Performance of Low-Latency DASH/CMAF and Low-Latency HLS Streaming Systems

Bo Zhang, Thiago Teixeira, Yuriy Reznik
Brightcove, Inc.

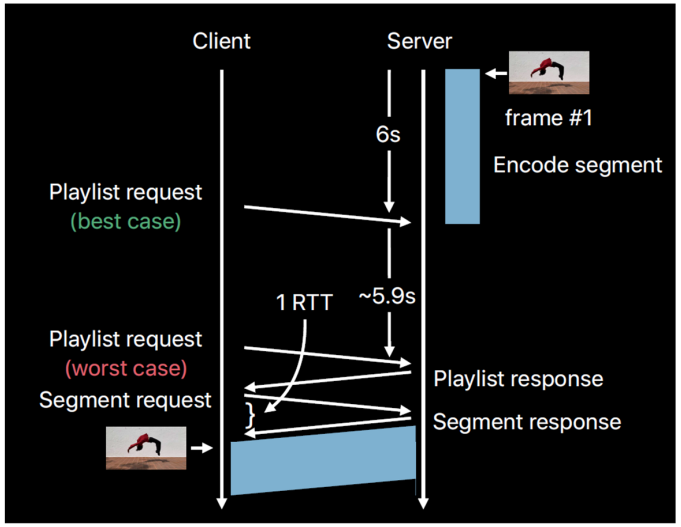
Outline

- Context & Objectives
- Evaluation Framework
- The Results
- Conclusions

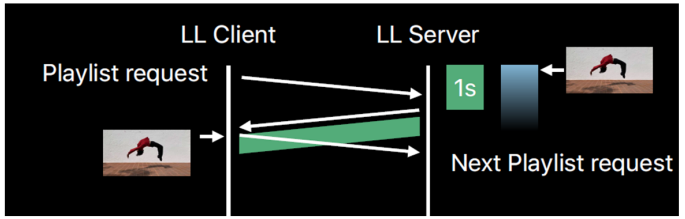
Context & Objectives

- Current state of HTTP-based streaming:
  - HLS and DASH work, but introduce significant (~10..30sec) delays
  - Delays are caused by long segments and transmission protocols
  - LL-DASH / ULL-CMAF and LL-HLS are 2 major new technologies
- Key design principles of LL-systems:
  - Server: encode and push video in smaller chunks (e.g. 1sec)
  - Client: request a segment as soon as first chunk becomes available
  - Desired end-to-end delay: <5 sec
- Objectives of this work:
  - Evaluate the existing implementations of LL-HLS/DASH systems
  - Understand tradeoffs between the delay and other QOE factors
  - Understand how mature implementations of LL-HLS/DASH systems are currently

Delays in traditional HLS streaming



Delays in LL-HLS streaming



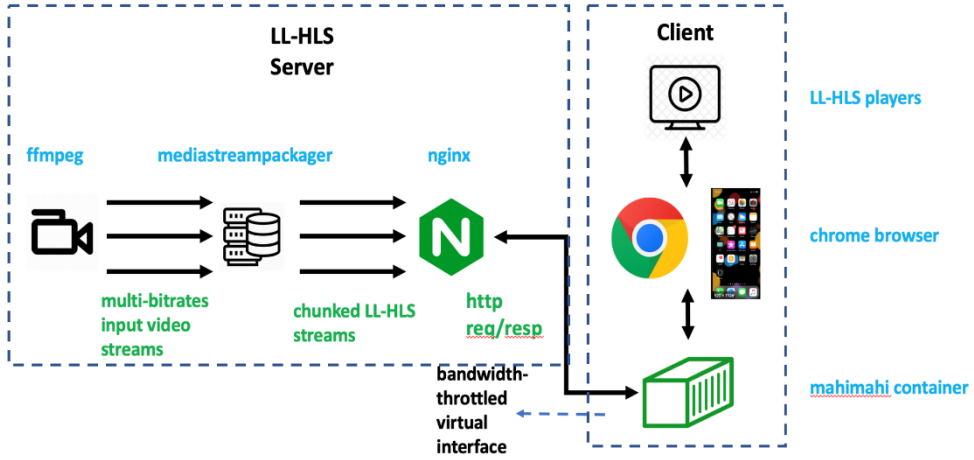
Source: Apple, Inc. WWDC19

Evaluation Framework

Streaming systems

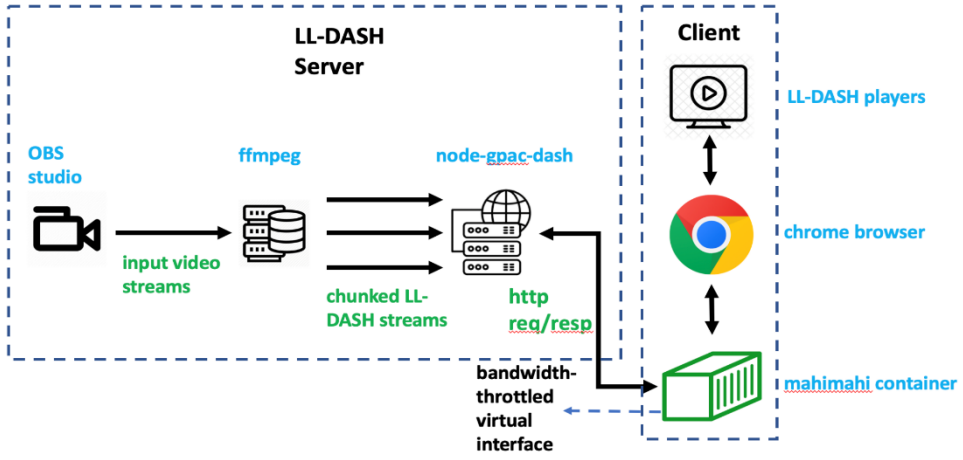
LL-HLS streaming system

- FFmpeg and Apple's HLS reference tools
- AVplayer, HLS.js, and Shaka players



LL-DASH streaming system

- FFmpeg and node-gpac-dash
- DASH.js (+ LoL and L2All) and Shaka players



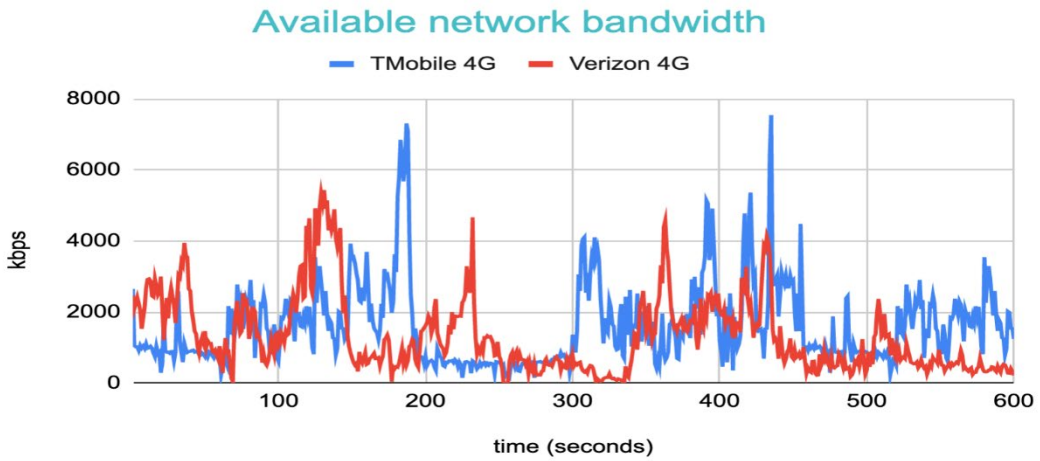
References:

1. B. Zhang, T. Teixeira, Y. Reznik, "Performance of Low-Latency HTTP-based Streaming Players," ACM MMSys'2021
2. B. Zhang, "Setting up your Own Low-Latency HLS Server to Stream from any Source Inputs", https://bozhang-26963.medium.com
3. B. Zhang, "Low-Latency DASH Streaming Using Open-Source Tools", https://bozhang-26963.medium.com
4. M. Lim, et al, "When they go high, we go low: low-latency live streaming in dash.js with LoL," ACM MMSys'2020
5. T. Karagkioules, et al, "Online Learning for Low-Latency Adaptive Streaming," ACM MMSys'2020

Network Emulation

Network models

- Mahimahi network simulator is used to simulate network behaviors
- Two network traces are used for analysis: Verizon 4G LTE network, and T-Mobile 4G LTE



Bandwidth statistics

Network	T-Mobile	Verizon
Average (kbps)	1607.43	1323.97
Variation (kbps)	1147.60	1075.80
Min. (kbps)	148.5	1.178
Max. (kbps)	7545	5433

References

1. R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, H. Balakrishnan, “Mahimahi: accurate record-and-replay for HTTP,” USENIX Annual Technical Conference, Santa Clara, CA, July 8-10, 2015

Test Content & Encoding

Content

- “Big Buck Bunny” video clip

Encoding profile

Parameter	Rendition 1	Rendition 2	Rendition 3
Bitrate (kbps)	279	925	1253
Frame rate (fps)	30	30	30
Video resolution (pixels)	320x180	640x360	768x432
Seg. duration (sec)	4	4	4
Chunk duration (sec)	1	1	1
Video codec	H.264	H.264	H.264
Video codec profile	Baseline	Baseline	Baseline
Media format	ISOBMFF	ISOBMFF	ISOBMFF

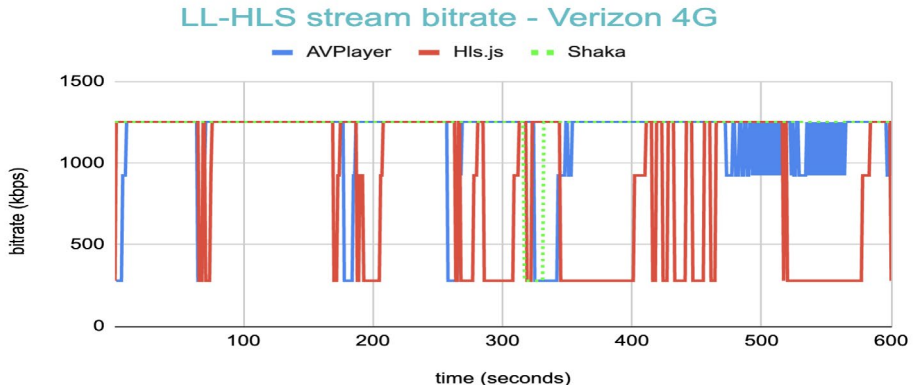
Notes

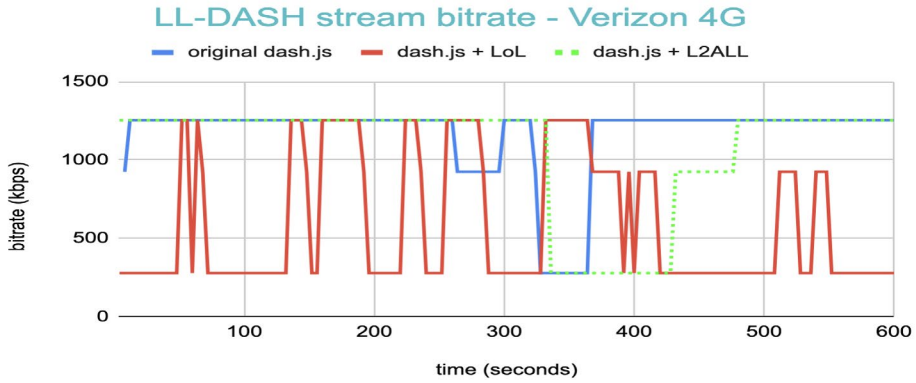
- All encodings are CBR
- Range of rendition bitrates well covers dynamic range of bitrates in both Verizon & T-Mobile networks
- Session durations used for analysis: 10min

Test Results

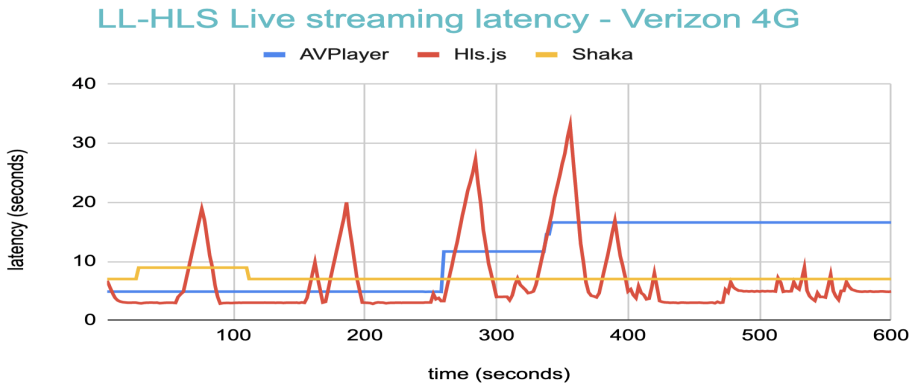
Results for Verizon 4G LTE network

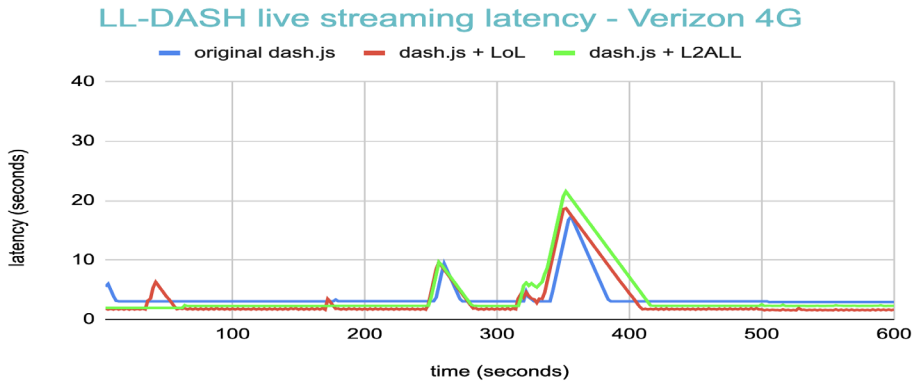
Bitrate variation over time





Latency variation over time





Results for Verizon 4G LTE network

Summary statistics

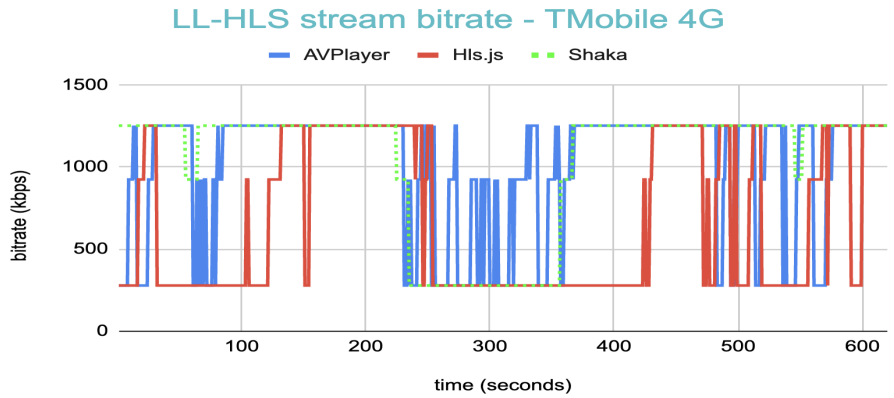
Metrics	LL-HLS players			LL-DASH players		
	HLS.js	Shaka	AVplayer	DASH.js	LoL	L2All
Avg. bitrate (kbps)	783	1043	1037	1225	537	1251
Avg. height (pixels)	311	378	378	426	248	432
Avg. latency (secs)	5.82	4.48	7.78	3.06	1.78	2.28
Var. playback speed	3.62	0	0	0.23	1.62	0.42
# of switches	50	8	72	4	28	0
# of rebufferings	43	18	1	1	69	13
Downloaded MBs	156	81	92	93	42	94
Downloaded objects (chunks + segments)	965 (743+222)	621 (621+0)	703 (698 +5)	151	152	151

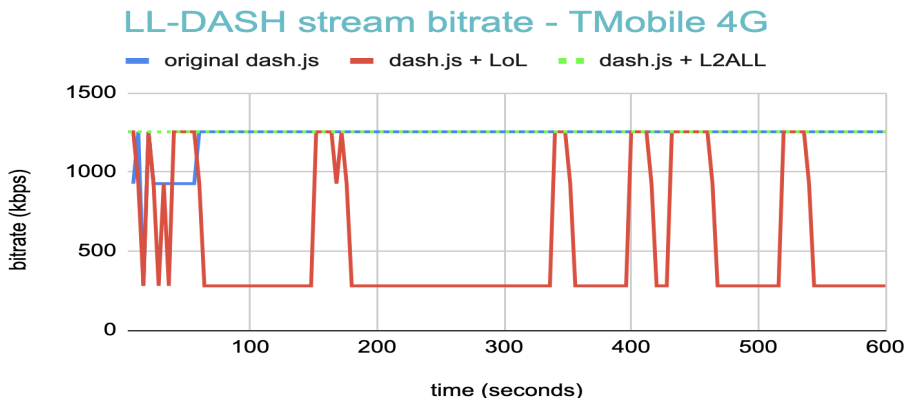
Key observations

- Both LL-HLS and LL-DASH buffer and switch a lot!
- AVplayer is most robust among LL-HLS players; DASH.js is most robust among LL-DASH players
- LL-DASH players achieve lover average latency
- LL-HLS players load many more objects

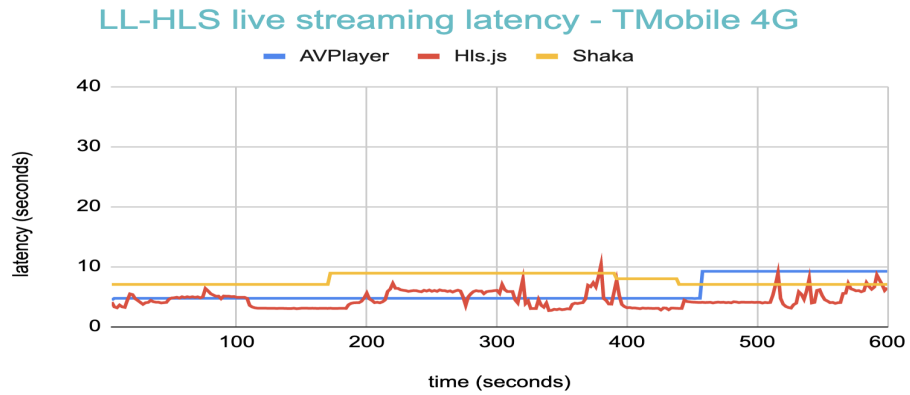
Results for T-Mobile 4G LTE network

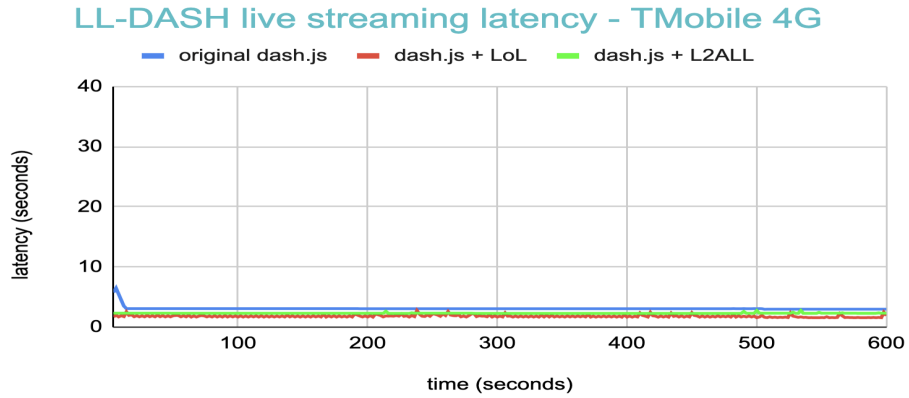
Bitrate variation over time





Latency variation over time





Results for T-Mobile 4G LTE network

Summary statistics

Metrics	LL-HLS players			LL-DASH players		
	HLS.js	Shaka	AVplayer	DASH.js	LoL	L2All
Avg. bitrate (kbps)	849	1228	1136	1165	595	1073
Avg. height (pixels)	328	426	404	410	262	387
Avg. latency (secs)	4.32	7.28	15.96	3.71	3.2	3.9
Var. playback speed	3.97	0	0	0.19	0.39	0.44
# of switches	48	2	130	6	29	3
# of rebufferings	36	12	2	5	79	56
Downloaded MBs	85	90	99	88	45	81
Downloaded objects (chunks + segments)	673 (662+11)	587 (587+0)	669 (611+58)	152	151	152

Key observations

- Fewer rebuffering events observed, but still many switches! Especially with AVPlayer
- AVplayer is most robust among LL-HLS players; DASH.js is most robust among LL-DASH players
- LL-DASH players achieve lover average latency
- LL-HLS players load many more objects

Conclusions

Conclusions

Both LL-HLS and LL-DASH reduce delays

- 3-7 sec end-to-end streaming latency is achievable today

Observed drawbacks

- *Reliability* – number of buffering events
- *Scalability and delivery costs* – increased intensity of data exchanges with origin servers and CDNs (particularly with LL-HLS)
- *Consistency of experience* – increased number of switches, delay variability, etc.

Some aspects can be improved

- Frequency of switches and buffering can possibly be minimized with smarter algorithms in streaming clients; each year we see new algorithms at MMSys!
- But clearly, there is still considerable work ahead!

THANK YOU!

Yuriy Reznik
Brightcove, Inc.
yreznik@brightcove.com