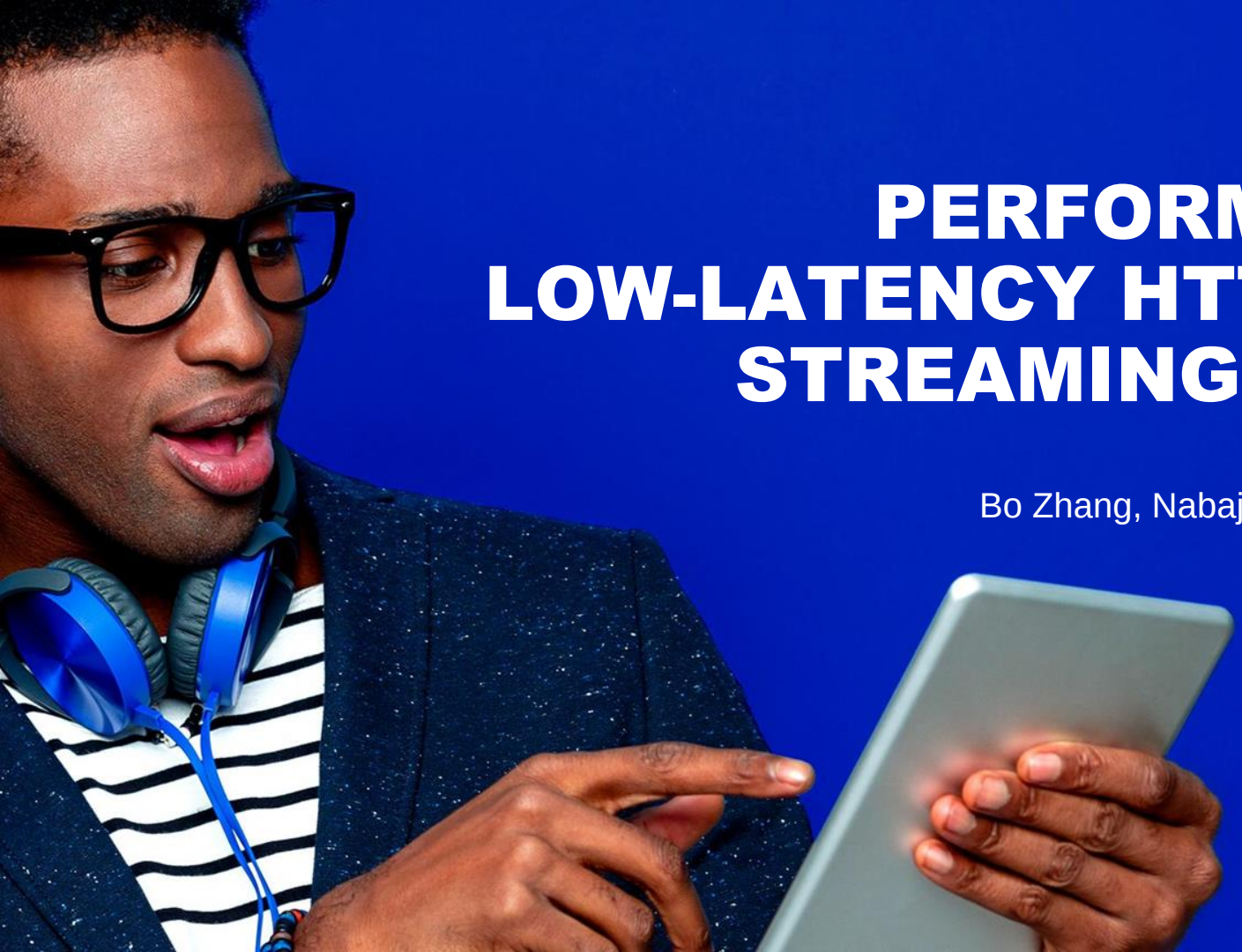
BRIGHTCOVE



PERFORMANCE OF LOW-LATENCY HTTP-BASED STREAMING PLAYERS

Bo Zhang, Nabajeet Barman, Yuriy Reznik,
Brightcove

May 7-10, 2023, Denver, CO

BRIGHTCOVE®

Outline

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

BRIGHTCOVE

Context & Objectives

Current state of HTTP-based streaming:

- HLS and DASH work, but introduce significant (~30 secs) delays
- Delays are caused by long segments, transmission protocols and player side buffering
- LL-DASH and LL-HLS are 2 major new technologies

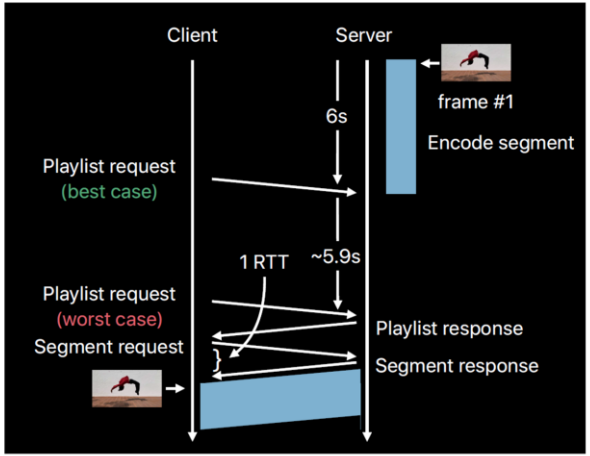
Key design principles of LL-systems:

- Server: encode and push video in smaller chunks (e.g. 1 sec)
- Client: request a segment as soon as first chunk becomes available
- Desired end-to-end delay: < 5 secs

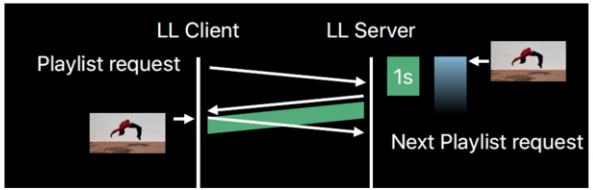
Objectives of this work:

- 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

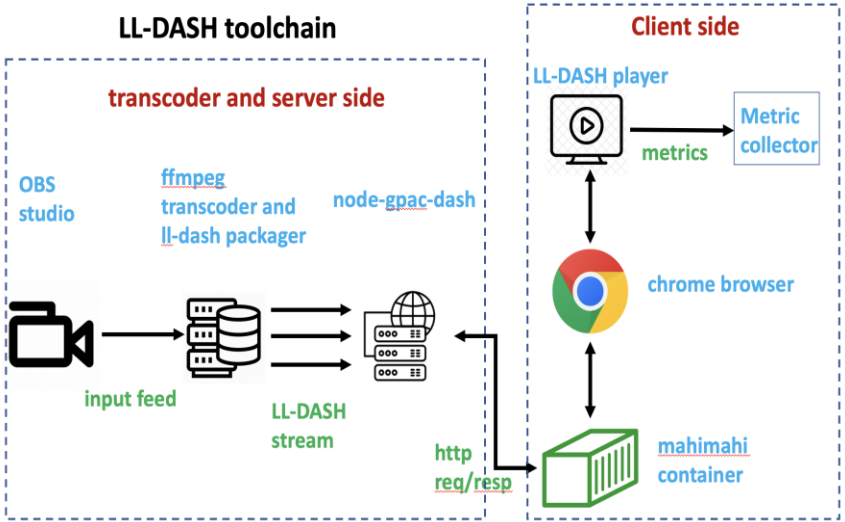
BRIGHTCOVE®

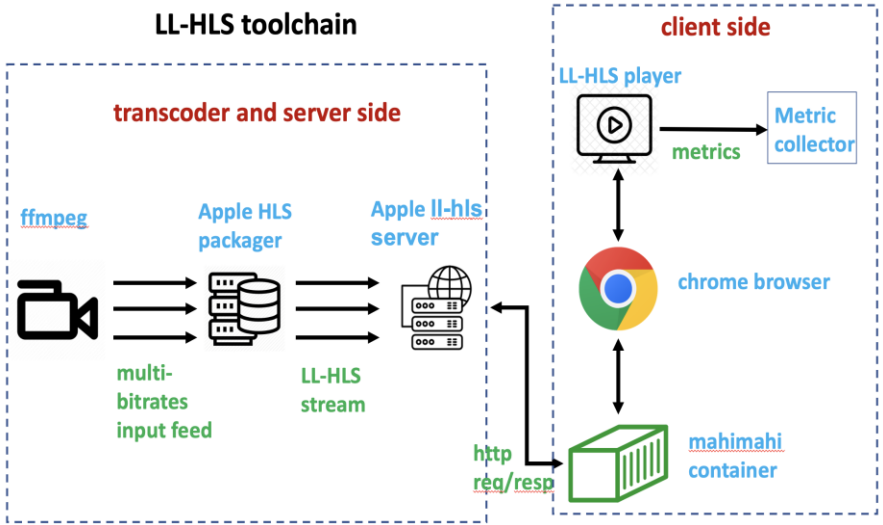
Evaluation Framework

Streaming systems

BRIGHTCOVE

LL-DASH	LL-HLS
OBS studio, FFmpeg and node-gpac-dash	FFmpeg and Apple's HLS tools
DASH.js (+ LoL and L2All), Shaka players, Theo player	HLS.js, and Shaka players





BRIGHTCOVE®

Experiment setup

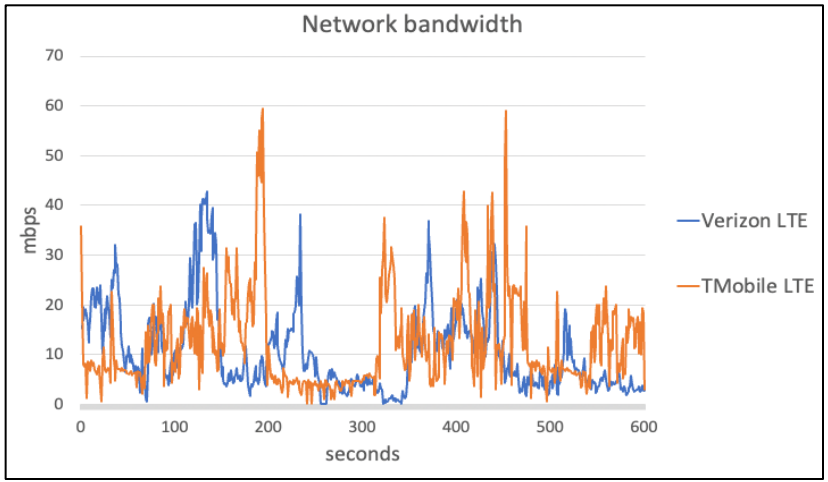
- Test stream: Big buck bunny looped for each 10 minutes streaming session.
- 4 live encoding profiles

Parameter	Low	Mid	High	Top
Codec	H.264	H.264	H.264	H.264
Video resolution	768 x 432	1024 x 576	1600 x 900	1920 x 1080
Video framerate (fps)	30	30	30	30
Bitrate (kbps)	949	1854	3624	5166

- Notes:
  - Media segment duration = 4 seconds, media chunk duration = 1 second
  - All encodings are CBR
  - Range of rendition bitrates well covers dynamic range of bandwidth in both Verizon & T-Mobile networks

Network emulation

- Mahimahi network emulator is used to emulate real-world network condition.
- Two network traces are used for analysis: T-Mobile 4G LTE network and Verizon 4G LTE network



Bandwidth statistics

Network	T-Mobile	Verizon
Average (kbps)	12258	10565
Variation (kbps)	9314	8619
Min. (kbps)	12	12
Max. (kbps)	59460	42804

- Reference

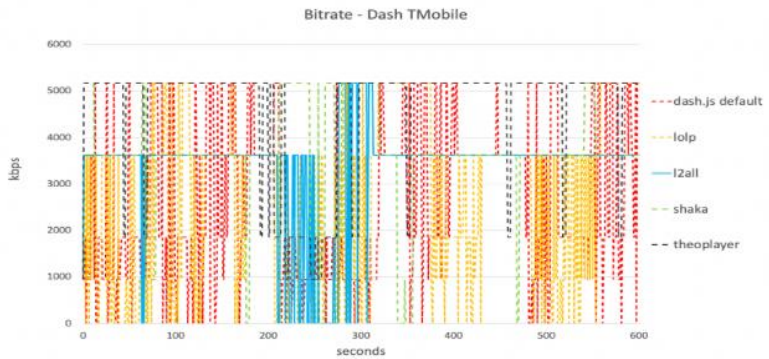
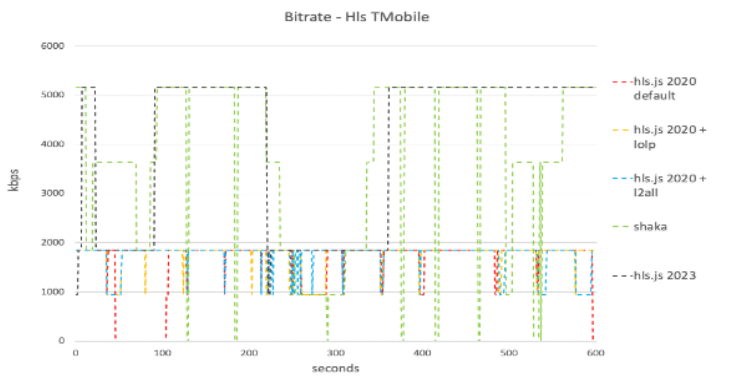
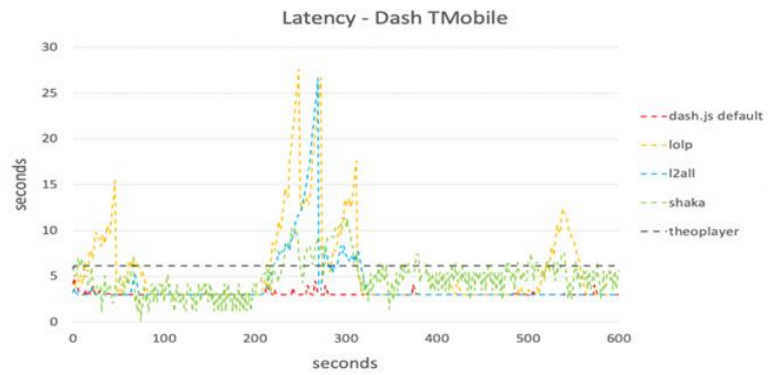
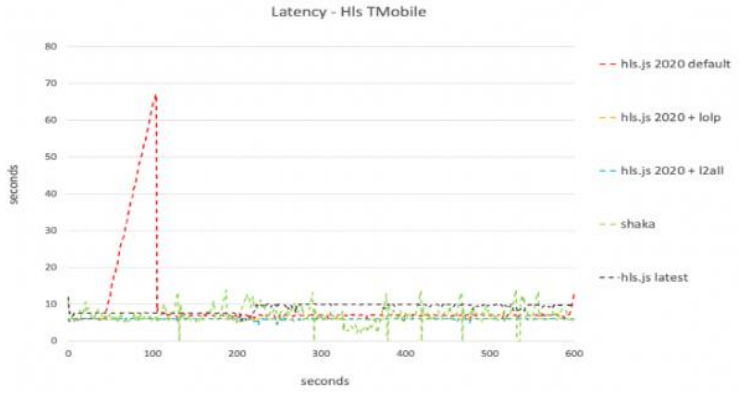
R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, H. Balakrishnan,
“Mahimahi: accurate record-and-replay for HTTP,” USENIX Annual Technical Conference, Santa Clara, CA, July 8-10, 2015

BRIGHTCOVE®

Test results

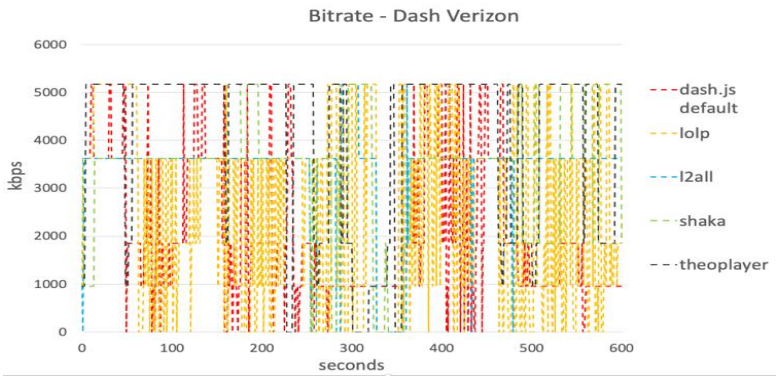
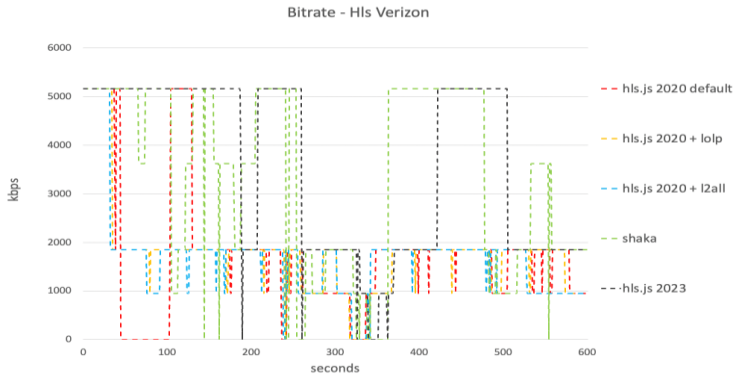
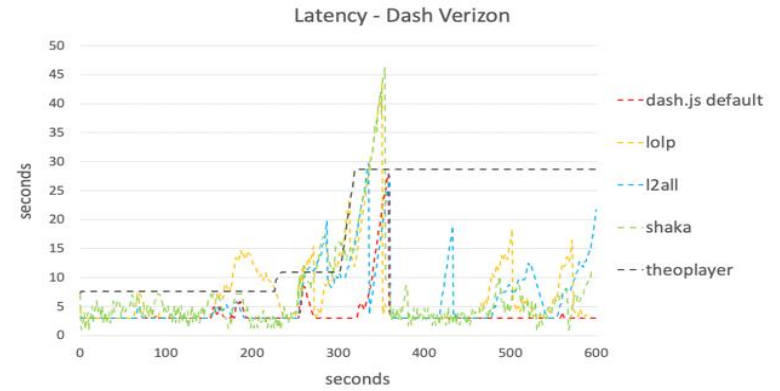
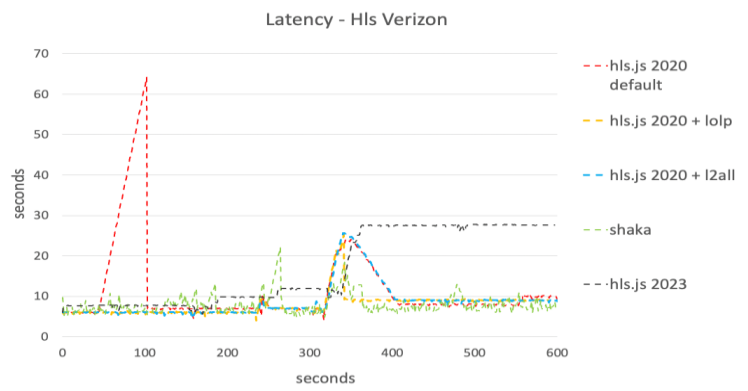
BRIGHTCOVE®

Results for T-Mobile LTE network

	LL-DASH	LL-HLS
Stream bitrate over time		
Latency over time		

BRIGHTCOVE®

Results for Verizon LTE network

	LL-DASH	LL-HLS
Stream bitrate over time		
Latency over time		

Statistics

BRIGHTCOVE

Table 5: Performance statistics – T-Mobile LTE network

Player/Algorithm	Avg. bitrate [kbps]	Avg. height [pixels]	Avg. latency [secs]	Latency var. [secs]	Speed var. [%]	Number of switches	Buffer events	Buffer ratio [%]	MBs loaded	Objects loaded
DASH.js default	2770	726	3.06	0.21	10.4	93	38	7.99	352.2	256
DASH.js LolP	3496	853	5.65	4.59	22.7	70	53	21.96	369.4	210
DASH.js L2all	3699	908	4.14	3.18	19.9	5	19	7.99	368	147
Shaka player (dash)	3818	916	4.92	2.06	0	16	5	4.66	360.3	155
THEO player (dash)	4594	993	6.16	0.01	0	27	0	0	418.7	152
HLS.js default 2020	1763	562	10.08	10.91	8.1	26	2	9.8	130.7	589
HLS.js LolP 2020	1756	560	5.97	0.2	6.1	24	0	0	148.1	688
HLS.js L2all 2020	1752	560	6	0.23	5.9	34	0	0	133.1	686
HLS.js default 2023	3971	895	8.93	1.13	0	8	0	0	360.8	613
Shaka player (hls)	3955	908	7.18	2.23	0	14	7	3.8	230	475

Table 6: Performance statistics – Verizon LTE network

Player/Algorithm	Avg. bitrate [kbps]	Avg. height [pixels]	Avg. latency [secs]	Latency var. [secs]	Speed var. [%]	Number of switches	Buffer events	Buffer ratio [%]	MBs loaded	Objects loaded
DASH.js default	2131	627	3.79	3.16	14.9	91	23	7.99	260	207
DASH.js LolP	3368	829	7.29	6.8	23.9	106	51	21.96	351	221
DASH.js L2all	3672	905	6.47	5.36	23.3	7	7	7.99	338	135
Shaka player (dash)	3653	886	6.96	7.08	0	26	5	4.66	329	148
THEO player (dash)	4153	909	18.19	10.08	0	33	2	0	383	153
HLS.js default 2020	2085	610	11.66	10.25	11.4	28	3	9.8	140	606
HLS.js LolP 2020	1890	580	7.86	2.63	7.0	24	2	0	146	569
HLS.js L2all 2020	1803	567	8.84	4.43	10.7	26	2	0	145	547
HLS.js default 2023	3541	822	16.78	9.13	0	9	5	0	280	598
Shaka player (hls)	3669	860	7.98	2.41	0	22	9	3.8	260	570

BRIGHTCOVE®

Observations

- Some players failed to stream with low-latency when network bandwidth is low and dynamic.
- Players frequently re-buffer in slower and dynamic network. Ensuring continuous and smooth playback is critical for general QoE.
- Some players double-download from two renditions in hope of avoiding buffer underrun. Our results show this strategy does not work well in practice but could lead to unnecessary complexity and bandwidth waste.
- Some players switch bitrate too often leading to reduced QoE.
- Some players change playback speed too often, while others stick to the natural speed.
- All the players shows better performance than our old study 2 years ago. HLS.js shows the greatest improvement.

BRIGHTCOVE®

Conclusions

BRIGHTCOVE®

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. But clearly, there is still considerable work ahead!

BRIGHTCOVE®

THANK YOU