LL-HLS
LHLS
DASH-LL

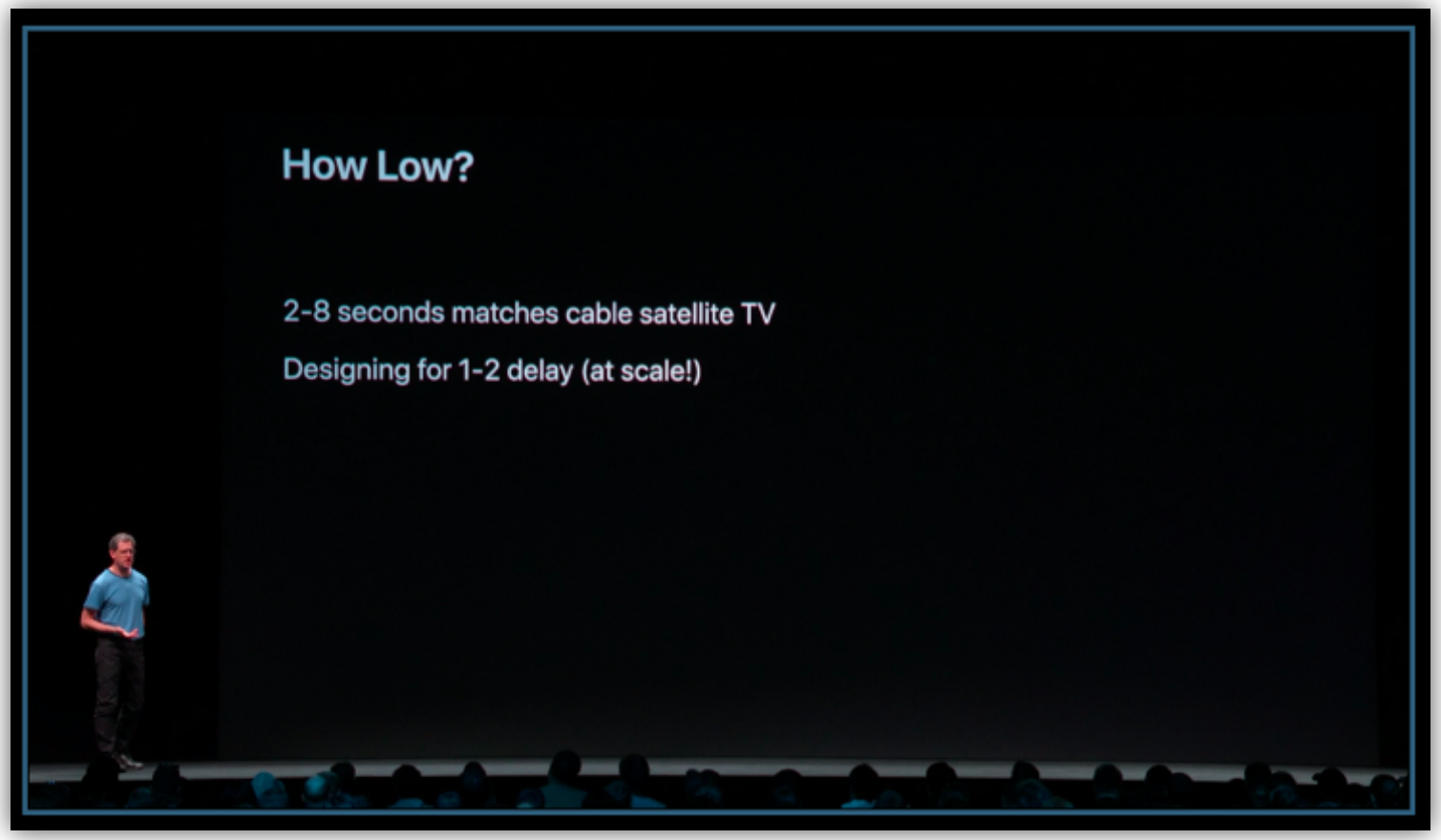
Challenges and differences

Will Law | Chief Architect | June 2019

Akamai
Experience the Edge

WWDC – San Jose – June 5, 2019

- Roger Pantos announced Low-Latency HLS (LL-HLS)
  https://developer.apple.com/videos/play/wwdc2019/502/



LHLS proposal – April 2018 – FOMS NYC

John Bartos proposal https://github.com/video-dev/hlsjs-rfcs/pull/1/commits

- #EXTM3U
- #EXT-X-VERSION:3
- #EXT-X-TARGETDURATION:2
- #EXT-X-PROGRAM-DATE-TIME:2018-09-05T20:59:06.531Z
- #EXTINF:2.000
- https://foo.com/bar/0.ts
- #EXT-X-PROGRAM-DATE-TIME:2018-09-05T20:59:08.531Z
- #EXTINF:2.000
- https://foo.com/bar/1.ts
- #EXT-X-PREFETCH:https://foo.com/bar/2.ts
- #EXT-X-PREFETCH:https://foo.com/bar/3.ts

DASH Low Latency – June 2019

https://dash-industry-forum.github.io/docs/DASH-IF-IOP-CR-Low-Latency-Live-Community-Review.pdf

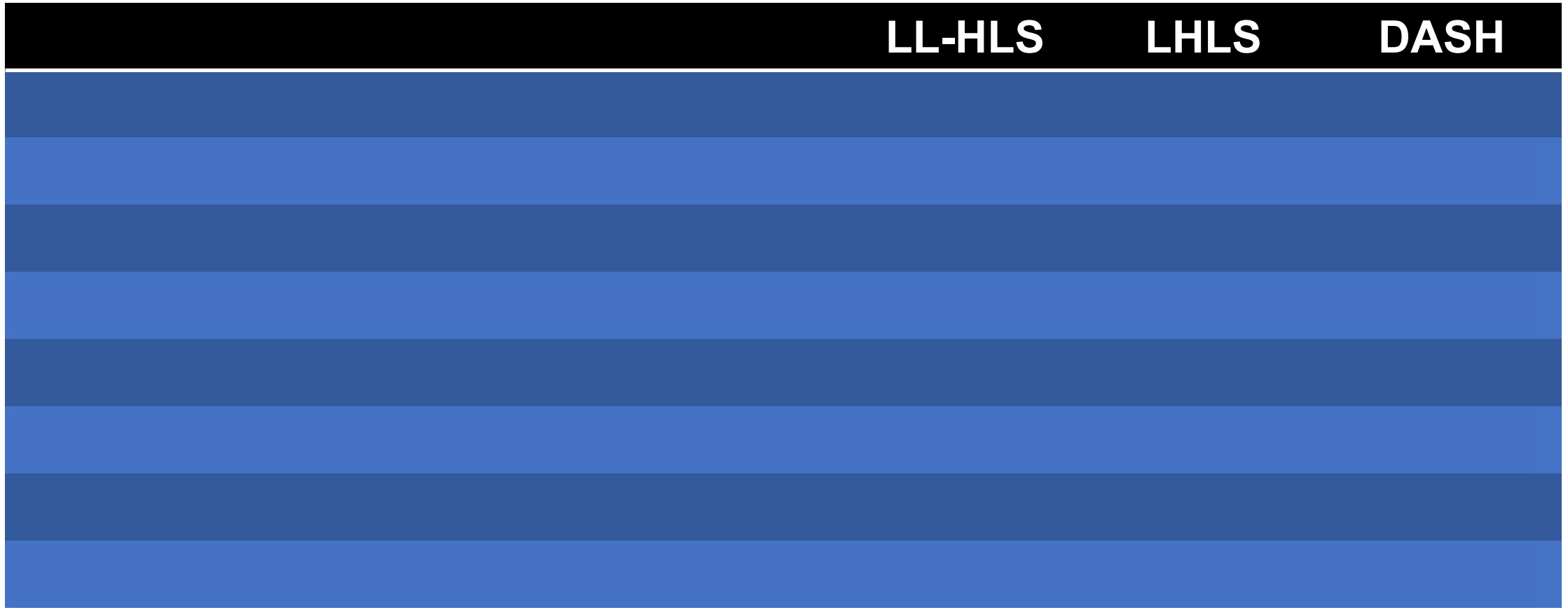
Title:	Low-latency Modes for DASH
Source:	Live TF
Supporting Companies:	Akamai, castLabs, Comcast, Elemental Technologies, Ericsson, Harmonic, Qualcomm Incorporated, Sony, TNO, Unified Streaming, Frontier Communications
Category:	**A** ***Date:*** 2019-06-28 *Use one of the following categories:* ***C*** *(correction)* ***A*** *(addition of feature)* ***B*** *(editorial modification)*
Reason for change:	DASH-IF collected information related to Low-Latency Streaming in a Report, together with DVB. The report (available here: https://dash-industry-forum.github.io/docs/Report%20on%20Low%20Latency%20DASH.pdf) provides use cases, service scenarios, deployment experience and existing potential technologies. Also, DASH-IF already generated the DASH profile for ATSC that includes a mode supporting low latency. Based on this information in the report, it is recommended to add low latency to DASH-IF IOP Guidelines.
Summary of change:	This change provides a new clause for live services that addresses specification updates as well as implementation guidelines to support Low-Latency DASH services addressing the requirements above.

DASH Industry Forum

Digital Video Broadcasting

Commonalities

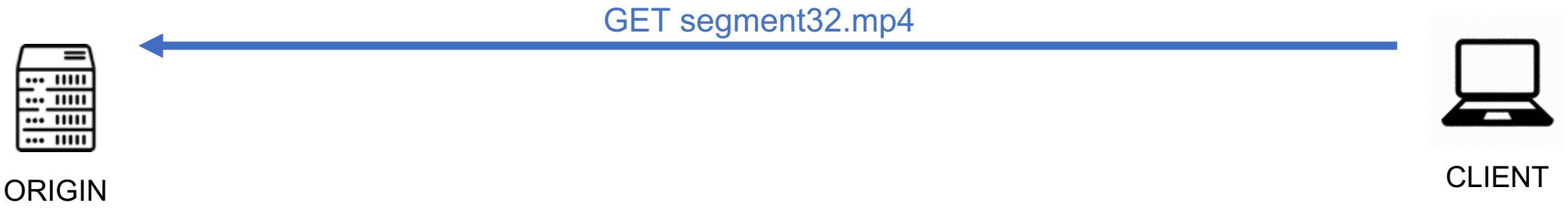
Differences



Chunked transfer encoding (CTE)

- LHLS and DASH-LL use chunk encoded transfer to release the segment from the origin, through the CDN and to the client.
- Benefit:
  - Only requires one request per segment
  - Delivers the sequential data in the correct order, as fast as it is produced
  - Allows small chunks to be used (down to 1 frame, or 30 chunks per second)
- Problem
  - Data speed is encoder-limited, versus line-speed limited. This makes it difficult to estimate the throughput by timing the receipt of the object.

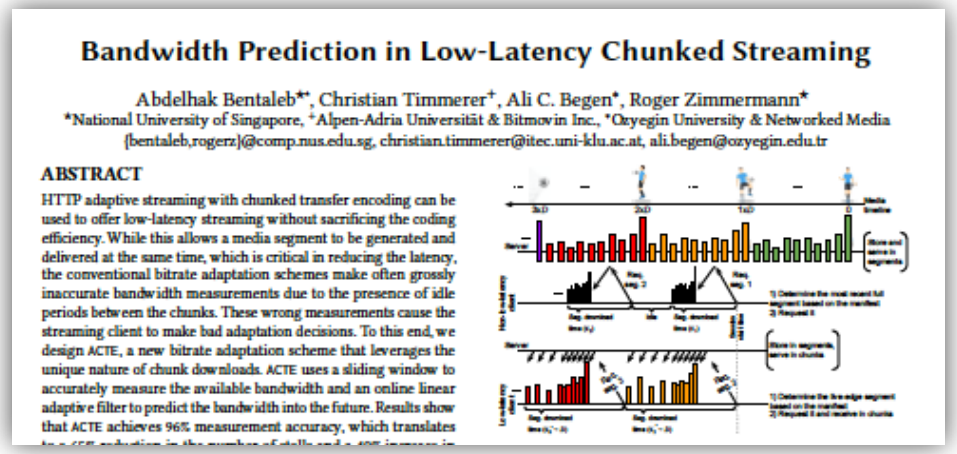
CTE



NON-CTE

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

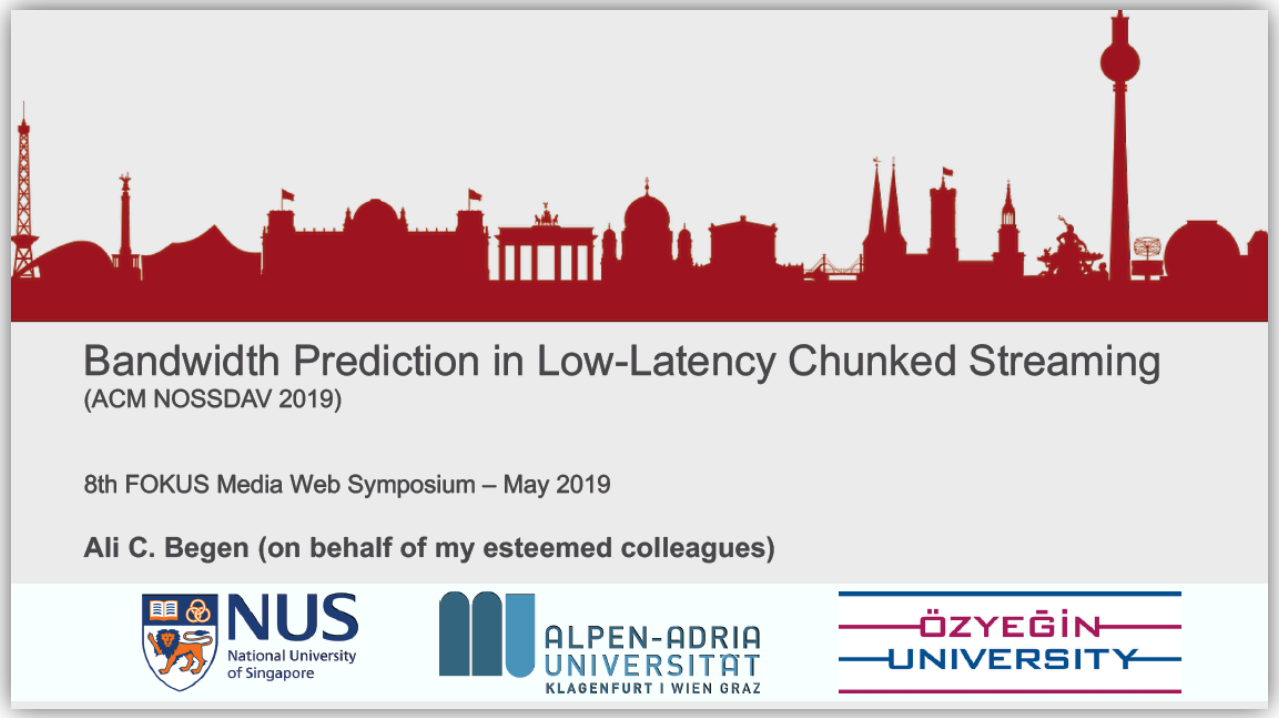


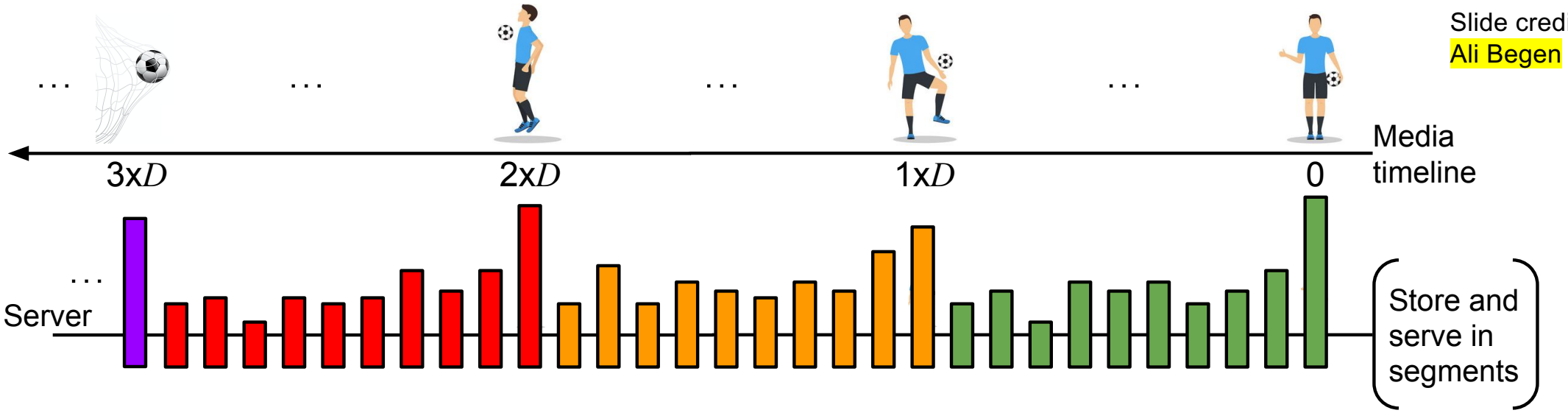
Bandwidth Prediction in Low-Latency Chunked Streaming

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

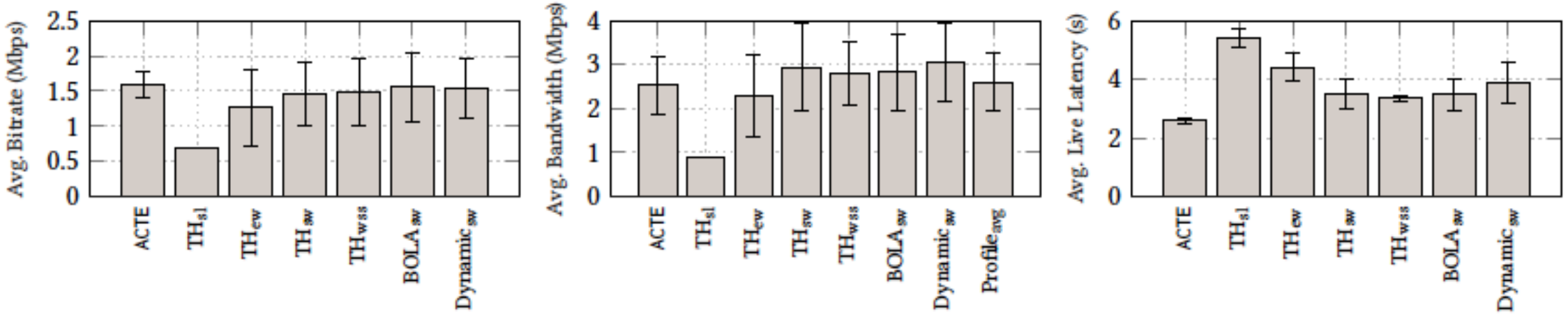



Slide credit:
Ali Begen
Media
timeline
3xD
2xD
1xD
0
Server
Store and
serve in
segments

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^{itu})	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.

Why not use CTE?

- At its optimum, a LL-HLS player would request chunks the moment they are produced. It will also request them in the order they are encoded, since this is the order in which they must be decoded.
- CTE gives you this for free. A player makes one request for the segment and then receives the chunks in the correct order and as soon as they are produced by the encoder.
- Since the major downside of CTE is bandwidth estimation and this now has a solution, it is possible that the reason LL-HLS chooses to ignore CTE is primarily for intellectual property (IP) reasons.

Why have internal segment descriptions?

- Player can understand where the IDR's are at any point in time
- This improves start-up logic
- Also allows for switching at IDR boundaries instead of segment boundaries. (Every 2s instead of every 6s for example)
- What is the point of describing non-independent chunks?

  To allow latency less than the IDR spacing, when you are subject to the HLS doctrine that objects must be fully available when added to the media playlist.

Why describe individual parts?

LL-HLS with part descriptions

#EXT-X-Part:DURATION=0.33334,URI="filePart271.11.ts"

#EXTINF:4.00008,

fileSequence271.ts

#EXT-X-PROGRAM-DATE-TIME:2019-02-14T02:13:60.106Z

#EXT-X-PART:DURATION=0.33334,URI="filePart272.a.ts",INDEPENDENT=YES

#EXT-X-PART:DURATION=0.33334,URI="filePart272.b.ts"

#EXT-X-PART:DURATION=0.33334,URI="filePart272.c.ts"

#EXT-X-PART:DURATION=0.33334,URI="filePart272.d.ts"

#EXT-X-PART:DURATION=0.33334,URI="filePart272.e.ts"

#EXT-X-PART:DURATION=0.33334,URI="filePart272.f.ts",INDEPENDENT=YES

#EXT-X-PART:DURATION=0.33334,URI="filePart272.g.ts"

#EXT-X-PART:DURATION=0.33334,URI="filePart272.h.ts"

#EXT-X-PART:DURATION=0.33334,URI="filePart272.i.ts"

1.3s of buffer

2.1s of buffer

LL-HLS without part descriptions

#EXTINF:4.00008,

fileSequence271.ts

2.7s behind live, or 6.7s if prior segment is loaded.

Starting buffer requirement determines starting position

How do the other formats deal with this starting problem?

LHLS

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:2
#EXT-X-PROGRAM-DATE-TIME:2018-09-05T20:59:06.531Z
#EXTINF:2.000
https://foo.com/bar/0.ts
#EXT-X-PROGRAM-DATE-TIME:2018-09-05T20:59:08.531Z
#EXTINF:2.000
https://foo.com/bar/1.ts
#EXT-X-PREFETCH:https://foo.com/bar/2.ts
```

- The player will load this final segment. It will have anywhere from 0 – 2s of data.
- It can only discover how much data after downloading and parsing the segment.
- It may have to wait for more data to arrive before starting.

If they need more than 2s of buffer, they would start by loading the prior segment

Possible LL-HLS media playlist variations

BEFORE v9 (or LHLS)

- video_800kbps.m3u8

AFTER

- video_800kbps.m3u8?_HLS_push=1&_HLS_msn=273&_HLS_part=3
- video_800kbps.m3u8?_HLS_push=1&_HLS_msn=273&_HLS_part=3&_HLS_report=../1M/waitForMSN.php&_HLS_report=../4M/waitForMSN.php&_HLS_skip=NO
- video_800kbps.m3u8?_HLS_push=1&_HLS_msn=273&_HLS_part=3&_HLS_report=../1M/waitForMSN.php&_HLS_skip=YES
- video_800kbps.m3u8?_ .php&_HLS_skip=YES
- video_800kbps.m3u8?_ .php&_HLS_skip=YES
- video_800kbps.m3u8?_ .php&_HLS_skip=NO
- video_800kbps.m3u8?_
- video_800kbps.m3u8?_
- video_800kbps.m3u8?_
- video_800kbps.m3u8?_ .php&_HLS_skip=NO
- video_800kbps.m3u8?_
- video_800kbps.m3u8?_HLS_push=1&_HLS_msn=273&_HLS_part=3&_HLS_report=../2M/waitForMSN.php&_HLS_report=../4M/waitForMSN.php&_HLS_skip=YES
- video_800kbps.m3u8?_HLS_push=1&_HLS_msn=273&_HLS_part=3&_HLS_report=../2M/waitForMSN.php&_HLS_report=../5M/waitForMSN.php&_HLS_skip=YES
- …

For a stream with 18 parts per segment, and a mixture of clients that request PUSH and those that don't, as well as those that skip and those that don't, there could easily be 72 versions of a LL-HLS media playlist.

Request rates compared for a 6s segment with 333ms chunks/parts.

DASH-LL

20 requests/min

LHLS

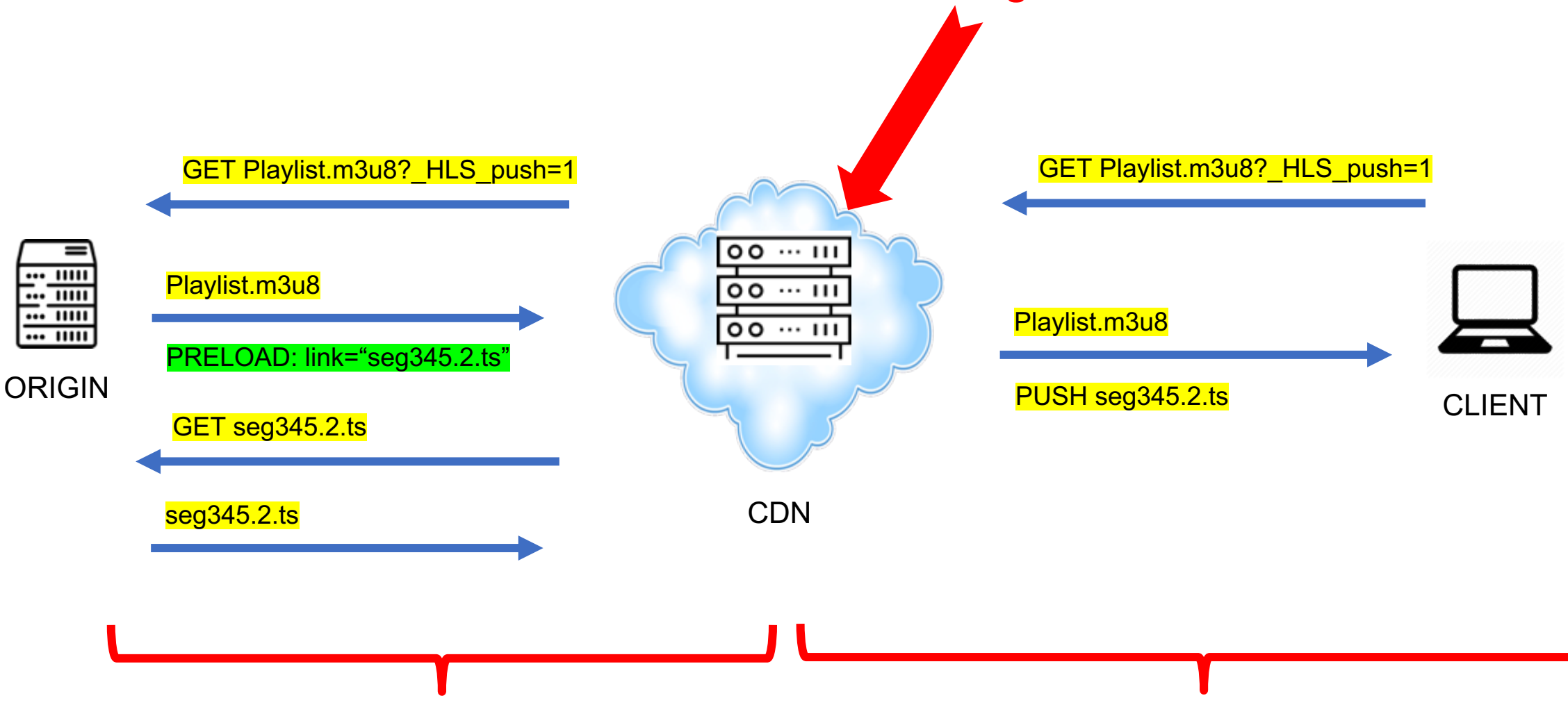
40 requests/min

LL-HLS

720 requests/min

H2 PUSH with LL-HLS

The same physical server must deliver both playlists and segments.



Time saved is these two RTTs

Significant changes that come with LL-HLS

- Origin intelligence due to mutable playlists via query args. _HLS prefix is reserved.
- Large increase in request rate per client
- To implement PUSH, same edge server must deliver media playlists and segments. This is large architectural problem for server-side ad insertion solutions, where typically an ad-stitching platform delivers the modified playlists while a CDN (or multiple CDNs) deliver the media segments.
- Requirement for H2 push at CDN edge - not many media CDNs support this feature today.

Conclusions

- In analysis, it seems as though LL-HLS is designed somewhat to avoid IP that could read against either LHLS or DASH-LL solutions.
- There is a interoperable workable solution for LHLS/DASH-LL for those providers who are willing to write an app for iOS/TVOS/MacOS.
- The requirement to have the same physical server deliver playlists and segments will hamper the deployment of LL-HLS with SSAI solutions.
- Using contiguous 2s segments gets you broadcast equivalence in the US, for a lot less complexity ☺

Questions at the end ☺

Thanks for your time.

Akamai
Experience the Edge