Optimizing Mass-Scale Multiscreen Video Delivery

By Yuriy Reznik, Xiangbo Li, Karl Lillevold, Robert Peck, Thom Shutt, and Peter Howard

Abstract

We discuss the design of a mass-scale streaming video delivery system, targeting devices with different decoding, playback, and network capabilities. This system incorporates several advanced tools: dynamic packaging, dynamic devices detection, dynamic manifest generation, rules engine, analytics engine, and context-aware encoding (CAE). We explain how this system is implemented using the current cloud computing and content delivery network platforms and how it is tuned to achieve optimal end-to-end performance, delivering the best possible user experience with minimized transcoding, bandwidth, and storage costs.

Keywords

Adaptive bitrate (ABR) streaming, cloud-based video processing, context-aware encoding (CAE), end-to-end optimizations, over-the-top (OTT)

Introduction

Over the past two decades, over-the-top (OTT) streaming, used for the delivery of media content, has evolved from a pioneering concept to a mainstream technology. Important steps in this evolution included:

- The invention of adaptive bitrate (ABR) streaming.[1,2]
- The emergence of content delivery networks (CDNs) and HyperText Transfer Protocol (HTTP)-based streaming models.
- The emergence of suitable standards for codecs, file formats, and delivery systems [e.g., H.264/Advanced Video Coding (AVC),[3] High-Efficiency Video Coding (HEVC),[4] HTTP live streaming (HLS),[5] Dynamic Adaptive Streaming over HTTP (DASH),[6] and common media application format (CMAF)[7]].
- Consolidation of Digital Rights Management systems (DRMs) to fewer recognized systems (e.g., FairPlay,[8] PlayReady,[9] and Widevine[10]) that are broadly supported.
- Improvements in client technologies such as Media Source Extensions[11] and Encrypted Media Extensions,[12] making it possible to use web browsers as streaming clients.

Despite many improvements, standards, and consolidation, modern-day OTT media delivery systems are still facing fragmentation when it comes to streaming client devices and their support for codecs, formats, and DRMs.

Despite all these improvements, standards, and consolidation, modern-day OTT media delivery systems are still facing fragmentation when it comes to streaming client devices and their support for codecs, formats, and DRMs.

For example, it is well known that most existing devices can decode H.264/AVC streams, including streams encoded using H.264 Main and High profiles. Some newer devices, most notably Apple devices with iOS 11 or later, can also decode HEVC.[4] However, many older devices, including Android devices with versions prior to 6.0, most likely can only play streams encoded using the H.264 Baseline profile.[3]

Likewise, it is not a secret that one has to use HLS streaming to reach Apple devices, whereas the Motion Picture Experts Group (MPEG)-DASH is preferred for Androids and Smart TVs. Moreover, one still needs to use Smooth streaming[13] to reach some older TVs and game consoles, and has to resort to using progressive downloads to reach legacy mobiles (Blackberries, feature phones, etc.). The support for different types of DRMs across different devices is also fragmented, as illustrated in **Fig. 1**.

Known methods of handling fragmented support for different codecs and formats include:[14]

- The creation of separate copies of streams, packaged specifically to different delivery formats (HLS, DASH, Smooth, etc.) and DRMs (PlayReady, FairPlay, Widevine, etc.)
- Dynamic transmuxing and dynamic encryption of streams, encoded and stored in some intermediate formats to match the requirement of final delivery formats and DRMs

Digital Object Identifier 10.5594/JMI.2020.2973561
Date of publication: xx xxx xxxx

	Media players	PlayReady	Widevine Modular	Widevine Classic	FairPlay
Browsers	Chrome (35+)	✗	✓	✗	✗
	Firefox (47+) ON WINDOWS VISTA+, MAC OS X 10.9+, LINUX	✗	✓	✗	✗
	Internet Explorer (11) ON WINDOWS 8.1+	✓	✗	✗	✗
	Microsoft Edge	✓	✗	✗	✗
	Opera (31+)	✗	✓	✗	✗
	Safari SAFARI 8+ ON MACOS & SAFARI ONIOS 11.2+	✗	✗	✗	✓
Mobile	Android (6+)	✗	✓	✗	✗
	Android (4.4 – 5.1)	✗	✓	✓	✗
	Android (3.1 – 4.3)	✗	✗	✓	✗
	iOS (6+)	✗	✗	✗	✓
	Windows Phone	✓	✗	✗	✗
Set-top-boxes	Chromecast	✓	✓	✗	✗
	Android TV	✓	✓	✗	✗
	Roku	✓	✓	✗	✗
	Apple TV	✗	✗	✗	✓
	Amazon Fire TV	✓	✗	✗	✗
	Google TV	✓	✗	✓	✗
Smart TVs	Samsung (Tizen) 2017-2018+ MODELS	✓	✓	✗	✗
	Samsung (Tizen) 2015-2017 MODELS	✓	✗	✓	✗
	Samsung (Orsay) 2010-2015 MODELS	✓	✗	✓	✗
	LG (webOS & Netcast)	✓	✗	✓	✗
	Smart TV Alliance LG, PHILIPS, TOSHIBA, PANASONIC	✓	✗	✓	✗
	Android TV	✓	✓	✗	✗
GCs	Xbox One / 360	✓	✗	✗	✗
	PlayStation 3 / 4	✓	✗	✗	✗

FIGURE 1. Support of DRMs across different devices.

- The creation of separate encoding profiles (and ABR stacks of streams) using each codec (e.g., H.264, HEVC, and H.264 baseline).
- The creation of mixed encoding profiles, where low-bitrate streams, targeted to legacy devices, are encoded using the H.264 baseline profile, whereas higher bitrate streams are encoded using the H.264 Main and High profiles.

Most of these methods, however, have certain limitations or disadvantages, causing the delivery system to be more expensive to operate and/or be suboptimal in terms of quality. For example, the creation of many versions of encoded streams for different codecs, protocols, and DRMs dramatically increases transcoding, storage, and CDN costs. It also affects the efficiency of the CDNs, as their edge cache space is limited, and the

use of multiple copies of the same content will inevitably increase the cache miss probability. The use of dynamic (or just-in-time) transmuxing and encryption could, in principle, minimize storage, transcoding, and CDN costs. However, they require the implementation of a whole set of additional system elements. This includes dynamic device detection, dynamic generation of manifests, and dynamic delivery of final streams over CDNs, and the means for doing all these operations at the scale and sufficiently close to the edge to achieve practically acceptable delays.

Similarly, the use of separate encoding profiles for different codecs or codec profiles/level combinations seems wasteful. It may create more streams than are really needed for efficient delivery. The use of mixed encoding profiles sounds like a better idea, but its use is complicated by the fact that some players may not be able to switch between different codecs (e.g., H.264/AVC and HEVC streams).

In addition, depending on the operator or the region, the population of streaming client devices and their playback capabilities may be very different. For example, there could be a different split between PCs and mobiles, Apple versus Android platforms, average age of the devices, their connection speeds, and so on. Hence, ideally, encoding profiles and delivery optimizations should be customized, accounting for the context of each operator or region.

Finally, what also makes video encoding and delivery challenging is that the video content, by itself, is highly variable. For example, high-action/sports content may require more bits to reach the acceptable quality level than videos with little or occasional motion (e.g., interviews, documentaries, and soap operas). This makes the use of "static" (preconfigured for all content) ABR encoding profiles suboptimal. To address this issue, in recent years, several techniques have been proposed based on the concept of dynamic generation of encoding profiles for each content item. These include the so-called *per-title*,[15] *content-aware*,[16] and *context-aware*[17–19] encoding techniques. However, in most cases, such techniques have been developed to work only with a single codec (e.g., H.264 or HEVC). See the recent article[19] for an exception to this. The use of multiple codecs and specifics of fragmentation of their support have not yet been incorporated in the problem definition addressed by these techniques.

Summarizing all of the above, we note that although many effective techniques, standards, and guidelines for building mass-scale multiscreen delivery systems have already been developed, there are still many areas where additional improvements could be made. Such areas include improvements in the overall delivery system architecture design, coupling and joint optimization of different modules (e.g., device detection, transmuxing, dynamic manifest generation, and encoding profile generation), and end-to-end system performance optimizations. The objective of this article is to offer some results on the above-mentioned topics.

This article is organized as follows. In the following section, we will describe a multiscreen delivery system architecture, highlighting commonly known and some unique elements, and the reasoning behind them. We then focus on explaining tools developed for end-to-end optimizations. We next present examples of system statistics and explain performance gains that have been achieved. The last section offers concluding remarks.

Architecture of Cloud-Based Multiscreen Video Delivery System

System Overview

In **Fig. 2**, we present the high-level architecture of the proposed cloud-based video delivery system. It consists

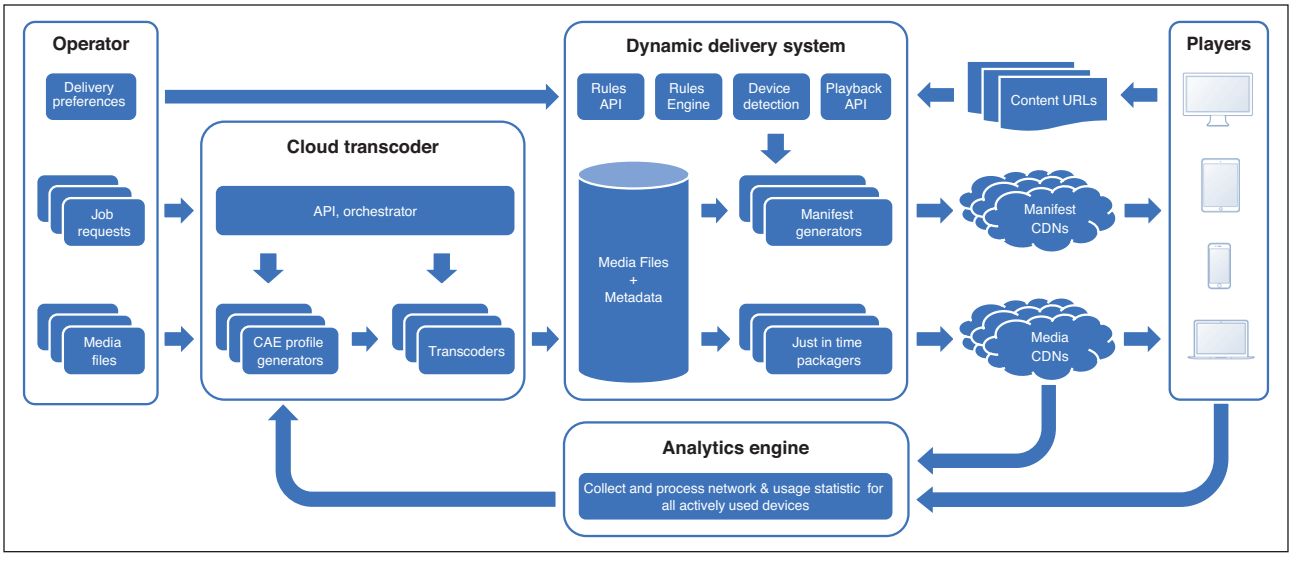

FIGURE 2. High-level architecture of cloud based video delivery system.

of several functional blocks and where all exchanges, as common in cloud-based systems, are done by means of RESTful Application Programming Interfaces (APIs).

For example, an operator can use an API to instruct the system to ingest the content, transcode it, and then deliver it using a CDN or several CDNs, based on their choice.

The transcoding of the content is done in two steps. The first step is responsible for the generation of ABR encoding profiles, followed by the traditional transcoding process and producing a set of transcoded streams (or renditions). The resulting streams, along with additional metadata, are then placed on the storage used by the dynamic delivery system. We note that such steams are not yet encrypted or packaged into the final delivery formats (e.g., HLS or DASH segments). Instead, they are stored in an intermediate format, thereby allowing fast transmuxing operations.

The dynamic delivery system is essentially a layer performing selective transmuxing, encryption, and transfer of data to CDNs used for caching and delivery. It is also responsible for manifest generation. This module is implemented as a highly distributed system that allows such operations to be performed sufficiently close to the players.

The analytics engine is a system that collects information from players as well as CDNs for the purpose of system performance analysis and end-to-end system optimizations.

In the following subsections, we describe the operations of the elements of this system during the video delivery process.

Playback Initiation

When the dynamic delivery system receives a playback request for a particular media content, it generates a list of manifest URLs, representing all possible combinations of supported delivery protocols, formats, and DRMs. This list of URLs is subsequently presented to a player. If the player recognizes any of the formats in this list, it then tries to load the corresponding manifest based on the URL provided. This brings control to the manifest CDN, which is used to implement the *device detection* and *manifest generation* functions.

Device Detection

The objective of device detection is to identify the type, capabilities, and location of a playback device. For such purposes, the device detector uses user–agent, and the other fields in HTTP headers present at the time manifest is requested. The list of properties that device detection is trying to establish is shown in **Table 1**.

Manifest Generation and Rules Engine

The primary function of dynamic manifest generation is to match the features and streams specified in the manifest to the capabilities of the receiving device. For example, for a device that can only decode H.264 baseline, only such renditions will be retained. On the other hand, if the device can support HEVC, H.264, and DASH, and can also switch across adaptation sets and codecs, then the output can be an Media Presentation Description (MPD) with both the H.264 and HEVC adaptation sets and supplemental properties declaring the adaptation sets as switchable.

TABLE 1. Properties that device detection seeks to establish.

Property	Possible values
Device type	PC, smartphone, tablet, TV, etc.
OS type/version	Android 6.0, iOS 11, etc.
Browser type/version	Chrome 51, Mozilla 5.0, etc.
Geographic region of device	Country code
Video codec support	H.264 baseline, H.264, HEVC, etc.
Supports codec switching	Yes/No
Maximum supported resolution	1080p, 540p, 480p, etc.
Maximum supported bitrate	1.2 Mbits/s, 4 Mbits/s, 10 Mbits/s, etc.
Formats and DRM support	HLS v4, PlayReady, etc.
HDR video support	Yes/No

The other function of the manifest generator is to apply certain additional rules defined by the operators. For example, based on the geo location and certain other parameters, an operator may decide to use a different CDN or limit the maximum resolution, bitrate, etc. The corresponding blocks providing for such functionality in **Fig. 2** are the rules API and rules engine.

Just-in-Time Packaging

When the manifest is finally received by the player, it starts retrieving media segments from the CDN. Such media segments again may or may not be present in the CDN cache. In case of cache misses, the CDN response brings the control back to the dynamic delivery system and its just-in-time packager. In turn, the just-in-time packager retrieves the corresponding segments of the content, transmuxes them to the required format [e.g., MPEG-2 transport stream (TS)[20] or ISO Base Media File Format (ISOBMFF)],[21] and passes them back to CDN for delivery.

In other words, the segments in all permutations of formats and DRMs are never generated or stored in a permanent way on the cloud storage. Instead, this system stores only single copies of the content in an intermediate format. This significantly reduces the cloud storage, bandwidth, and operation costs.

Cascaded CDN Architecture

To reduce the frequency at which just-in-time packagers are invoked, the delivery system uses the second-layer CDN architecture. The first-layer CDN, which interfaces with the packagers, is used for initial caching

and propagation of the content to different regions. The second-layer CDNs are then added in each region according to the operator's preferences and provide edge caching as needed for delivery to end users.

Based on the above description, the proposed architecture is designed specifically to minimize transcoding, transmuxing, storage, and CDN delivery costs while supporting the plurality of existing codecs, formats, and DRMs as needed for multiscreen delivery.

Measuring and Tuning Overall System Performance

Bandwidth and Usage Statistics

The collection of bandwidth, usage, and other relevant statistics in the system, shown in **Fig. 2**, is done by the analytics engine. It collects information from two sources: players and CDNs. Players' data are needed to understand which content segments have been played as well as to measure buffering and start-up latencies. CDN statistics show which segments have been delivered to the device and at which speed. By pooling such data, the analytics engine is able to collect variety of statistics, including information about usage and bandwidth distributions related to different categories of client devices.

Examples of bandwidth distributions, as measured for three different OTT operators, are shown in **Figs. 3–5**. The associated device usage and average bandwidth statistics are shown in **Tables 2–4**.

The information provided in the aforementioned tables and figures suggests that bandwidth and usage statistics in these three cases are very different. Operator 1 streams predominantly to mobiles, and its effective average bandwidth across all devices is only 3.3 Mbits/s. Operator 2 has a mixed distribution to PCs, mobiles, tablets, and TV screens. Its effective average bandwidth across all devices is about 16.393 Mbits/s. Operator 3 streams only to TVs, and the effective average bandwidth in this case is much higher—around 35.77 Mbits/s.

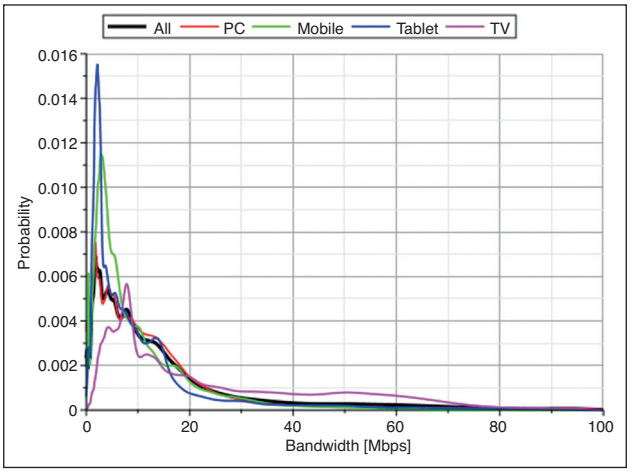

FIGURE 4. Bandwidth histograms measured for operator 2.

Playback Statistics

The analytics engine module also collects and reports a variety of statistics related to the Quality of Experience (QOE) delivered by the system. Examples of such statistics collected from the aforementioned operators are presented in **Tables 5–7**. In all cases, the statistics were collected for the streaming of the same test content, encoded using nine renditions as prescribed by the standard HLS ladder for the H.264 codec.[22] This ladder is shown in **Table 8**.

The first nine rows in **Tables 5–7** list the estimated load probabilities of each rendition. This is followed by the QOE parameters, such as buffering probability, start-up latency (in seconds), average bandwidth (in Kbits/s), average delivered resolution (in frame heights), and the average encoding quality [expressed in Structural Similarity Index Metric (SSIM)[23]], as delivered by a combination of streams pulled by streaming clients. The reported SSIM values were initially computed during the encoding stage, and then retrieved and aggregated into the final average value based on the rendition

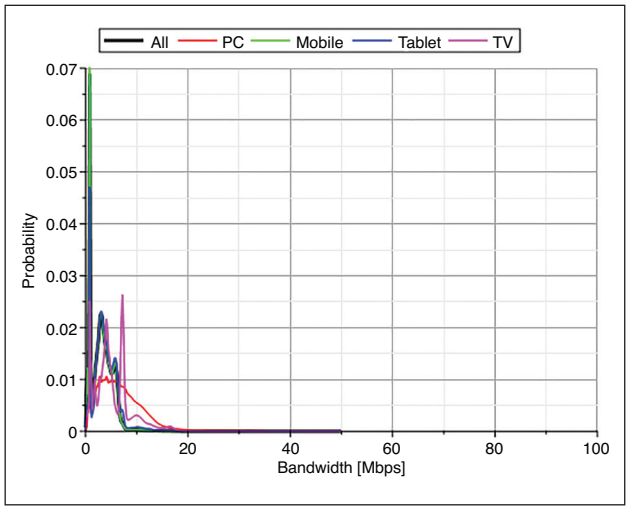

FIGURE 3. Bandwidth histograms measured for operator 1.

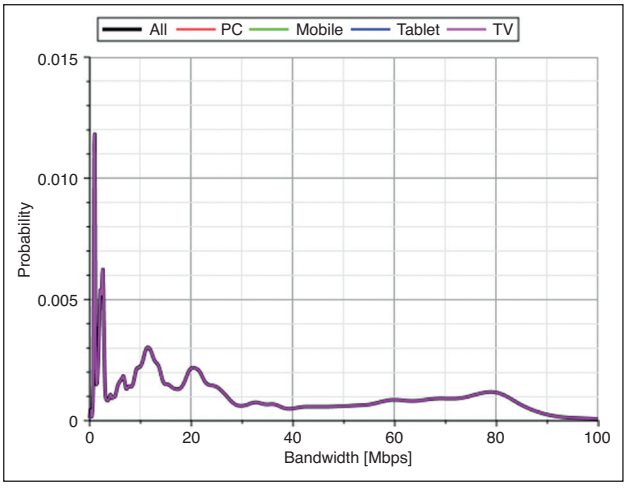

FIGURE 5. Bandwidth histograms measured for operator 3. This operator streams only to connected TVs.

TABLE 2. Usage and average bandwidth statistics for operator 1.

Device type	Usage (%)	Average bandwidth (Mbits/s)
PC	0.004	7.5654
Mobile	94.321	3.2916
Tablet	5.514	3.8922
TV	0.161	5.4374
All devices	**100**	**3.3283**

TABLE 4. Usage and average bandwidth statistics for operator 3.

Device type	Usage (%)	Average bandwidth (Mbits/s)
PC	0.0	N/A
Mobile	0.0	N/A
Tablet	0.0	N/A
TV	100	35.7736
All devices	100	35.7736

load statistics. Such statistics are provided separately for each category of devices, as well as in a combined form, averaged across all devices (last column).

As easily noticed, despite the fact that content in all delivery scenarios is encoded identically, the QOE delivered by these three operators is very different. Operator 3 pulls mostly top bitrate 1080p rendition (with a probability of about 0.83), delivering about 1003 lines of resolution on average and encoding quality of about 0.971 SSIM. In the case of operator 2, the probability of loading of top-most rendition drops to about 0.62 considering all devices, and just about 0.43 for mobiles. This results in lower average resolutions (about 938 lines across all devices and only 867 lines for mobiles), as well as lower average encoding quality, as the ladder shown in **Table 8** progressively drops the encoding quality for lower resolutions. Finally, in the case of operator 1, the situation is even worse. Renditions 3 and 5–7 become the most commonly used, resulting in an average delivered resolution of about 631 lines (628 for mobiles) and an encoding quality of about 0.966 SSIM.

Means for Tuning the System

The system depicted in **Fig. 2** includes several tools and means by which it can be adjusted or tuned to achieve the best performance, given each operator's context and needs.

The analytics engine, as described above, provides a relevant set of performance statistics. Such statistics can be localized to regions, jobs, content, delivery devices, CDNs, etc. They help the operators to monitor the health and efficiency of the system.

The use of *rules API* and *rules engine* allows operators to select local CDNs and distribute traffic between them

TABLE 3. Usage and average bandwidth statistics for operator 2.

Device type	Usage (%)	Average bandwidth (Mbits/s)
PC	63.49	14.720
Mobile	6.186	10.609
Tablet	9.165	12.055
TV	21.15	24.986
All devices	100	16.393

dynamically without disrupting operations. It also allows operators to impose limits and effectively add or remove certain streams that can be delivered. The addition of streams, especially the low-bitrate ones, may be considered as a means for reducing the buffering probability or load times. On the other hand, the removal of certain streams may be considered for reducing the bandwidth usage or for improving the CDN cache performance.

Finally, the system shown in **Fig. 2** also allows encoding profile generation for each new content item to be generated dynamically. Such generation accounts for the characteristics of the content, as well as bandwidth-, usage-, and playback-statistics collected for each operator. Such profile generation and encoding process is called *Context-Aware Encoding* (CAE). This step effectively closes the feedback loop provided by the analytics engine and allows better encoding of new content, accounting for the current context (delivery and playback statistics) of each operator.

Context-Aware Profile Generation

When the CAE profile generator is activated, it analyzes the content first, trying to model the space of quality-rate operating points achievable for a given codec and the content. This is followed by an optimization process, which selects a set of rates, resolutions, and other parameters for ABR encoding profiles, trying to achieve a sufficient level of quality while minimizing bandwidth, storage, compute, and other resources required for delivery.

Importantly, in such an optimization process, the quality estimates for each possible resolution and bitrate come from the content analysis, and the estimates of stream load probabilities at each rate come from the network bandwidth statistics provided by the analytics engine. Such network bandwidth statistics are collected separately for each type of client device. In computing the final optimization cost expression, the CAE generator aggregates estimates obtained for each device type according to the device usage distribution—also provided by the analytics module.

In other words, CAE profile generation performs functions of an end-to-end optimization process for multidevice/multiscreen delivery. The formal mathematical description of the underlying optimization problem is given by Reznik et al.[17] Chen et al.[18] extend it to a case of

TABLE 5. Playback statistics for operator 1.

Statistics	PC	Mobile	Tablet	TV	All
Rendition 1	0.00331	0.02046	0.01024	0.00678	0.01987
Rendition 2	0.01732	0.05157	0.03159	0.0207	0.05042
Rendition 3	0.01738	0.1402	0.09481	0.06734	0.13757
Rendition 4	0.05788	0.06888	0.05975	0.0676	0.06837
Rendition 5	0.09267	0.18057	0.18157	0.07306	0.18045
Rendition 6	0.14752	0.26691	0.28177	0.24079	0.26769
Rendition 7	0.14315	0.18247	0.19578	0.14113	0.18313
Rendition 8	0.15852	0.06816	0.09574	0.21973	0.06993
Rendition 9	0.36199	0.0161	0.04503	0.16131	0.01794
Buffering	0.00026	0.00468	0.00372	0.00156	**0.00463**
Start time	2.14374	4.02900	3.46468	2.60737	**3.98948**
Bandwidth	5131.21	2730.21	3174.90	4218.81	**2757.25**
Resolution	861.221	628.680	676.972	794.530	**631.624**
SSIM	0.97030	0.96666	0.96836	0.96879	**0.96676**

designing profiles using multiple codecs and fragmentation of their support across different devices.

In passing, we must note that the complexity/cost overhead that CAE adds to the overall encoding process is rather minor. The most complex step is the content analysis step which, complexity-wise, is roughly equivalent to a single-pass encode of the content at a single bitrate and resolution. Such an analysis needs to be done once for each content item. Considering that next step is the generation of multiple (typically 3–9) renditions of the same content and that such final renditions are usually produced by using a much slower two-pass encoding process, the time spent on CAE profile generation is bound to be in the range of about 5%–15% of the overall time spent on encoding. Moreover, in practice, since the use of CAE in most cases results in the reduction of the number of renditions in the final ABR profiles, the overall encoding and compute costs with CAE are typically lower than that without it.

Examples of Optimizations

In this section, we provide a few examples of optimizations achieved by using the above-described tools. Primarily, we will focus on optimizations to operator contexts and video content.

Adaptations to Different Networks and Devices

Let us now consider three operators with bandwidth and usage statistics as presented in **Figs. 3–5** and **Tables 2–4**, respectively. The same test video sequence is used in all cases. CAE encoding profiles generated for this sequence, given the statistics from each of the operators, are shown in **Tables 9–11**.

TABLE 6. Playback statistics for operator 2.

Statistics	PC	Mobile	Tablet	TV	All
Rendition 1	0.00798	0.00573	0.00374	0.00022	0.00452
Rendition 2	0.01475	0.0119	0.00937	0.00093	0.00953
Rendition 3	0.01193	0.01635	0.01805	0.00197	0.01319
Rendition 4	0.06136	0.05944	0.10466	0.01077	0.05341
Rendition 5	0.10589	0.05767	0.13437	0.02598	0.06098
Rendition 6	0.14685	0.07741	0.0953	0.05187	0.07794
Rendition 7	0.10422	0.07573	0.07808	0.05372	0.07306
Rendition 8	0.08825	0.07463	0.08211	0.08126	0.07756
Rendition 9	0.45717	0.61389	0.47271	0.77318	0.62495
Buffering	0.00136	0.00648	0.00141	0.00009	0.00435
Start time	1.94761	1.79835	2.00201	1.60687	1.77841
Bandwidth	3840.61	4159.37	3736.25	4655.01	4206.01
Resolution	963.553	993.104	949.804	1053.253	1000.070
SSIM	0.96267	0.96346	0.96236	0.96500	0.96364

TABLE 7. Playback statistics for operator 3.

Statistics	TV	All
Rendition 1	0.00066	0.00066
Rendition 2	0.00232	0.00232
Rendition 3	0.03165	0.03165
Rendition 4	0.02472	0.02472
Rendition 5	0.04815	0.04815
Rendition 6	0.01428	0.01428
Rendition 7	0.01874	0.01874
Rendition 8	0.02806	0.02806
Rendition 9	0.83091	0.83091
Buffering	0.00051	**0.00051**
Start time	1.58783	**1.58783**
Bandwidth	6927.70	**6927.70**
Resolution	1003.316	**1003.316**
SSIM	0.97133	**0.97133**

In all cases, the CAE profile generator was also given the same overall constraints that generally match the characteristics of the HLS reference-encoding ladder (**Table 8**). This includes constraints on the minimum and maximum bitrates, the maximum gap between adjacent bitrates, the maximum number of renditions, as well as constraints on resolutions, framerates, and pixel aspect ratios.

The resulting CAE-generated profiles for operators 1–3 are presented in **Tables 9–11**. We observe that these profiles are different. For operator 1, CAE generated seven renditions, with high density of points around 0–1 Mbit/s range. For operator 2, it also generated seven renditions, however, with faster rump toward higher resolutions and bitrates. Note, specifically, that instead of selecting 540p resolution at the fourth rendition, it selects 576p. Finally, in the case of operator 3, CAE generated only five renditions, which are even more sparsely placed apart. Such use of fewer renditions leads to lower transcoding costs and better CDN efficiency.

Besides the changes in the numbers of renditions, we also notice significant changes in bitrates in the CAE-generated profiles relative to the HLS reference profile, as shown in **Table 8**. All CAE-generated profiles require significantly lower bandwidth and storage.

One notable difference between the CAE-generated and reference HLS profiles is that CAE uses a mixed set of H.264 profiles, starting with Baseline, followed by the Main and High profiles. In contrast, the HLS ladder, recommended in the latest Apple deployment guidelines,[22] uses only the High profile across all renditions. CAE-generated profiles can therefore reach a much broader set of devices, including those that can only decode the H.264 baseline.

Next, in **Tables 12–14**, we present playback statistics as measured for the CAE-encoded content delivered by all three operators. As described earlier, the first rows in these tables list measured load probabilities for all renditions, followed by the standard QOE metrics.

Table 15 presents relative change values, computed for average numbers reported across all devices for each operator. The negative values mean that the use of CAE leads to the reduction in the value of the respective parameter by the given percentage. The positive values imply the increase in the parameter value due to the use of CAE.

Based on the information presented in **Table 15**, it can be observed that the use of CAE resulted in significant savings of resources in all the three cases. The number of renditions and, consequently transcoding and compute costs, are reduced by 22.2%–44.4%. The amount of storage is reduced by 56.9%–68.7%, thereby, reducing cloud storage costs. Savings in the average bandwidth usage are also significant but are more dependent on the operator's context. For example, for operators 2 and 3, which deliver mostly over high-speed networks, we observe bandwidth savings in the range of 31.3% to 33.9%.

TABLE 8. Standard HLS encoding ladder[22] and SSIM quality levels achieved for the content used in the experiment.

Rendition	Profile	Resolution	Framerate	Bitrate	SSIM
1	High	416 × 234	23.976	145	0.92231
2	High	640 × 360	23.976	365	0.94337
3	High	768 × 432	23.976	730	0.95776
4	High	768 × 432	23.976	1100	0.96788
5	High	960 × 540	23.976	2000	0.97148
6	High	1280 × 720	23.976	3000	0.96931
7	High	1280 × 720	23.976	4500	0.9753
8	High	1920 × 1080	23.976	6000	0.96861
9	High	1920 × 1080	23.976	7800	0.97217
Storage				25640	

TABLE 9. CAE-generated encoding ladder for operator 1.

Rendition	Profile	Resolution	Framerate	Bitrate	SSIM
1	Baseline	320 × 180	30	125	0.93369
2	Baseline	480 × 270	30	223.08	0.93793
3	Main	640 × 360	30	398.11	0.94636
4	Main	960 × 540	30	774.78	0.94953
5	Main	1280 × 720	30	1549.5	0.95637
6	High	1600 × 900	30	2765.3	0.96105
7	High	1920 × 1080	30	4935.1	0.96576
Storage				10771	

TABLE 10. CAE-generated encoding ladder for operator 2.

Rendition	Profile	Resolution	Framerate	Bitrate	SSIM
1	Baseline	320 × 180	30	125	0.93338
2	Baseline	480 × 270	30	239.71	0.94122
3	Main	640 × 360	30	469.54	0.95202
4	Main	1024 × 576	30	939.08	0.95221
5	Main	1280 × 720	30	1568.8	0.95658
6	High	1600 × 900	30	2765.3	0.96105
7	High	1920 × 1080	30	4935.1	0.96576
Storage				11026	

For operator 1, delivering over slower connections (with average bandwidth around 3.3 Mbits/s), the reduction in bandwidth usage is more modest—about 8.44%. However, the average resolution delivered to this operator is over 27% higher (804 lines on average versus 631), and the average start-up latency also decreases by over 5.7%. In other words, the use of CAE optimizations for operator 1 resulted in an increase of QOE, apart from savings in bandwidth, storage, and compute costs.

This comparison shows the benefits that can be achieved by optimizing encoding profiles to operator's networks and device usage statistics.

Adaptations to Different Types of Content

As discussed earlier, as part of the overall optimization process, the CAE profile generator also adapts profiles to specific properties of each input content. For example, for "easier" to encode content, such as cartoons or screen captures, CAE may assign lower bitrates or higher resolutions at the same bitrates, whereas for more "complex" content, such as high-action sports or movies, it may assign higher bitrates or lower resolutions at the same bitrates.

To estimate the average savings that can be achieved for different categories of content, we have performed a study using 500 video assets, with a combined duration of over 120 hr, and representing 33 different categories, such as action movies, sports, documentary, etc.

Such content was encoded using the standard HLS profile (**Table 8**) and by using CAE. Both versions of the content were subsequently delivered to the viewers and playback statistics have been captured. The same operator was used in both the tests.

The results are summarized in **Table 16**. All numbers represent relative changes between respective statistics

TABLE 11. CAE-generated encoding ladder for operator 3.

Rendition	Profile	Resolution	Framerate	Bitrate	SSIM
1	Baseline	320 × 180	30	125	0.93447
2	Baseline	512 × 288	30	307.42	0.94855
3	Main	960 × 540	30	803.59	0.95050
4	Main	1280 × 720	30	1727.8	0.95864
5	High	1920 × 1080	30	5050.7	0.96599
Storage				8014.6	

TABLE 12. Playback statistics for operator 1 after CAE optimization.

Statistics	PC	Mobile	Tablet	TV	All
Rendition 1	0.00084	0.00678	0.00398	0.00247	0.00662
Rendition 2	0.00359	0.01851	0.00856	0.00593	0.01794
Rendition 3	0.01834	0.07164	0.04614	0.02805	0.07016
Rendition 4	0.04087	0.13809	0.09536	0.08767	0.13564
Rendition 5	0.10114	0.17519	0.17164	0.08743	0.17485
Rendition 6	0.21248	0.37255	0.39131	0.32508	0.3735
Rendition 7	0.62253	0.21339	0.27992	0.46209	0.21747
Buffering	0.00021	0.00385	0.00309	0.00128	**0.00382**
Start time	2.56661	3.95220	3.49462	2.91179	**3.92152**
Bandwidth	3857.24	2504.93	2832.92	3399.98	**2524.53**
Resolution	966.381	801.556	851.838	915.236	**804.521**
SSIM	0.96266	0.95797	0.95948	0.96119	**0.95806**

TABLE 13. Playback statistics for operator 2 after CAE optimization.

Statistics	PC	Mobile	Tablet	TV	All
Rendition 1	0.00248	0.00357	0.00153	0.00008	0.00258
Rendition 2	0.01192	0.00604	0.00513	0.00037	0.00512
Rendition 3	0.01402	0.01654	0.01427	0.00158	0.01301
Rendition 4	0.03352	0.03715	0.05427	0.00538	0.03177
Rendition 5	0.11148	0.07551	0.16928	0.02499	0.07564
Rendition 6	0.20711	0.1134	0.14396	0.07515	0.11391
Rendition 7	0.61811	0.74131	0.61015	0.89236	0.75362
Buffering	0.00136	0.00648	0.00141	0.00009	**0.00435**
Start time	1.94563	1.79721	2.00044	1.60611	**1.77729**
Bandwidth	3844.52	4162.01	3739.18	4657.22	**4208.66**
Resolution	963.553	993.104	949.804	1053.25	**1000.07**
SSIM	0.96274	0.96352	0.96242	0.96507	**0.96370**

obtained for encoding produced using the default HLS ladder (**Table 8**) versus CAE. For compactness of presentation, only the changes in renditions, storage, bandwidth, and resolution are presented. The changes in other statistics were minor (<2%).

TABLE 14. Playback statistics for operator 3 after CAE optimization.

Statistics	TV	All
Rendition 1	0.00064	0.00064
Rendition 2	0.00555	0.00555
Rendition 3	0.04259	0.04259
Rendition 4	0.0785	0.0785
Rendition 5	0.87229	0.87229
Buffering	0.00043	**0.00043**
Start time	1.56135	**1.56135**
Bandwidth	4579.37	**4579.37**
Resolution	1023.74	**1023.74**
SSIM	0.96464	**0.96464**

By looking at the data in **Table 16**, it can be observed that CAE improvements are significant across all categories of content. We also note that for some categories of content, such as "Interviews" or "Golf," the changes in bandwidth are extremely high (we see savings of about 74%), whereas for some other categories, such as "Swimming" or "Hockey," such savings are considerably

TABLE 15. Effects of CAE optimization for three operators.

Statistic	Relative changes (%) for each operator		
	Operator 1	Operator 2	Operator 3
Renditions	–22.222	–22.222	–44.444
Storage	–57.991	–56.932	–68.741
Bandwidth	–8.4402	–31.307	–33.897
Resolution	+27.373	+6.5968	+2.0362
SSIM	–0.9003	–0.7447	–0.6895
Buffering	–1.7494	–1.0493	–1.5686
Start time	–5.7035	–1.0081	–1.6676

TABLE 16. Average savings as measured for different content categories, operator 2.

Category	Relative changes (%) due to using CAE			
	Renditions	Storage	Bandwidth	Resolution
Action	–35.05	–77.28	–59.16	+3.57
Adventure	–29.63	–70.17	–51.33	+3.32
Comedy	–25.12	–62.16	–41.28	+2.33
Drama	–32.36	–73.29	–55.83	+3.55
Scifi	–31.38	–71.89	–53.17	+3.27
Cartoon	–30.15	–68.82	–47.71	+2.93
Video game	–29.2	–67.76	–46.17	+3.17
Baseball	–21.57	–61.09	–50.89	+0.76
Basketball	–22.1	–57.82	–34.15	+1.72
Boxing	–23.71	–65.33	–43.03	+3.1
Cricket	–14.29	–58.12	–50.13	+0.97
Cycling	–23.11	–58.92	–36.55	+2.35
Field hockey	–22.22	–51.57	–22.66	+1.1
Football	–28.57	–79.12	–52.25	+1.69
Golf	–28.57	–79.38	–74.2	+1.69
Gymnastics	–26.1	–65.45	–44.01	+2.79
Hockey	–22.22	–51.26	–20.39	+0.08
Mixed sports	–23.63	–55.47	–29.22	+1.35
Racing	–28.57	–74.68	–66.96	+1.5
Running	–23.3	–56.66	–31.99	+2.52
Squash	–27.56	–67.18	–47.11	+3.22
Swimming	–22.22	–50.04	–19.67	+0.17
Tennis	–18.72	–61.04	–51.44	+1.07
Weightlifting	–31.44	–72.6	–51.66	+3.78
Documentary	–25.72	–59.85	–34.19	+2.19
Game show	–28.16	–65.18	–40.95	+3.02
Interview	–37.33	–81.17	–74.2	+1.6
Kids channel	–24.75	–59.52	–34.04	+1.69
Talk show	–36.07	–77.76	–59.02	+3.99
News	–25.97	–62.36	–39.64	+2.24
Reality TV	–24.94	–58.51	–33.52	+2.46
Sitcom	–31.49	–71.93	–54.04	+3.23
Soap opera	–34.92	–76.61	–58.83	+3.8
Overall	**–28.42**	**–65.64**	**–43.76**	**+2.65**

lower (19%–22%). The savings in storage are more consistent across all categories of content. The changes in the numbers of renditions are also more consistent across all categories of content.

The above study was done using the H.264 encoder and HD/8-bit standard dynamic range (SDR) content. In our experience, we also noticed that when using the HEVC codec, CAE savings are generally similar in magnitude and have the same general dependency on the characteristics of the content. The relative savings in the cases of 4K content are typically higher, as CAE manages to save even more renditions and bits relative to typical static ladders.[22]

Multicodec Profile Optimizations

One of the features of the CAE profile generator is the ability to generate ABR profiles for plurality of existing codecs. In this case, the generator also uses information on the support of such codecs by different categories of receiving devices. Such information is supplied as part of operator usage and bandwidth statistics, provided by the analytics engine.

The use of multicodec profile generation leads to additional savings in the total number of renditions, as well as quality gains achievable by clients that can switch

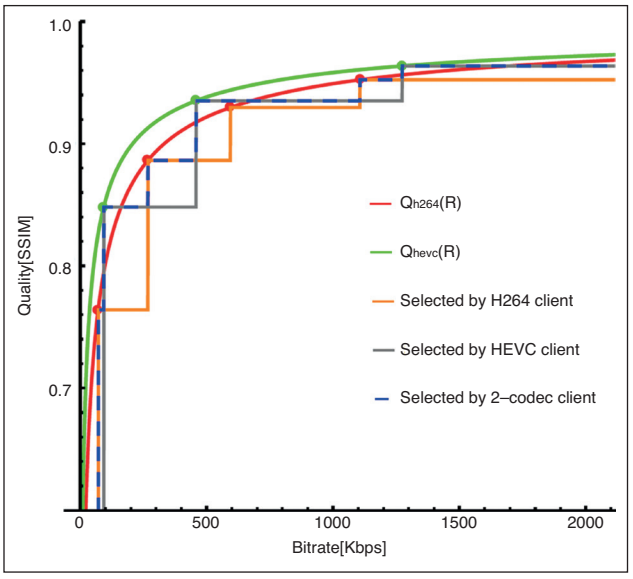

FIGURE 6. H.264 and HEVC-encoding ladders and quality levels achievable by different types of client devices.

between the codecs. To illustrate this, let us consider a mixed H.264+HEVC ladder presented in **Fig. 6**.

The green and red curves in the figure show shapes of quality-rate functions achievable by the HEVC and H.264 codecs, respectively. The points along these curves represent characteristics of renditions included in the encoding profile. The orange "staircase" represents the set of quality levels that is achievable by the H.264-only client as it switches between renditions. Similarly, the gray "staircase" represents quality levels achievable by the HEVC-only client. The dashed blue line "staircase" shows the composition of quality levels achievable by a client that can switch between both codecs. By following the shape of this dashed blue staircase, it becomes immediately obvious that such a hybrid and/or switchable client achieves better performance than the other two clients, as it effectively operates with a finer grain ladder.

To enable such improvements, the multicodec CAE profile generator solves a rather complex optimization problem, where the overall optimization cost function includes a weighted sum of average quality values achievable by H.264-only, HEVC-only, and switchable H264/HEVC clients, and where the weights are based on the distribution of devices of each kind as specific to an operator. The formal definition of this problem as well as analysis and more results can be found in Ref. 19.

Conclusion

We have described the architecture of a large-scale multiscreen OTT video delivery system. This system was designed for the effective handling of plurality of codecs, DRMs, and formats as needed for delivery to a population of client devices with different capabilities. We have also described specific tools and techniques that were added to optimize the end-to-end performance of such systems. The effectiveness of the proposed techniques has been illustrated with system statistics, before and after optimizations.

References

1. D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, "Streaming Video Over the Internet: Approaches and Directions," *IEEE Trans. Circuits Syst. Video Technol.*, 11(3):282–300, Mar. 2001.
2. G. J. Conklin, G. S. Greenbaum, K. O. Lillevold, A. F. Lippman, and Y. A. Reznik, "Video Coding for Streaming Media Delivery on the Internet," *IEEE Trans. Circuits Syst. Video Technol.*, 11(3):269–281, Mar. 2001.
3. International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) 14496-10:2003, "Information Technology—Coding of Audio–Visual Objects—Part 10: Advanced Video Coding," Dec. 2003.
4. International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) 23008-2:2013, "Information Technology—High Efficiency Coding and Media Delivery in Heterogeneous Environments—Part 2: High Efficiency Video Coding," Dec. 2013.
5. R. Pantos and W. May, "HTTP Live Streaming, RFC 8216," Aug. 2017. [Online]. Available: https://tools.ietf.org/html/rfc8216
6. International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) 23009-1:2012, "Information Technology—Dynamic Adaptive Streaming Over HTTP (DASH)—Part 1: Media Presentation Description and Segment Formats," Feb. 2012.
7. International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) 23000-19:2018, "Information Technology—Coding of Audio–Visual Objects—Part 19: Common Media Application Format (CMAF) for Segmented Media," Jan. 2018.
8. "FairPlay Streaming." [Online]. Available: https://developer.apple.com/streaming/fps/
9. "PlayReady." [Online]. Available: https://www.microsoft.com/playready/overview/
10. "Widevine." [Online]. Available: https://www.widevine.com/solutions/widevine-drm
11. Worldwide Web Consortium (W3C), "Media Source Extensions." [Online]. Available: https://www.w3.org/TR/media-source/
12. Worldwide Web Consortium (W3C), "Encrypted Media Extensions." [Online]. Available: https://www.w3.org/TR/encrypted-media/
13. "Microsoft Smooth Streaming." [Online]. Available: https://www.iis.net/downloads/microsoft/smooth-streaming
14. J. Ozer, "Encoding for Multiple Devices," *Streaming Media Magazine*, Mar. 2013. [Online]. Available: http://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=88179&fb_comment_id=220580544752826_937649
15. A. Aaron, Z. Li, M. Manohara, J. De Cock, and D. Ronca, "Per-Title Encode Optimization," 15 Dec. 2015. [Online]. Available: https://medium.com/netflix-techblog/per-title-encode-optimization-7e99442b62a2
16. Ultra HD Forum, "Ultra HD Forum Phase B Guidelines," Apr. 2018. [Online]. Available: https://ultrahdforum.org/wp-content/uploads/Ultra-HD-Forum-Phase-B-Guidelines-v1.0.pdf
17. Y. A. Reznik, K. O. Lillevold, A. Jagannath, J. Greer, and J. Corley, "Optimal Design of Encoding Profiles for ABR Streaming," *Proc. 23rd Packet Video Workshop* (*PV'2018*), Amsterdam, The Netherlands, pp. 43–47, 12 Jun. 2018.
18. C. Chen, Y. Lin, S. Benting, and A. Kokaram, "Optimized Transcoding for Large Scale Adaptive Streaming Using Playback Statistics," *Proc. 25th IEEE Int. Conf. Image Processing* (ICIP), Athens, pp. 3269–3273, Oct. 2018.

19. Y. A. Reznik, X. Li, K. O. Lillevold, A. Jagannath, and J. Greer, "Optimal Design of Multi-codec Profiles for ABR Streaming," in *Proc. IEEE Int. Conf. Multimedia Expo* (ICME'2019), Shanghai, China, 8–12, pp. 348–353, Jul. 2019.
20. International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) 13818-1:2019, "Information Technology—Generic Coding of Moving Pictures and Associated Audio Information: Systems—Part 1: Systems," *ISO/IEC*, Jun. 2019.
21. International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) 14496-12:2015, "Information Technology—Coding of Audio–Visual Objects—Part 12: ISO Base Media File Format," 2015.
22. Apple Inc., "HLS Authoring Specification for Apple Devices," Sep. 2018. [Online]. Available: https://developer.apple.com/documentation/http_live_streaming/hls_authoring_specification_for_apple_devices
23. Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Trans. Image Process.*, 13(4):600–612, Apr. 2004.

About the Authors

Yuriy Reznik is a technology fellow and head of research at Brightcove. Previously, he worked at RealNetworks, InterDigital, and Qualcomm, and contributed to a number of well-known standards (ITU-T H.264/MPEG-4 AVC, ITU-T H.265/MPEG HEVC, MPEG DASH, etc.) as well as products in the domains of internet streaming and wireless multimedia. He received his PhD degree in computer science from Kiev University, Kiev, Ukraine, and was also a visiting scholar at Stanford University, Stanford, CA. He has co-authored more than 100 academic articles and co-invented more than 60 granted U.S. patents related to his studies and work.

Xiangbo (Bill) Li is a research engineer at Brightcove. He is working on streaming performance analysis, advanced analytics, and end-to-end optimizations. He holds a PhD degree in computer science from the University of Louisiana at Lafayette, Lafayette, LA. He is a coauthor of 12 academic articles and several U.S. patent applications.

Karl Lillevold is a distinguished engineer at Brightcove. He is leading the work on context-aware encoding and other advanced video compression technologies in Brightcove products. Previously, he worked at Telenor Research, Intel, and RealNetworks, and contributed to several generations of video coding standards (MPEG-2 H.263, H.26L/H.264/AVC), as well as to the family of RealVideo codecs used in pioneering products for internet streaming. He holds an MS degree in electrical engineering from the Norwegian Institute of Technology, Trondheim.

Robert Peck is an engineering manager at the Brightcove Arizona office, working on the Brightcove Dynamic Delivery platform focusing on its Service Side Ad Insertion capabilities. Prior to joining Brightcove, he worked at Limelight Networks on their content delivery network configuration application programming interface. He graduated from Arizona State University, Tempe, AZ with a BSc degree in computer science.

Thom Shutt is an engineering manager at Brightcove, U.K., working on the Dynamic Delivery and Delivery Rules projects. Shutt has previously worked as a video engineer for the British Broadasting Corp., building out the next generation of their "iPlayer" platform.

Peter Howard is a solutions engineer for Brightcove, ANZ. He supports a variety of customers, with a particular focus on over-the-top (OTT) delivery, both ad-supported and subscription-based, across live and premium video-on-demand (VOD). He works to identify solutions that can scale, while finding the best fit for business and user needs.

Presented at the METexpo19 Conference & Exhibition, Sydney, Australia, 17–19 July 2019.