with each segment request

Several useful properties that the client can send the server are

- Session markers (GUID per session with no PII)
  - Being able to allocate log-lines to user-sessions enables more accurate QoE data to be extracted.
- Segment duration
  - The server can tell whether it is delivering the segment too slowly and take proactive action to switch to an alternate upstream path
- Segment max bitrate
  - If a segment contains media encoded at 3Mbps and the client has a 20Mbps connection, default is to deliver the response as fast as possible. However server could deliver the response at 10Mbps and allocate the bandwidth to other competing clients. This would be fairer all around and still fast enough for the player to build a starting buffer.

Provisional headers are shown

Origin: http://mediapm.edgesuite.net

Referer: http://mediapm.edgesuite.net/will/dash/lowlatency/low-latency-public-variable.html?url=https://akamaibroadcasteruseast.akamaized.net/cmaf/live/657078/akasource/out.mpd&latency=3.2

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36

Segment-Duration: 4004

Bitrate-kbps: 2000

Segment-Properties: 4004,2000

Akamai

6. Provide hints to a player as to where to start in a bitrate ladder

What is your player algorithm for selecting the starting bitrate?

- Too low and the user sees poor visual quality
- Too high and you prebuffer immediately or have a long startup if segment abandonment is in place.
- It is a fallacy that starting at the lowest bitrate meaningfully reduces the VST. It does, but at a level that most users will not notice and that is not an acceptable trade-off for the reduced quality, which sers will notice.

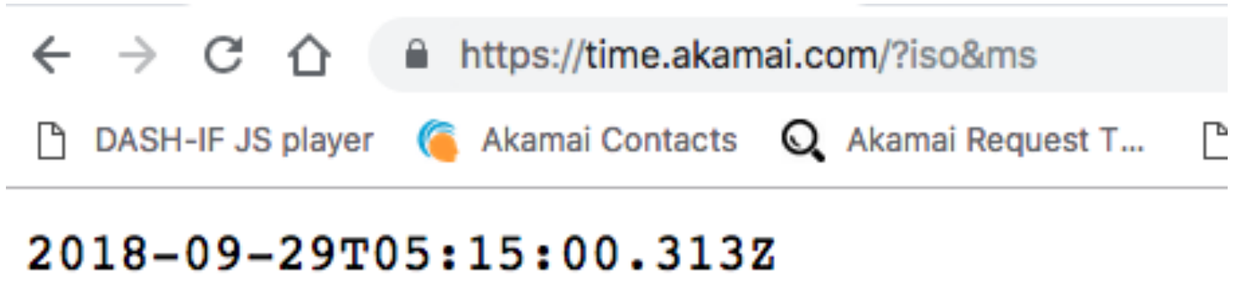
Where get hints from?

- CDN can provide throughput hint on manifest and/or init segment response
- Run a bandwidth test as page || app is loading.
- Concurrent peers in same AS (use your metrics reporting system to collect)
- Stored values (cookies, indexDB) are often unreliable unless you categorize by AS and type of connectivity.

Akamai

7. Never trust the client clock

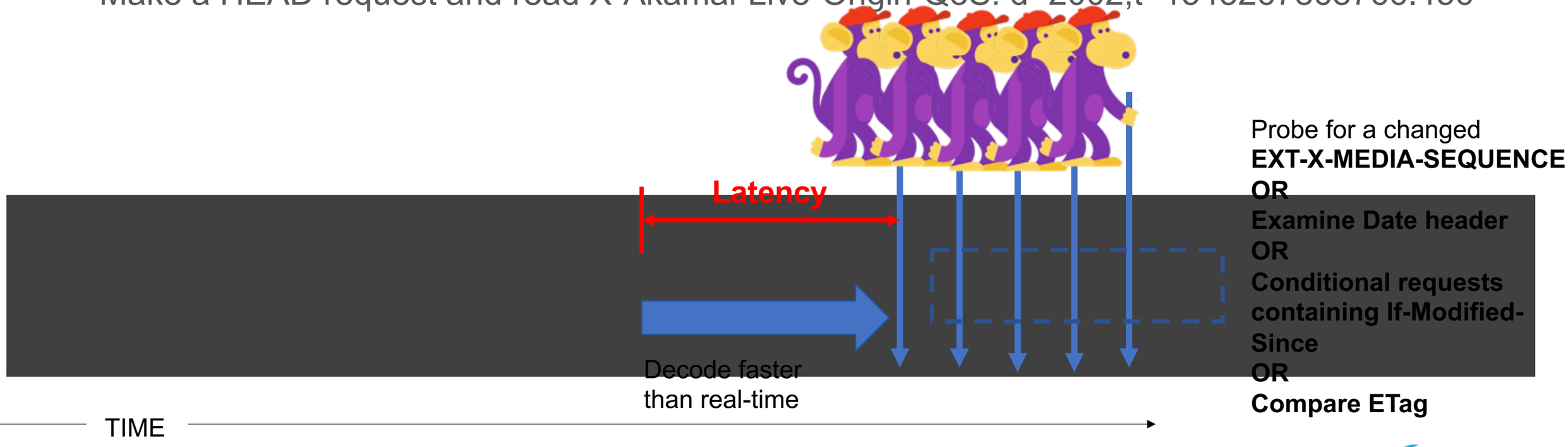
- Use either the time reference provided in the manifest
- Or a default external time source such as http://time.akamai.com/?iso&ms



This is NOT NTP-accurate, due to response-construct time and RTT, but its still better than most client clocks.

8. HLS has an opaque timing model

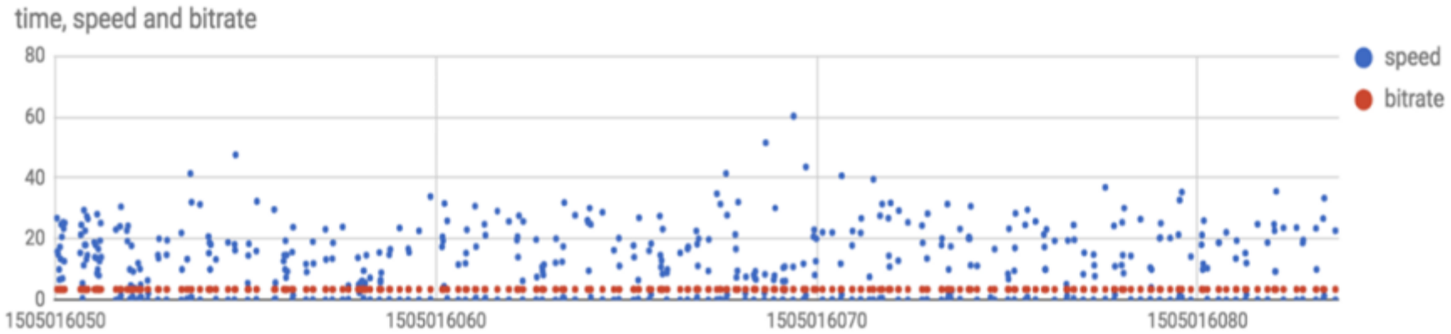
- HLS is deceptively simple – load a media playlist, load the last segment defined, and you have the lowest latency
- If you have 6s segments, then your live latency will vary +/- 6s between players if you randomly request the media playlist
- But when should you load the media playlist for lowest latency?
- Make a HEAD request and read X-Akamai-Live-Origin-QoS: d=2002;t=1543267868766.456



9. Estimating throughput when faced with chunked encoding

Industry has tried

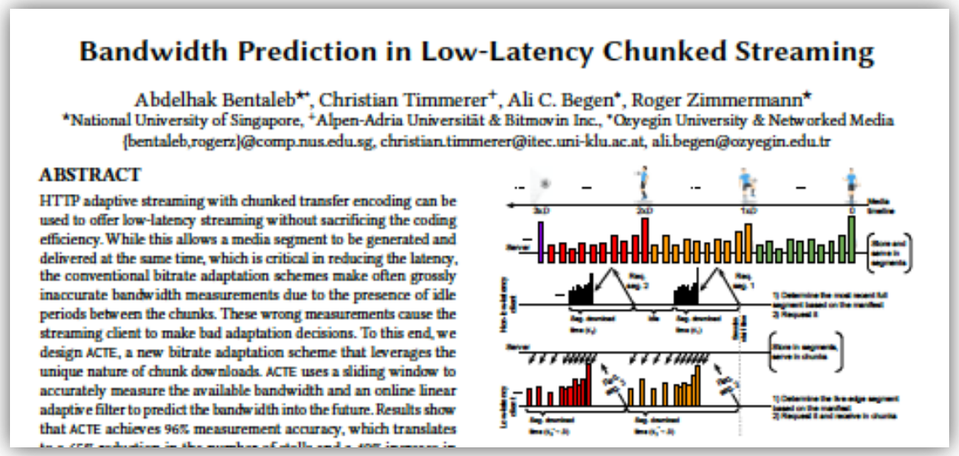
- Machine learning
- Conservative averaging
- Side-loading of test payloads



Akamai

New academic research published in June 2019 at ACM MMSys in Amherst MA.

- Novel method of estimating true throughput when faced with chunked segment transfer
- Discards chunks whose delivery is encode-limited

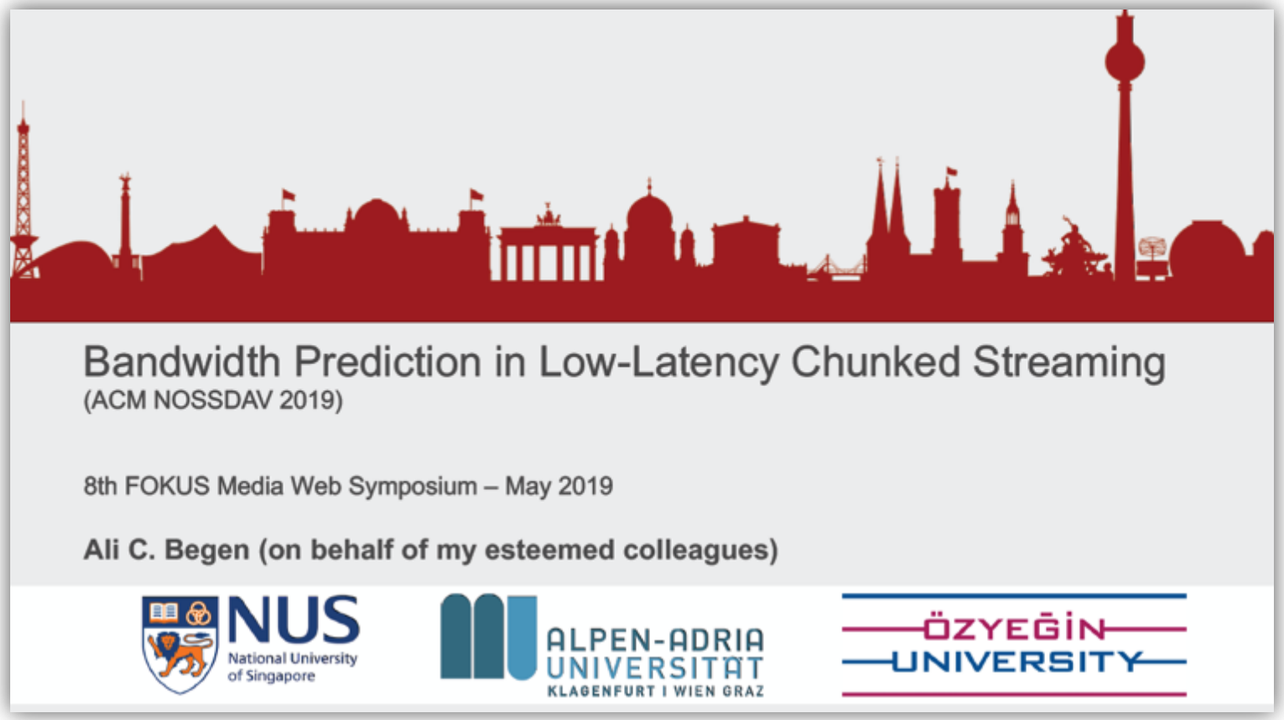


Bandwidth Prediction in Low-Latency Chunked Streaming

Abdelhak Bentaleb*†, Christian Timmerer†, Ali C. Begen*, Roger Zimmermann*
*National University of Singapore, †Alpen-Adria Universität & Bitmovin Inc., *Ozyegin University & Networked Media
{bentaleb,rogerz}@comp.nus.edu.sg, christian.timmerer@itec.uni-klu.ac.at, ali.begen@ozyegin.edu.tr

ABSTRACT

HTTP adaptive streaming with chunked transfer encoding can be used to offer low-latency streaming without sacrificing the coding efficiency. While this allows a media segment to be generated and delivered at the same time, which is critical in reducing the latency, the conventional bitrate adaptation schemes make often grossly inaccurate bandwidth measurements due to the presence of idle periods between the chunks. These wrong measurements cause the streaming client to make bad adaptation decisions. To this end, we design ACTE, a new bitrate adaptation scheme that leverages the unique nature of chunk downloads. ACTE uses a sliding window to accurately measure the available bandwidth and an online linear adaptive filter to predict the bandwidth into the future. Results show that ACTE achieves 96% measurement accuracy, which translates



Bandwidth Prediction in Low-Latency Chunked Streaming
(ACM NOSSDAV 2019)

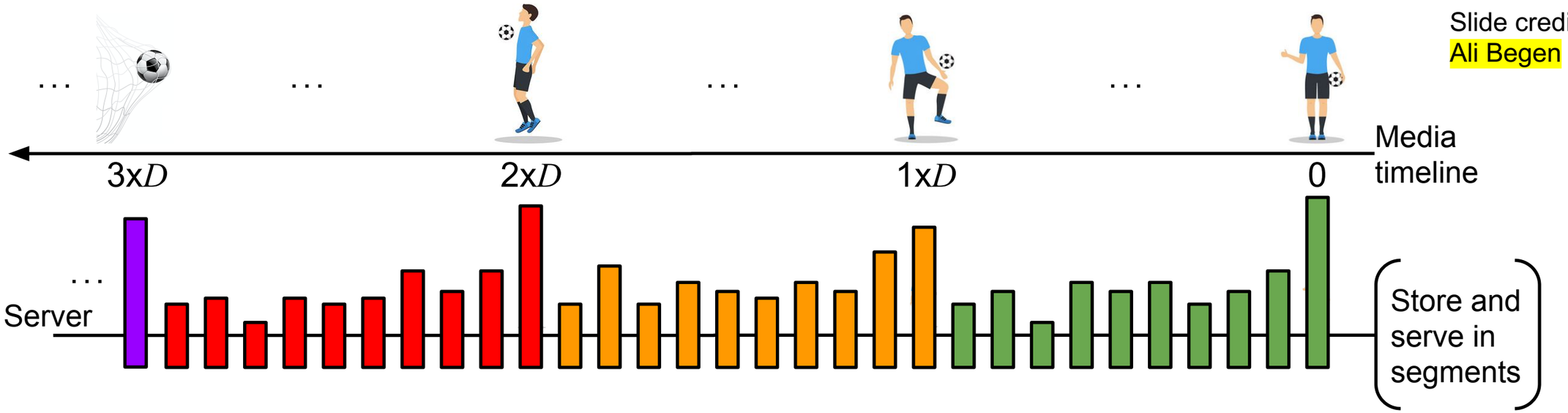
8th FOKUS Media Web Symposium – May 2019

Ali C. Begen (on behalf of my esteemed colleagues)

NUS National University of Singapore

ALPEN-ADRIA UNIVERSITÄT KLAGENFURT I WIEN GRAZ

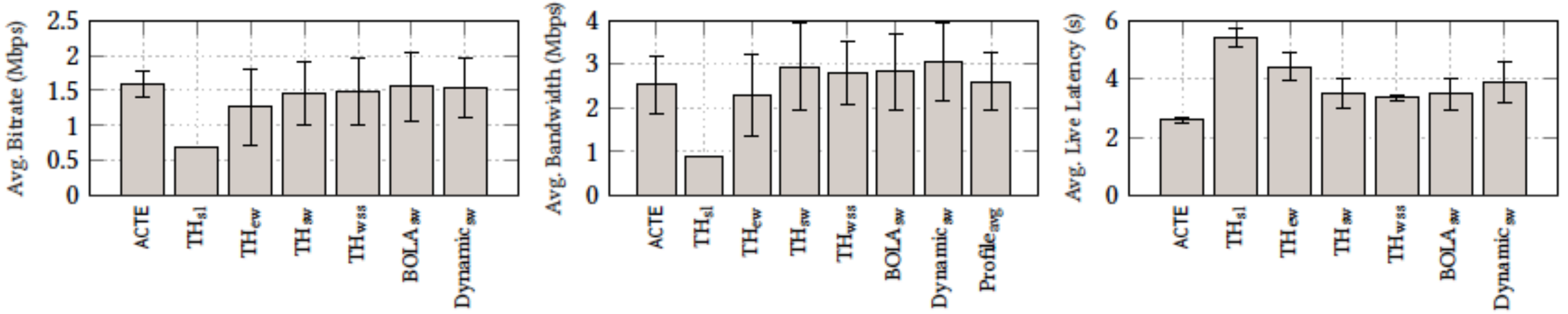
ÖZYEĞİN UNIVERSITY


Slide credit:
Ali Begen
Media
timeline
3xD
2xD
1xD
0
Server
Store and
serve in
segments

Results – (demo at http://bit.do/ACTEdemo)

Table 4: Average bitrate level, live latency, QoE with its metrics, RMSE and prediction accuracy.

	Avg. Selected Bitrate (Mbps)	Avg. Buffer Occupancy (s)	Avg. Live Latency (s)	Avg. # of Switches	Avg. # of Stalls & Duration (s)	Avg. Startup Delay (s)	Avg. N-QoE (N-QoE^{ltu})	Avg. RMSE	Avg. Prediction Accuracy (%)
ACTE	1.4 to 1.7 (1.6)	2.1 to 3.0 (2.5)	2.3 to 3.0 (2.5)	17	3 & 0.76	0.71	0.95 (0.92)	0.033	96.63 %
TH_{sl}	0.7 to 0.7 (0.7)	3.6 to 5.0 (4.3)	5.0 to 5.8 (5.4)	0	2 & 0.86	1.46	0.42 (0.50)	-	
TH_{ew}	0.6 to 1.9 (1.3)	1.9 to 3.9 (2.9)	4.0 to 5.0 (4.5)	18	21 & 66	1.06	0.60 (0.60)	-	-
TH_{sw}	1.0 to 1.9 (1.5)	1.9 to 3.5 (2.8)	3.0 to 4.0 (3.6)	24	27 & 33	1.03	0.76 (0.65)	-	-
TH_{wss}	1.0 to 2.0 (1.5)	2.0 to 3.1 (2.5)	3.1 to 3.3 (3.2)	23	16 & 9	0.88	0.85 (0.78)	-	-
$BOLA_{sw}$	1.1 to 2.1 (1.6)	1.6 to 3.0 (2.3)	3.0 to 4.0 (3.6)	20	58 & 119	1.66	0.65 (0.68)	-	-
$Dymanic_{sw}$	1.2 to 2.0 (1.5)	1.6 to 3.0 (2.4)	3.0 to 5.0 (3.9)	30	53 & 68	0.92	0.74 (0.72)	-	-

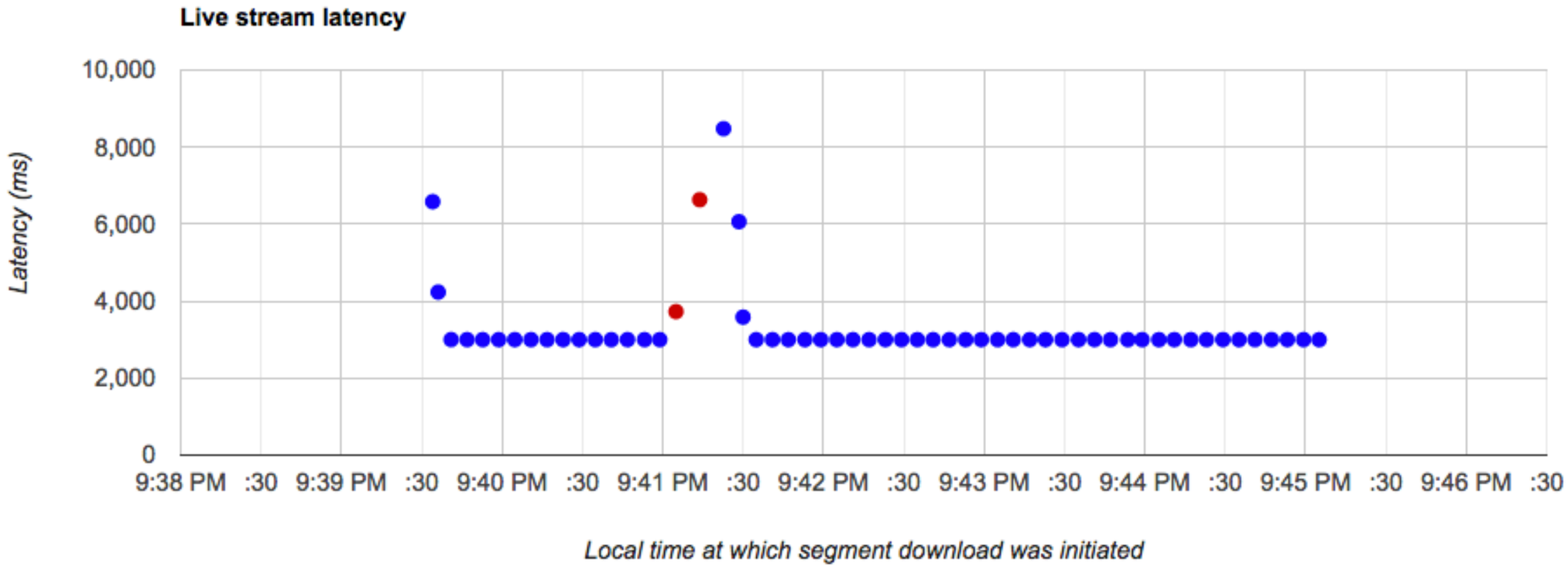


(a) Average selected bitrate. (b) Average measured bandwidth. (c) Average live latency.

Figure 6: Average selected bitrate, measured bandwidth and live latency for different ABR over 10 runs.

10. Make use of catch up and rate adjustment

- Once started, a live player cannot adjust its latency …
- Unless its speeds up 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.
- Allows simple sync between players

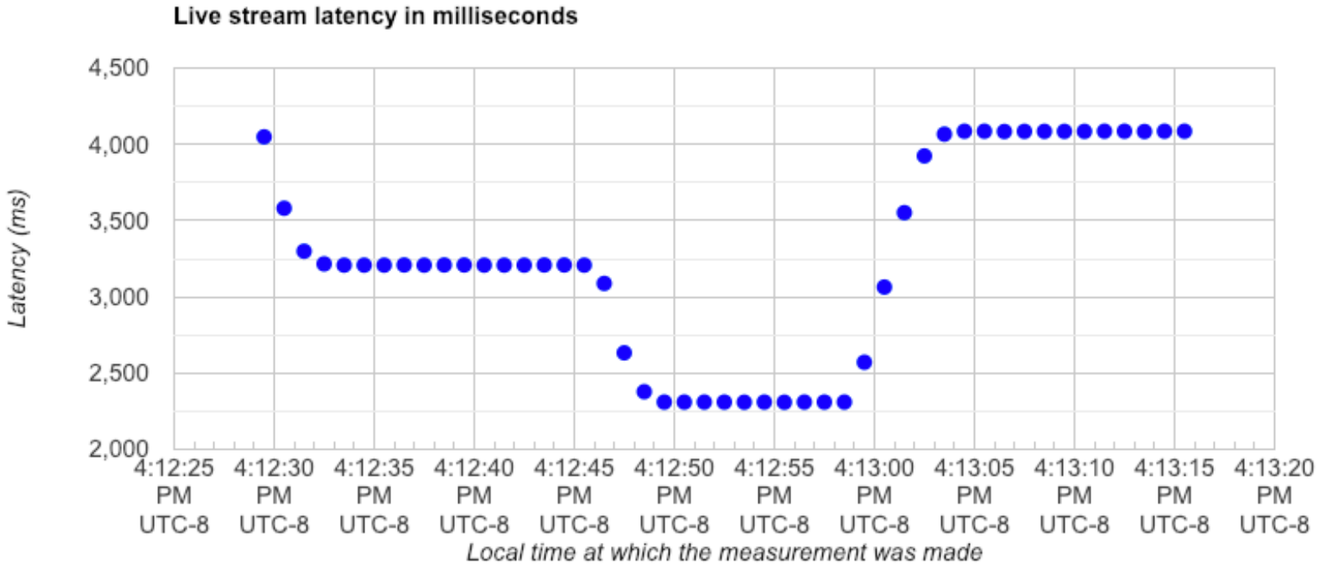


Akamai

11. Live player should never have just one fixed target buffer

The size of the forward buffer should be a function of

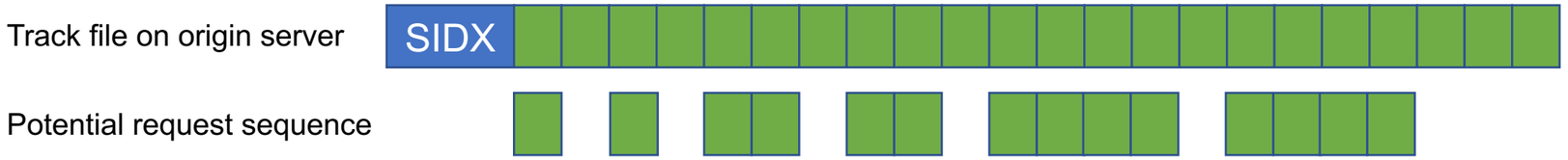
1. The variability in the segment download times (itself a function of RTT, network type, loss, protocol, CPU load etc)
2. The ratio of throughput/bitrate
3. The application target latency



- Keep a record of the segment download times of the last n segments and compute running averages over different time bases.
- Adjust the buffer to protect the player, especially after a buffer-empty

Akamai

12. Variable length range requests for VOD



- You don’t have to fetch one GOP interval at a time.
- If fetching VOD segments using range requests with HLS or DASH, experiment with adjusting the request to maximize throughput. Larger objects encoded at lower bitrates allows TCP to ramp up to higher throughput.
- Additionally, for mobile radio performance – fewer larger burst requests consume less power than a succession of short requests which keep the radio in constant TX mode.

Akamai

13. Don’t randomly pick target segment durations which are integer multiples of seconds

Instead , pick segment duration durations which are INTEGER multiples of both video frames and audio samples. This allows audio and video segments to end at precisely the same time, preventing gaps and oscillating segment durations. http://anton.lindstrom.io/gop-size-calculator/

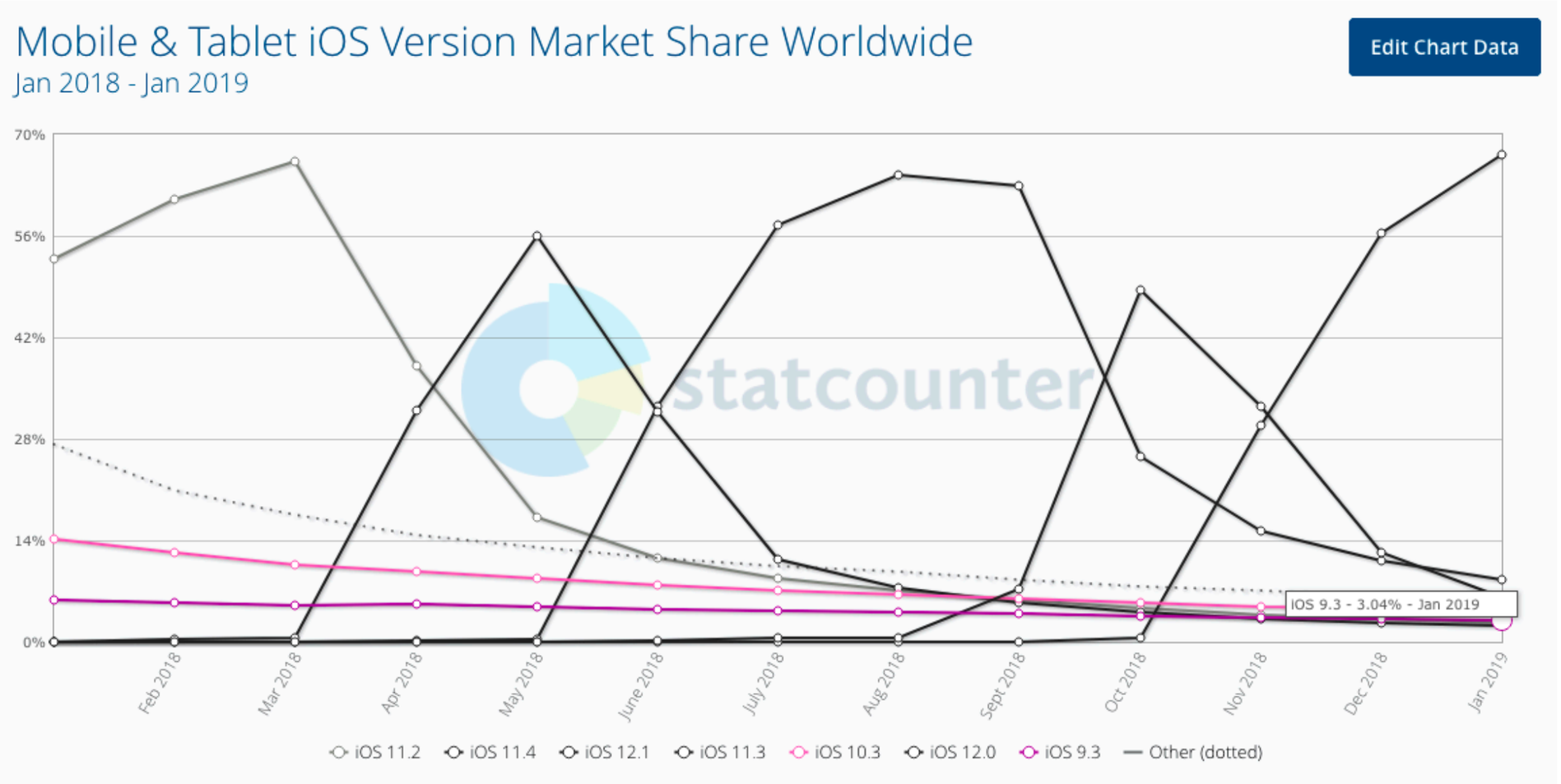
24fps, AAC 48kHz		
Segment duration (seconds)	Video frames	Audio samples
8.00	192	375

25fps, AAC 48kHz		
Segment duration (seconds)	Video frames	Audio samples
1.92	48	90
3.84	96	180
6.40	160	300

30fps, AAC 48kHz		
Segment duration (seconds)	Video frames	Audio samples
1.60	48	75
4.80	144	225
6.40	192	300

Akamai

14. Standardize on CMAF containers for HLS and DASH



Only 3% of IOS devices world-wide are NOT CMAF compliant.

Akamai

15. Implement MultiCDN in the client versus the server

- Only the client has visibility in to the last mile connectivity it is experiencing
- Avoid solutions which only switch once at the start of playback
- The optimum switcher is a per-segment performance-based switch implemented in the client.
  - Pulling non-sequential segments from a CDN hurts prefetching and therefore performance. Try to pull sequential ranges. Do not round-robin between sources.
  - Use a opportunistic greedy algorithm, which assigns new download tasks to the best performing source
  - Given 4 sources [A..D], a good sequence may be AAAABBBBCCCCDDDDBBBBBBBBBBBBAAAABBBBBBBBBCCCCBBBBBBBBBDDDD
  - The first 16 requests establish which source is performing best, then future requests are biased to this source, with repeated trials of the other sources to see if the status quo has changed.

Akamai

Thank you.

Questions?

Akamai