MI

MOTION IMAGING JOURNAL

COVERING EMERGING TECHNOLOGIES FOR THE GLOBAL MEDIA COMMUNITY

SMPTE

Use of Common Media Client and Server Data to Improve Streaming Playback Performance

By Ali C. Begen, Yasser Syed, and Alex Giladi

Introdução:

O artigo desta edição trata da importância de melhorar a performance de reprodução de vídeos em streaming, especialmente com a crescente demanda por serviços de baixa latência e mais recentemente vídeos em 4K. O estudo destaca que os serviços competem por largura de banda, o que pode gerar rebufferings (interrupções de reprodução).

Para enfrentar esses problemas são usados dois padrões: Common Media Client Data (CMCD), que permite que os receptores informem ao servidor seu status de buffer, ajudando assim a distribuir melhor sua capacidade entre os clientes, o que reduz as interrupções. Já o CMSD (Common Media Server Data) propõe que o servidor adie as respostas a receptores com buffer suficiente, priorizando aqueles com buffers baixos. Os testes indicam que, com essas implementações, a duração média de rebuffering foi reduzida em até 56% sem comprometer a qualidade do vídeo. Como fazer isso? Borá para a leitura do artigo na integra. Boa leitura.

Tom Jones Moreira

Abstract

Smooth playback is critical for any streaming service. Naturally, viewers do not enjoy rebuffering events, but less frequent and shorter ones are preferred when they cannot be avoided. However, ensuring smoother playback becomes more challenging as more demanding types of services such as low-latency live (LLL) and 4K streaming become increasingly used. It has been shown many times that adaptive streaming clients compete with each other for available bandwidth and server capacity. The server's priority in responding to the individual requests sent by these clients should be based on the clients' time-varying playback buffer conditions. In an earlier article, we presented that the server could wisely allocate its output capacity among the incoming requests and largely mitigate the rebufferings suffered by the clients, provided that the clients informed the server about their buffer statuses using the Common Media Client Data (CMCD) standard. In a more recent study, we developed an alternative solution to the same problem using the Common Media Server Data (CMSD) standard. In this solution, the server's response to a request that indicated a sufficient buffer level was delayed until the requests that indicated an insufficient buffer level were handled. The server attached a new CMSD parameter to the eventual response disclosing the length of the delay. This parameter avoided misinterpretation and the subsequent incorrect decision by the client's rate-adaptation logic. Our experiments (for which we offer the source code) showed that the proposed CMSD parameter eliminated unnecessary rate shifting to lower bitrates while reducing the rebuffering rate as well as the rebuffering duration. This article summarizes these findings and illustrates other envisioned use cases for the CMSD standard.

Keywords

Adaptive streaming, adaptive bitrate (ABR), Common Media Client Data (CMCD), Common Media Server Data (CMSD), content delivery network (CDN), Dynamic Adaptive Streaming over HTTP (DASH), HTTP Live Streaming (HLS), server/network assistance.

It has been shown many times that adaptive streaming clients compete with each other for the available bandwidth and server capacity. The server's priority in responding to individual client requests should be based on clients' time-varying playback buffer conditions to avoid playback freezes happening in as few clients as possible.

Introduction

In the early days of Hypertext Transfer Protocol (HTTP) adaptive streaming, the content delivery network (CDN) providers did not envision HTTP servers communicating with streaming clients to enhance user experience or the overall system's delivery performance. The idea of cooperating servers, clients, and other network elements first took off in 2013[1] and became an MPEG standard[2] in 2017.

Although this standard had numerous useful applications, the industry had not shown enough interest in it since its publication. Then, in 2019, the WAVE project participants at the Consumer Technology Association (CTA) started working on a solution to a long-standing issue: how could a streaming client send media-, playback-, and session-related information to the CDN, such as type, duration, and format of a media segment, content ID, session ID, current buffer length, and current latency. The CDN could use this information to bind individual requests to streaming sessions and combine server and client logs to generate precise dashboard metrics showing delivery performance, specific player issues, and viewer experience. A similar issue that could be met through this method of passed information is how a streaming client could ask for a priority service when requesting a segment.

In September 2020, CTA published the Common Media Client Data (CMCD) specification.[3] Several

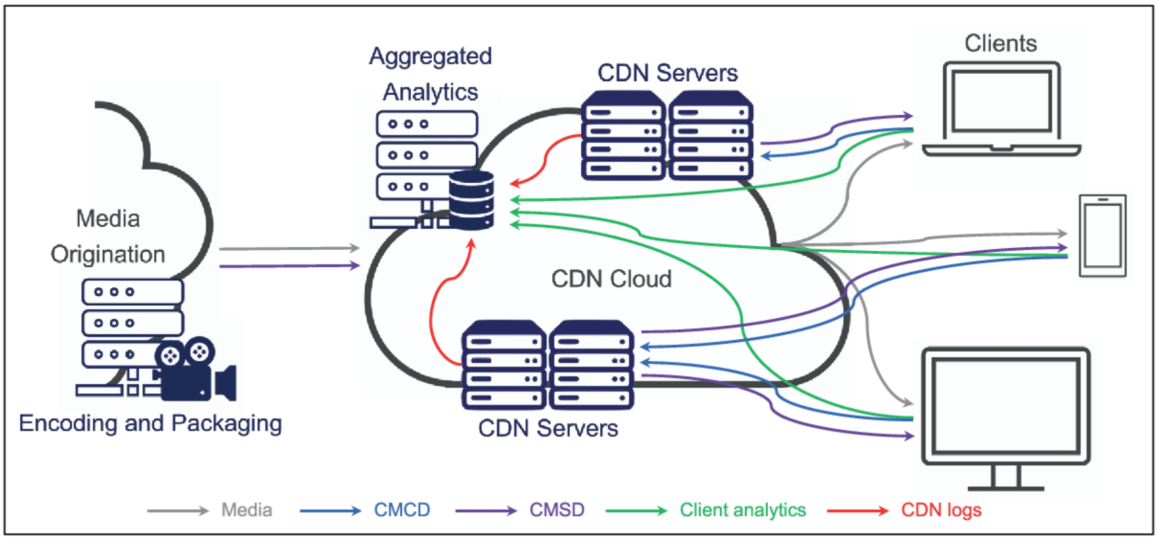


FIGURE 1. Streaming clients cooperating with the CDN servers through CMCD/CMSD (repurposed from Ref. 1).

early adopters from CDN providers and streaming client vendors immediately implemented this specification. For example, the dash.js reference client[4] has been fully compliant with CMCD since v3.2.1 (early 2021). Open-source libraries for other popular clients (e.g., hls.js and ExoPlayer) were made available. The first evaluation results and capability demonstrations (e.g., Refs. 5 and 6) came out soon after, and further studies are currently in progress. Readers interested in further details are referred to Ref. 1 for the timeline of the developments in this space and Ref. 7 for the detailed use cases of CMCD.

During the development of the CMCD specification, a proposal was made to send meta information and hints from the CDN servers to the streaming clients, similar to the capabilities provided in Ref. 2. The proposal led to the creation of a companion standard to CMCD, called *Common Media Server Data* (CMSD), which was finalized in late 2022.[8] **Figure 1** illustrates an end-to-end media distribution system with CDN servers and streaming clients enabled to use CMCD and CMSD.

It has been shown many times that adaptive streaming clients compete with each other for available bandwidth and server capacity.[9,10] To avoid playback freezes as much as possible, the server's priority in responding to individual client requests should be based on clients' time-varying playback buffer conditions.

Previously, we designed a buffer-aware bandwidth allocation algorithm for the server, which scheduled the response for each incoming segment request based on the (playback) buffer level reported using CMCD. Based on this algorithm, the server allocated its output capacity among all the requests more wisely, which then significantly reduced the rebufferings experienced by the clients.[5]

In a later study,[11] we developed an alternative solution to the same problem using the (draft) CMSD standard. In this solution, the response to a request that indicated a sufficient buffer level was to be delayed until the requests that indicated an insufficient buffer level were handled. The server attached a new CMSD parameter to the eventual response disclosing the length of the delay. This parameter avoided misinterpretation and the subsequent incorrect decision by the client's rate-adaptation logic. Our experiments (for which we offer the source code) showed that the proposed CMSD parameter eliminated unnecessary rate shifting to lower bitrates while reducing the rebuffering rate as well as the rebuffering duration. Specifically, we tested the proposed idea in different multi-client scenarios [a mix of video-on-demand (VoD) and low-latency live (LLL) streaming clients and 10–20 clients in total]. The results showed that the average rebuffering duration decreased by 33%–56% without degrading the video quality.

In this article, we summarize the main findings from these studies and illustrate other envisioned use cases for the CMSD standard.

Results Using the CMCD Standard

Test Setup

The test setup is given in **Figure 2**. On the client side, several instances of the dash.js client running the default ABR scheme (termed *dynamic*) were used. The desired network behavior was achieved using a tc-NetEm network emulator,[i] which throttled the total bandwidth available to the clients according to the Cascade and Spike bandwidth profiles defined by the DASH-IF. On the server side, an NGINX server[ii] with the JavaScript (NJS) module was used as the HTTP server. An NJS middleware application was developed to implement the bandwidth allocation functionality. The server hosted

[i]https://wiki.linuxfoundation.org/networking/netem

[ii]http://nginx.org/en/download.html

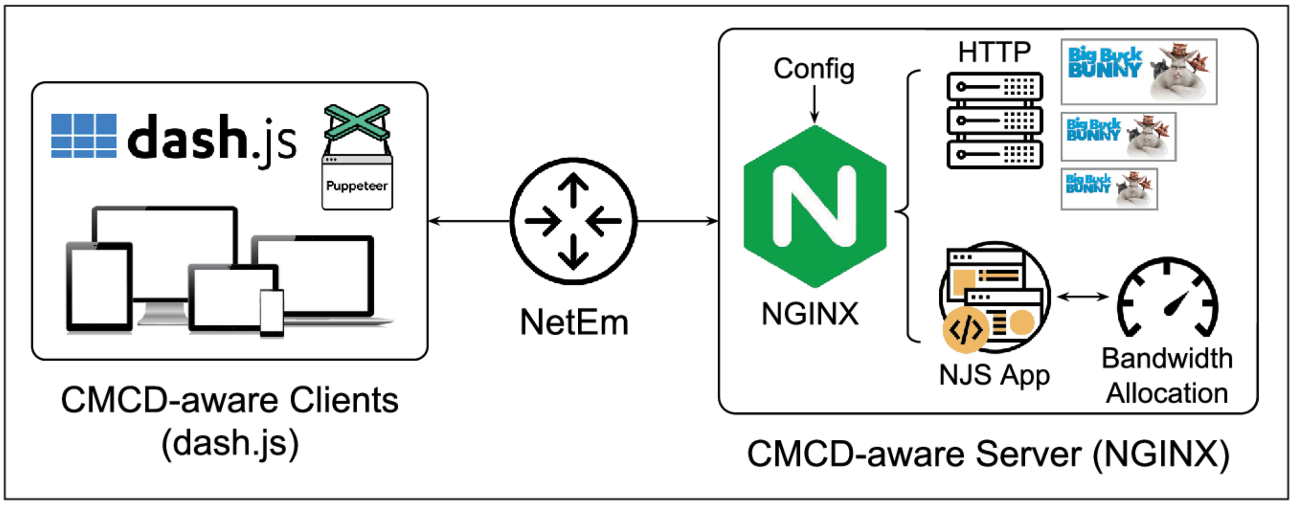


FIGURE 2. Implemented CMCD-aware system.[5,12]

the test content for testing on-demand and live video sessions. Further detailed information about the test setup can be found in Ref. 5, and the public source code is available in Ref. 12.

Results

The tests were repeated five times and the metrics shown in **Table 1** were used. We compared using CMCD with the buffer-aware bandwidth allocation algorithm against not using CMCD.

Table 1. Metrics used in the experiments.

Metric	Definition	Unit
Avg. BR	Average bitrate across all clients	Mbit/s
Min. BR	Average bitrate for the client that consumed the lowest average bitrate	Mbit/s
Avg. RD	Average total rebuffering duration across all clients	seconds
Max. RD	Total rebuffering duration for the client that suffered from the longest rebuffering duration	seconds
Avg. RC	Average rebuffering count across all clients	-
Avg. SC	Average bitrate switching count across all clients	-
Avg. LL	Average live latency across all clients	seconds
Max. LL	Average live latency for the client that experienced the longest live latency	seconds

On-Demand Video Sessions with 10 Clients

Table 2 shows that running the buffer-aware bandwidth allocation algorithm reduced Avg. RD, Max. RD, and Avg. RC substantially for both bandwidth profiles. Specifically, the reductions were as follows: Avg. RD by 74% and 83%, Max. RD by 72% and 78%, and Avg. RC by 57% and 66% for the Cascade and Spike bandwidth profiles, respectively. These reductions came at the cost of a nonnegligible (but much less significant) reduction in Avg. BR.

Table 2. On-demand video sessions with 10 clients.

	CascadeX10		SpikeX10	
Metric	w/ CMCD	w/o CMCD	w/ CMCD	w/o CMCD
Avg. BR	3.13	3.33	2.61	3.20
Min. BR	2.90	3.12	2.30	2.68
Avg. RD	5.36	20.84	12.43	71.90
Max. RD	10.72	38.84	18.49	83.54
Avg. RC	4.72	11.04	8.68	25.48
Avg. SC	35.80	36.70	49.14	53.90

The buffer-aware bandwidth allocation algorithm computes a fair share for each client (based on the buffer level reported through CMCD). The bandwidth allocated by the server implicitly controls the decisions taken by the client-side ABR scheme. Thanks to this algorithm, Avg. SC was also reduced. The results in **Table 2** are for the clients that set their minimum and maximum buffer levels to four and eight seconds, respectively. We observed that when we tripled these levels, the percentage of improvement was smaller since the larger playback buffer size provided more robustness against the bandwidth drops and reduced the chances of rebuffering.

LLL Video Sessions with Five Clients

The clients set their minimum and maximum buffer levels in these tests to one and three seconds, respectively. The target latency was also set to three seconds. The results are given in **Table 3**, where we see that CMCD-enabled clients achieved a reduction in Avg. RD of 20% and 26%, in Max. RD of 25% and 22%, and in Avg. RC of 36% and 21% for the Cascade and Spike bandwidth profiles, respectively.

Table 3. LLL video sessions with five clients.

	CascadeX5		SpikeX5	
Metric	w/ CMCD	w/o CMCD	w/ CMCD	w/o CMCD
Avg. BR	0.44	0.21	0.21	0.20
Min. BR	0.37	0.21	0.20	0.20
Avg. RD	3.56	4.45	3.62	4.91
Max. RD	3.87	5.16	4.27	5.44
Avg. RC	10.50	16.50	13.25	16.75
Avg. SC	21.75	18.00	5.75	3.25
Avg. LL	2.34	2.75	2.28	2.31
Max. LL	2.91	3.38	2.30	3.35

Moreover, enabling CMCD helped the clients achieve a live latency below the target (three seconds) without any violations. On the other hand, Max. LL surpassed the target value when CMCD was disabled. Interestingly, we also saw an increase in Avg. BR by 110% for the CMCD-enabled clients during the testing with the Cascade bandwidth profile. This was because the CMCD-enabled clients rebuffered less and this helped them request more segments from high-bitrate representations.

Results Using the CMSD Standard

Test Setup

The test setup is given in **Figure 3**. On the client side, several instances of the dash.js client running a throughput-based ABR scheme (abrThroughput.js) were used. The desired network behavior was achieved using a tc-NetEm network emulator, which throttled the total bandwidth available to the clients according to the scaled Cascade bandwidth profile. On the server side, an NGINX server with the JavaScript (NJS) module ran the HTTP server and an NJS middleware application implementing the scheduler for the outgoing responses. The server hosted the test content for testing on-demand and live video sessions. Further detailed information about the test setup can be found in Ref. 11, and the public source code is available in Ref. 13.

Results

The following test cases were run five times and the results were averaged. We compared CMSD with scheduling against no CMSD or scheduling using the same metrics from the previous section.

The two test cases were as follows:

(i) 10 VoD streaming clients under the CascadeX10 bandwidth profile and

(ii) 10 VoD streaming and 10 LLL streaming clients under the CascadeX20 bandwidth profile,

where the minimum and maximum buffer levels were set to four and 20 seconds, respectively, for the VoD clients and to two and six seconds, respectively, for the LLL clients.

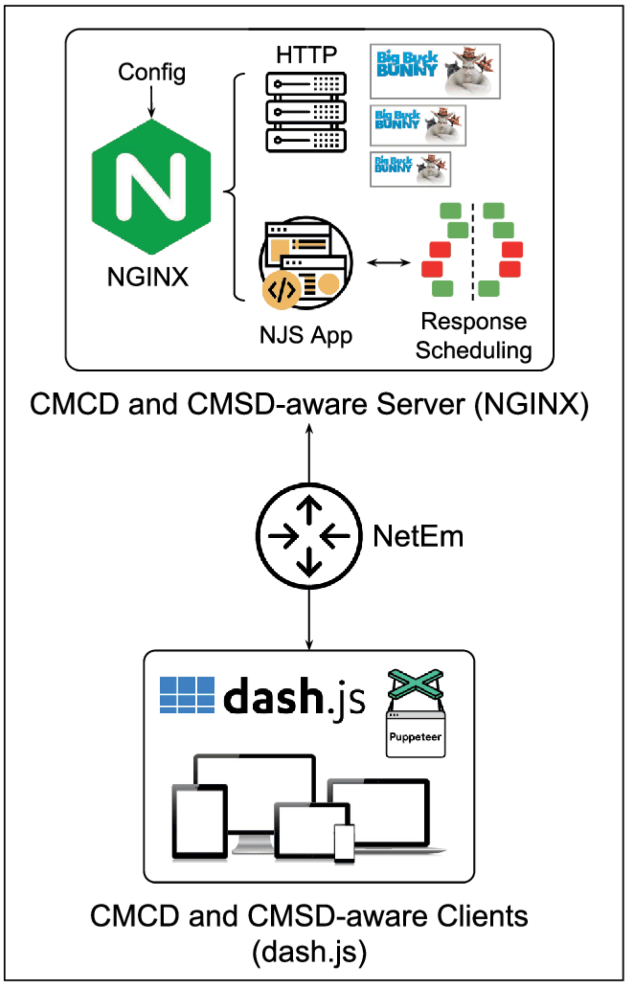

Config
HTTP
NGINX
NJS App
Response Scheduling
CMCD and CMSD-aware Server (NGINX)
NetEm
dash.js
Puppeteer
CMCD and CMSD-aware Clients (dash.js)

FIGURE 3. Implemented CMSD-aware system.[11,13]

Table 4. Results for the test cases: (i) 10 VoD streaming clients (CascadeX10) and (ii) 10 VoD and 10 LLL streaming clients (CascadeX20).

	CascadeX10		CascadeX20	
Metric	w/ CMSD	w/o CMSD	w/ CMSD	w/o CMSD
Avg. BR	3.46	3.55	3.20	3.26
Min. BR	3.15	3.27	2.45	2.59
Avg. RD	3.52	5.26	0.51	1.16
Max. RD	15.0	14.5	4.15	8.21
Avg. RC	1.52	2.18	0.40	0.55

The results shown in **Table 4** indicate that if the server ran the response scheduling algorithm and signaled any extra delay introduced to the client, it could reduce Avg. RD by 33% and 56% and Avg. RC by 30% and 27% for test cases (i) and (ii), respectively. At the same time, the reduction in Avg. BR was limited to a mere 2%. Overall, the response scheduling algorithm kept the average bitrate (quality) more or less unchanged even as it added a delay to segment downloads since the delay was signaled to the client using CMSD, and this delay was accounted for in the throughput calculations on the client side.

Further Use Cases for the CMSD Standard

CMSD can help in many ways. For example:

- Clients may frequently upshift and downshift.[14] This instability can be avoided using CMSD hints.
- CMSD can let clients know about server-side bandwidth measurements.
- Clients typically start fetching the lowest-bitrate segments due to the lack of knowledge of the network conditions. An edge server may know the available bandwidth a client has and can signal this value to the client.
- Caching capacity is always a limited resource. Thus, not all segments for all the representations can be cached on every server. Adaptive streaming clients can get confused if some of the objects requested are served from a nearby cache and others from a distant server.[15] CMSD can also help in this case by sending caching indications to let clients know what is cached or not. This way, clients can make more informed decisions.
- Clients might erroneously downshift due to the increase in download times because of a cache miss. Such clients could be informed via CMSD and they might decide not to downshift.
- In LLL streaming, CMSD can hint clients about the latest (live-edge) segment.
- CMSD can assist clients in viewing the content in a synchronized fashion.

Other use cases are also exemplified in Ref. 11.

Conclusion

The goal of the published CMCD and CMSD specifications is to improve cooperation between adaptive streaming clients and CDN servers to enhance streaming performance. Validating the use cases is vital to promote wider adoption. In this work, we summarized the main findings from two recent studies. The results showed that even a basic implementation benefiting from CMCD and CMSD could reduce the average duration and count of rebuffering without discernible loss of video quality.

References

1. A. C. Begen, "Manus Manum Lavat: Media Clients and Servers Cooperating With Common Media Client/Server Data," *ACM Appl. Netw. Res. Workshop (ANRW)*, 2021, doi: 10.1145/3472305.3472886.

2. International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) 23009-5:2017, "Information Technology—Dynamic Adaptive Streaming Over HTTP (DASH)—Part 5: Server and Network Assisted DASH (SAND)." Accessed: Sept. 10, 2022. [Online]. Available: https://www.iso.org/standard/69079.html

3. Consumer Technology Association, "CTA-5004: Web Application Video Ecosystem–Common Media Client Data," Sept. 2020.

4. DASH-IF, "DASH Reference Client." Accessed: Sept. 10, 2022. [Online]. Available: https://reference.dashif.org/dash.js/

5. A. Bentaleb, M. Lim, M. N. Akcay, A. C. Begen, and R. Zimmermann, "Common Media Client Data (CMCD): Initial Findings," *ACM NOSSDAV, 2021*, doi: 10.1145/3458306.3461444.

6. S. Pham, M. Avelino, D. Silhavy, T.-S. An, and S. Arbanowski, "Standards-Based Streaming Analytics and Its Visualization," *ACM MMSys*, 2021.

7. A. C. Begen, A. Bentaleb, D. Silhavy, S. Pham, R. Zimmermann, and W. Law, "Road to Salvation: Streaming Clients and Content Delivery Networks Working Together," *IEEE Commun. Mag.*, 59(11):123–128, 2021, doi: 10.1109/MCOM.121.2100137.

8. Consumer Technology Association, "CTA-5006: Web Application Video Ecosystem–Common Media Server Data," Nov. 2022.

9. S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What Happens When HTTP Adaptive Streaming Players Compete for Bandwidth?," *ACM NOSSDAV*, 2012, doi: 10.1145/2229087.2229092.

10. A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, "A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP," *IEEE Commun. Surveys Tuts.*, 21(1):562–585, Firstquarter 2019, doi: 10.1109/COMST.2018.2862938.

11. M. Lim, M. N. Akcay, A. Bentaleb, A. C. Begen, and R. Zimmermann, "The Benefits of Server Hinting When DASHing or HLSing," *ACM MHV*, 2022, doi: 10.1145/3510450.3517317.

12. NUS-OzU, "CMCD-DASH." Accessed: Sept. 10, 2022. [Online]. Available: https://github.com/NUStreaming/CMCD-DASH

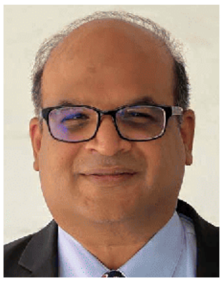
13. NUS-OzU, "CMSD-DASH." Accessed: Sept. 10, 2022. [Online]. Available: https://github.com/NUStreaming/CMSD-DASH

14. S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "Server-Based Traffic Shaping for Stabilizing Oscillating Adaptive Streaming Players," *ACM NOSSDAV*, 2013, doi: 10.1145/2460782.2460786.

15. D. H. Lee, C. Dovrolis, and A. C. Begen, "Caching in HTTP Adaptive Streaming: Friend or Foe?," *ACM NOSSDAV*, 2014, doi: 10.1145/2597176.2578270.

About the Authors

Ali C. Begen is currently a computer science professor at Ozyegin University, Istanbul, Turkiye, and a technical consultant with Comcast's Advanced Technology and Standards Group. Previously, he was a research and development engineer at Cisco. He received a PhD in electrical and computer engineering from Georgia Tech in 2006. To date, he has received several academic and industry awards (including an Emmy Award for Technology and Engineering) and was granted over 30 US patents. In 2020 and 2021, he was listed among the world's most influential scientists in the subfield of networking and telecommunications. More details are at https://ali.begen.net.



Yasser Syed is a distinguished architect at Comcast, whose work focuses on developing and implementing next-generation video technologies and workflows. He received a PhD in electrical engineering from the University of Texas at Arlington. He has contributed a substantial amount of content and effort to many video standards-related areas. Currently, he chairs the Society of Cable Telecommunications Engineers (SCTE) Working Group 7 (WG7) on DASH/adaptive streaming and also leads its Advanced Coding Technologies group, which created the high-dynamic-range (HDR) application and next-generation codec constraint specifications for the cable industry. He is also a 2021 recipient of the SCTE award for excellence in standards.

Alex Giladi is a Comcast fellow and an Emmy-winning technologist. He has been working on issues related to video content encoding and distribution since 2000. His current areas of interest are video encoding and adaptive streaming. He currently leads an advanced technologies group within Comcast. Prior to Comcast, he worked on various aspects video transport and video coding in InterDigital, Huawei, Vubiquity, Digital Fountain, and Harmonic. He has been a prolific contributor to the MPEG-DASH standard and served as an editor of several standards and amendments related to DASH, content security, and MPEG-2 Systems. He is the vice president of the DASH Industry Forum since 2019. He is the founder and co-organizer of the popular Mile-High Video annual conference series held in Denver, CO, since 2017. He holds an MSEE degree from Stanford University, Stanford, CA, and a BSc degree from Technion, Haifa, Israel. He is a senior member of the IEEE and holds more than 50 U.S. patents.

SMPTE

PARTICIPATE WITH SMPTE FOR A CHANGE

Keeps our members current and connected in industry, standards, education, engineering, and global media technology.