Toward Efficient Multicodec Streaming

By Yuriy A. Reznik

Abstract

One of the biggest challenges in modern-era media delivery systems is the fragmentation of the population of receiving devices in terms of codec capabilities. For example, modern Apple devices can decode and seamlessly switch between H.264/Advanced Video Coding (AVC) and high efficiency video encoding (HEVC) streams. Most new TVs or set-top boxes can also decode HEVC, but they cannot switch between HEVC and H.264/AVC streams in the same streaming session. And there are still plenty of older devices/streaming clients that can only receive and decode H.264/AVC streams. With the arrival of next-generation codecs, such as AV1 and Versatile Video Coding (VVC), the fragmentation of codec support across devices becomes even more complex. This situation prompts the question—how can we serve such a population of devices most efficiently by using codecs delivering the best performance in all cases yet producing the minimum possible number of streams such that the overall cost of media delivery is minimal? In this article, we explain how this problem can be formalized and solved at the stage of dynamic generation of encoding profiles for adaptive bitrate (ABR) streaming. The proposed solution effectively generalizes the per-title or context-aware encoding (CAE) class of techniques, considering multiple sets of renditions generated using each codec and codec usage distributions by the population of the receiving devices. We also discuss additional system-level means (proper manifest generation, HLS and Dynamic Adaptive Streaming over HTTP (DASH) quality annotations, device detection, and edge logic) needed to make the proposed solution practically deployable.

Keywords

Adaptive bitrate (ABR) streaming, multicodec streaming, HLS, Dynamic Adaptive Streaming over HTTP (DASH), H.264, high efficiency video encoding (HEVC), AV1

Digital Object Identifier 10.5594/JMI.2023.3263499
Date of publication: XX XXX XXXX

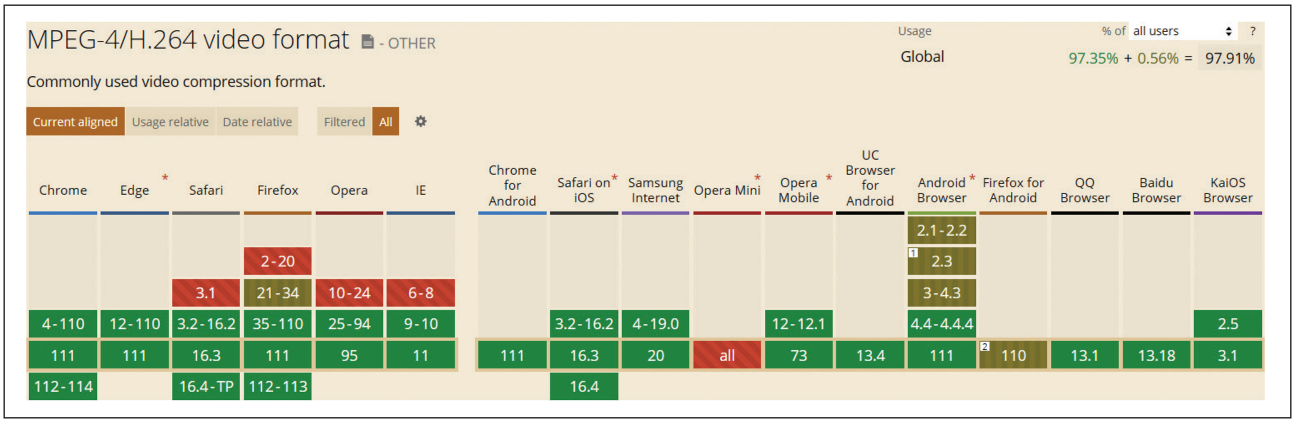
Introduction

During the last decade, most video streams sent over the internet have been encoded using the ITU-T H.264/MPEG-4 Advanced Video Coding (AVC) video codec.[1] Developed in the early 2000s, H.264/AVC has become broadly supported on various devices and platforms. According to the www.caniuse.com analytics website,[2] the current reach of H.264 across web platforms is approaching an overwhelming 97.91%.

However, in terms of technology, H.264/AVC codec is pretty old. Several more advanced codecs have been introduced in recent years. The two most well-known are the ISO/ISO MPEG and ITU-T standards committees' HEVC codec[3] and the Alliance for Open Media's AV1.[4] Both technologies claim a 40%–50% improvement in compression efficiency over H.264/AVC.[5,6,7] Even higher gains have been recently reported for an emerging Versatile Video Coding (VVC) coding standard.[8,7,9]

However, in terms of technology, H.264/AVC codec is pretty old. In recent years, several more advanced codecs have been introduced. The two best-known ones are: the HEVC codec from ISO/ISO MPEG and ITU-T standards committees and AV1 from the Alliance for Open Media. Both technologies claim about a 40%–50% gain in compression efficiency over the H.264/AVC. Even higher gains have been recently reported for an emerging VVC coding standard.

In theory, such gains should lead to a significant reduction in streaming costs. However, in practice, these new codecs can only reach particular subsets of existing devices or web browsers. For example, for HEVC, www.caniuse.com reports that only 18.19% of platforms fully support it, while in 69.32% of cases the support is partial, reduced to special versions of browsers, and/or the presence of hardware decode capability and OS settings needed to enable it. The situation with the support of AV1 on different devices is similar, with many devices still not supporting it. We present the most recent snapshots of caiuse.com reports for both H.264 and HEVC in **Figs. 1** and **2**, respectively. Green boxes show platforms fully supporting the codec, brown boxes show platforms with partial support, and red boxes show platforms where the codec is not supported at all. As we show in **Fig. 3**, the fragmentation of codecs support

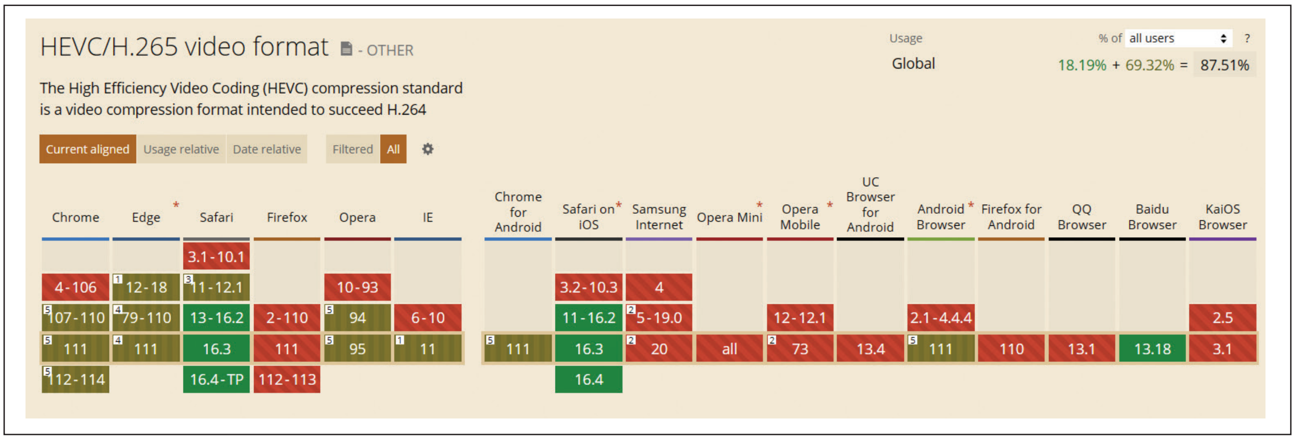

MPEG-4/H.264 video format - OTHER

Commonly used video compression format.

Usage: Global — % of all users: 97.35% + 0.56% = 97.91%

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
												2.1-2.2				
			2-20									[1] 2.3				
		3.1	21-34	10-24	6-8							3-4.3				
4-110	12-110	3.2-16.2	35-110	25-94	9-10		3.2-16.2	4-19.0		12-12.1		4.4-4.4.4				2.5
111	111	16.3	111	95	11	111	16.3	20	all	73	13.4	111	[2] 110	13.1	13.18	3.1
112-114		16.4-TP	112-113				16.4									

FIGURE 1. Support of H.264/AVC video codecs across different platforms (source: www.caniuse.com[2]).


HEVC/H.265 video format - OTHER

The High Efficiency Video Coding (HEVC) compression standard is a video compression format intended to succeed H.264

Usage: Global — % of all users: 18.19% + 69.32% = 87.51%

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
		3.1-10.1														
4-106	[1] 12-18	[3] 11-12.1		10-93			3.2-10.3	4								
[5] 107-110	[4] 79-110	13-16.2	2-110	[5] 94	6-10		11-16.2	[2] 5-19.0		12-12.1		2.1-4.4.4				2.5
[5] 111	[4] 111	16.3	111	[5] 95	[1] 11	[5] 111	16.3	[2] 20	all	[2] 73	13.4	[5] 111	110	13.1	13.18	3.1
[5] 112-114		16.4-TP	112-113				16.4									

FIGURE 2. Support of HEVC video codecs across different platforms (source: www.caniuse.com[2]).

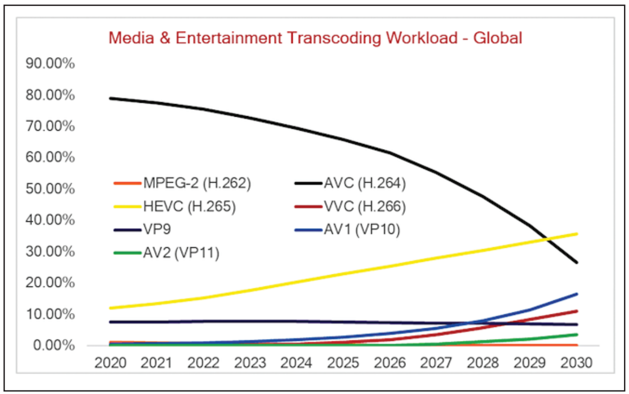


FIGURE 3. Predicted usage of video codecs in the next 10 years (source: RethinkTV[9]).

is unlikely to improve in the future, as the number of concurrently existing codecs will increase.

This situation brings a question—how do we serve such a fragmented population of devices most efficiently by using codecs delivering the best performance in all cases, yet producing the minimum possible number of streams, and such that the overall cost of media delivery is minimal?

In this article, we will explain how this problem can be formalized and solved at the stage of dynamic generation of encoding profiles for adaptive bitrate (ABR) streaming. The proposed solution effectively generalizes the per-title or context-aware encoding (CAE) class of techniques,[10,11,12,13,14,15] considering multiple sets of renditions generated using each codec and codec usage distributions by the population of the receiving devices. The proposed solution also utilizes advanced signaling mechanisms in HLS[16] and MPEG-Dynamic Adaptive Streaming over HTTP (DASH),[17] as well as current deployment guidelines and interoperability requirements developed for these standards.[18,19,20] An example of the practical realization of such a multicodec streaming system and various additional means for increased robustness and reach will also be described.

ABR Streaming: Main Principles

Let us first briefly review the main principles of operation of modern-era ABR streaming systems based on HLS[16] or DASH[17] standards. We show a conceptual diagram of such a system in **Fig. 4**. For simplicity, we only consider the video on demand (VOD) delivery case.

As shown in **Fig. 4**, when an input *video asset* is prepared for ABR streaming, it is typically transcoded into several *renditions* (or *variant streams*). Such renditions

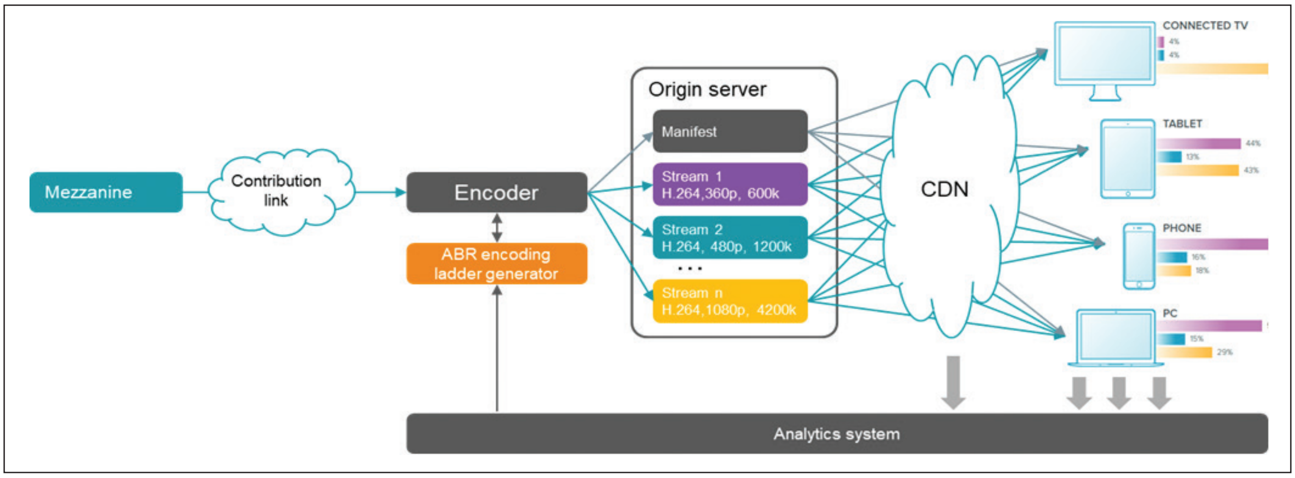


FIGURE 4. Typical architecture of HTTP-based ABR streaming system.

Table 1. Example single-codec ABR encoding ladder.

Rendition	Codec	Resolution	Frame rate	Bitrate [Kbit/s]	Quality [MOS]
1	H.264	384 × 216	25	261.59	2.178
2	H.264	512 × 288	25	513.54	2.719
3	H.264	768 × 432	25	1,024.37	3.408
4	H.264	1,280 × 720	25	2,075.71	4.215
5	H.264	1,920 × 1,080	25	4,203.03	4.769

typically have different bitrates, resolutions, and other codecs- and presentation-level parameters.

Once all renditions are generated, they are placed on an HTTP web server called the *origin server*. Along with the set of renditions, the origin server also receives *manifest files* describing the properties of the encoded streams. In the HLS streaming standard,[16] such manifests are called *playlists*, while in the MPEG-DASH standard,[17] they are called *media presentation description* or *mpd* files.

The subsequent delivery of the encoded content to user devices is done over HTTP and by using a content delivery network (CDN). The use of CDNs ensures the reliability and scalability of the delivery system.

To play the content, user devices use special software called *streaming clients*. A streaming client can be as simple as JavaScript code running in a web browser. It may also be a custom application or a video player supplied by the OS. Regardless of the implementation, most streaming clients include the logic for the adaptive selection of streams/renditions during the playback.

For example, if the client notices that the observed network bandwidth is too low to support realtime playback of the current rendition, it may decide to switch to a lower bitrate rendition. This switch prevents buffering. Otherwise, the client may switch to a higher bitrate/higher quality rendition if there is sufficient bandwidth. This switch leads to a better quality of experience. This logic is what makes streaming delivery adaptive. It is also the main reason for multiple (typically 5–10) renditions. For information about the origins of ABR streaming and other benefits of multirate design, the reader is referred to Refs.21–23.

The system depicted in **Fig. 4** has two additional components: the *analytics system*, collecting the playback statistics from CDNs and streaming clients, and the ABR *encoding ladder generator*, defining the number of renditions and properties of each rendition to create. For example, in the Brightcove VideoCloud streaming system,[24] this block corresponds to the CAE[25] module. We will explain the significance of the encoding ladder generation operation next.

Encoding Ladders

Let us now consider an example of an encoding ladder created for streaming. We present parameters of thisladder in **Table 1**. This particular example was created using the Brightcove CAE generator[25] for an action-movie asset.

This ladder defines five streams, enabling video delivery with resolutions from 216p to 1080p and using from about 260–4200 Kbit/s in network bandwidth. All streams are encoded using the H.264/AVC codec.[1] The last column in this table lists perceived visual quality scores as estimated for playback of these renditions on a PC screen. These values are reported using the standard mean opinion score (MOS) scale.[26]

In **Fig. 5**, we plot the (bitrate, quality) points corresponding to all renditions. We also show a trend of the best quality levels achievable by the streaming system with varying network bandwidth. This trend becomes a *step function*, shown in blue.

Figure 5 also plots the so-called *quality-rate model* function.[12,13,14] This function describes the quality achievable by encoding the content with the same encoder operating at every possible bitrate point within the same range of bitrates. A dashed red curve shows this function.

As it can be easily grasped, with proper ladder design, the rendition points become a subset of points from the quality-rate model, and the step function describing quality achievable by streaming becomes an approximation of this model. What influences the quality of the streaming system is the number of renditions in the encoding ladder and the placement of renditions along the bandwidth axis. The closer the resulting step function is to the quality-rate model, the better the quality that the streaming system can deliver.

This example shows that the encoding profiles/ladders for ABR streaming must be carefully designed. This is why most modern streaming systems employ special profile generators to perform this step dynamically by accounting for the properties of the content, networks, and other relevant contexts. Related techniques are known as *per-title*, *content-aware*, and *context-aware* encoding techniques.[10,11,12,13,14,15]

Multicodec Streaming: Ladders and Adaptation Models

Now that we have explained the key concepts, we can turn our attention to multicodec streaming. To make this presentation more specific, let us again consider an example of an encoding ladder generated by using two codecs: H.264/AVC and HEVC. We present parameters of this ladderin **Table 2**.

The plots of rendition points, quality-rate models, and quality achievable by streaming clients decoding only sets of H.264 and HEVC streams are presented in **Fig. 6**.

As easily observed, the quality-rate model function for HEVC is consistently better than the quality-rate model for H.264/AVC. By the same token, HEVC renditions should also deliver better quality-rate tradeoffs than renditions encoded using H.264/AVC encoder.

However, considering that there are typically only a few rendition points, and they may be placed sparsely

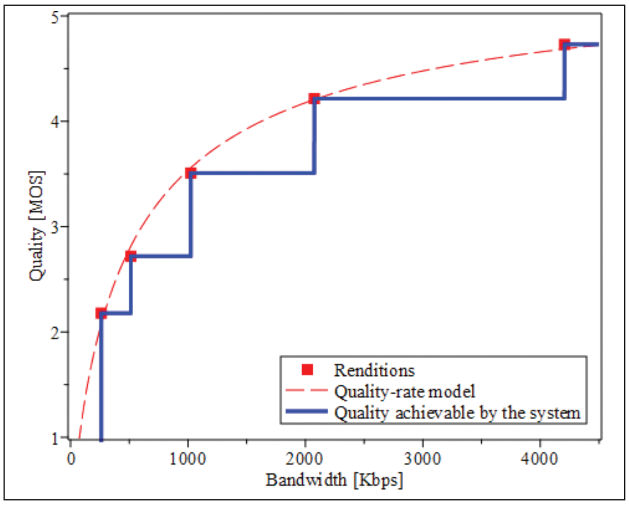


FIGURE 5. Best quality achievable by a streaming system as a function of network bandwidth.

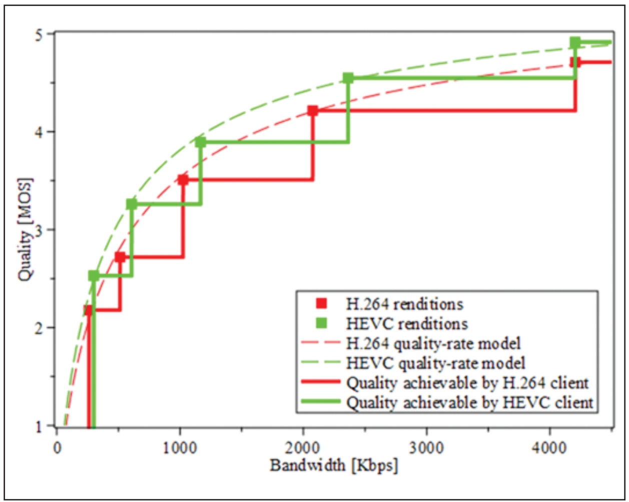


FIGURE 6. Best quality achievable by steaming systems using H.264 and HEVC codecs.

Table 2. Example encoding ladder for two codecs: H.264 and HEVC.

Rendition	Codec	Resolution	Frame rate	Bitrate [Kbit/s]	Quality [MOS]
1	H.264	384 × 216	25	261.59	2.178
2	HEVC	512 × 288	25	300	2.529
3	H.264	512 × 288	25	513.54	2.719
4	HEVC	768 × 432	25	607.89	3.260
5	H.264	768 × 432	25	1,024.37	3.408
6	HEVC	1,024 × 576	25	1,166.03	3.793
7	H.264	1,280 × 720	25	2,075.71	4.215
8	HEVC	1,600 × 900	25	2,362.74	4.549
9	H.264	1,920 × 1,080	25	4,203.03	4.769
10	HEVC	1,920 × 1,080	25	4,203.45	4.915

and in an interleaved pattern, this may create regions of bitrates, where H.264/AVC renditions may deliver better quality than the nearest HEVC rendition of smaller or equal bitrate. Such regions in **Fig. 6** are seen when step functions for H.264/AVC clients go above the same functions for HEVC clients.

What does this mean? This means that 2-codec ladder decoding of only HEVC-encoded streams does not automatically result in the best possible quality! Even better quality may be achieved by clients that selectively and intelligently switch between H.264/AVC and HEVC streams. We illustrate the quality achievable by such "2-codec clients" in **Fig.** 7.

In this example, the 2-codec client can make nine adaptation steps instead of just five in HEVC or H.264-only ladders. This enables better utilization of the available network bandwidth and delivery of better quality overall. This also allows fewer renditions using both H.264 and HEVC codecs to be generated, as both subsets are effectively used for adaptation.

Multicodec Support in Existing Streaming Clients

As we just noted, the ability of the streaming client not only to decode, but also intelligently and seamlessly switch between H.264/AVC and HEVC streams is extremely important. This leads to better quality and allows the reduction in the total number of streams, reducing streaming costs.

Perhaps, the best-known examples of existing clients supporting codec switching are native players in most new Apple devices: iPhones, iPads, Mac computers, and so on. They can decode and seamlessly switch between H.264/AVC and HEVC streams. Recent versions of Chrome and Firefox web browsers have added support for the so-called *changeType()* method, which allows JavaScript-based streaming clients to implement switching between codecs. Using this method we may expect most web-browser-based clients to add codec-switching capability in the future.

But there are indeed some other platforms, such as some SmartTVs, set-top boxes, and so on that can only decode either H.264/AVC or HEVC streams and would not switch to another codec during a streaming session. And naturally, there are also plenty of legacy devices that can only decode H.264/AVC encoded streams.

This fragmented space of streaming clients and their capabilities must be accounted for at stages of the generation of encoding ladders, properly defining HLS and DASH manifests, and design of the delivery system for multicodec streaming.

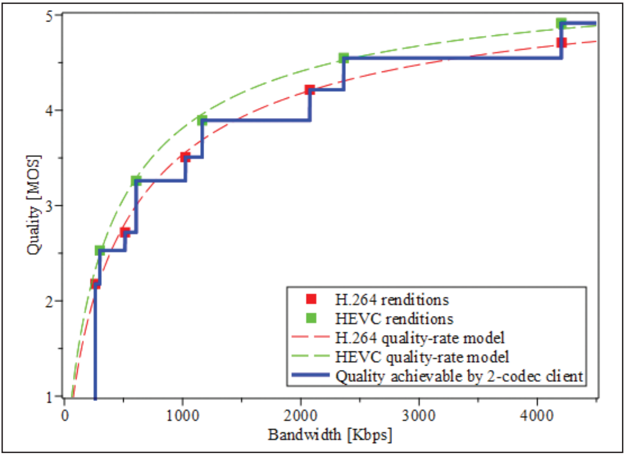


FIGURE 7. Best quality achievable by a steaming system employing "2-codec client."

Efficient Multicodec ABR Profile Generation

We will next discuss the problem of the optimal design of encoding ladders. Our presentation will closely follow the notations and concepts introduced in Refs. 12 and 13.

Model of the Delivery System

For simplicity, we will consider a case of two codecs and three clients with a conceptual diagram of the system presented in **Fig. 8**.

In practice, codec 1 is H.264/AVC, codec 2 is HEVC, client 1 is only capable of decoding H.264/AVC streams, client 2 is capable of decoding HEVC streams, and client 3 is capable of decoding both H.264/AVC and HEVC streams and can switch between them during the streaming session.

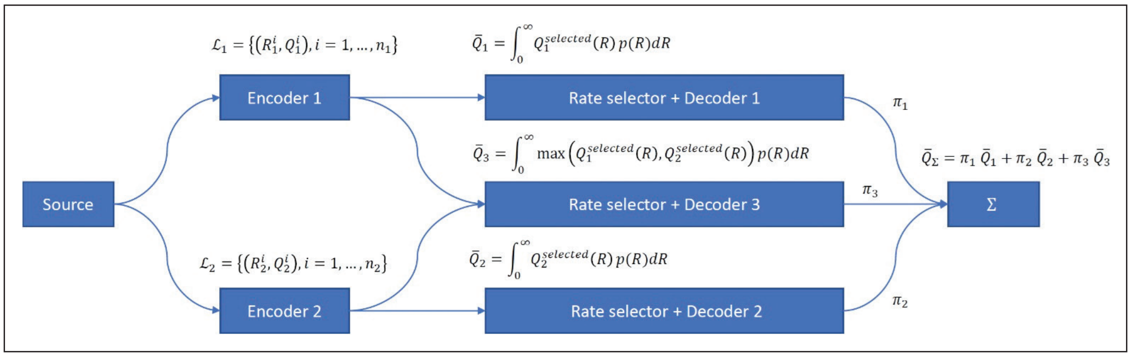

$\mathcal{L}_1 = \{(R_1^i, Q_1^i), i = 1, \dots, n_1\}$
$\bar{Q}_1 = \int_0^\infty Q_1^{selected}(R)\, p(R)dR$
Encoder 1
Rate selector + Decoder 1
π_1
$\bar{Q}_3 = \int_0^\infty \max\left(Q_1^{selected}(R), Q_2^{selected}(R)\right) p(R)dR$
$\bar{Q}_\Sigma = \pi_1 \bar{Q}_1 + \pi_2 \bar{Q}_2 + \pi_3 \bar{Q}_3$
Source
π_3
Rate selector + Decoder 3
Σ
$\mathcal{L}_2 = \{(R_2^i, Q_2^i), i = 1, \dots, n_2\}$
$\bar{Q}_2 = \int_0^\infty Q_2^{selected}(R)\, p(R)dR$
π_2
Encoder 2
Rate selector + Decoder 2

FIGURE 8. Conceptual diagram ABR streaming system with two encoders and three types of decoders/clients. Decoders 1 and 2 can decode only streams from Encoders 1 and 2, respectively. Decoder/client 3 can decode and switch between streams from both encoders.

By $\mathcal{L}_1$ and $\mathcal{L}_2$ we denote *encoding ladders* produced by using codecs of type 1 and 2, respectively. Mathematically, we will assume that each ladder is presented by a set of (rate, quality) points, corresponding to the bitrate and quality characteristics of its renditions

$$\mathcal{L}_1 = \left\{ \left(R_1^i, Q_1^i \right), i = 1, \ldots, n_1 \right\}, \text{ and}$$
$$\mathcal{L}_2 = \left\{ \left(R_2^i, Q_2^i \right), i = 1, \ldots, n_2 \right\}.$$

Here, by R_j^i we denote bitrates and by Q_j^i we denote the quality values of each rendition. The subindices indicate codec type. The upper indices indicate the rendition number in each set. The values n_1 and n_2 denote the number of renditions in each set, and $n = n_1 + n_2$ denotes the total number of all renditions used for streaming.

As in Ref. 12, we further assume that the performance of each codec can be modeled by certain *quality-rate functions*: $Q_1(R)$ and $Q_2(R)$, and that above (quality, rate) points can be understood as samples taken from these functions

$$Q_1^i = Q_1\left(R_1^i\right), \ i = 1, \ldots, n_1, \text{ and}$$
$$Q_2^i = Q_2\left(R_2^i\right), \ i = 1, \ldots, n_2.$$

Also, as in Ref. 12, the behavior of the *network bandwidth* during the streaming session is modeled as a *continuous random variable* R with known (or empirically measured) probability density function $p(R)$.

On the receiver end, we assume that clients follow *idealized client model*,[12] selecting the maximum ladder rate R^i that is less than or equal to the available network bandwidth R

$$R_1^{\text{selected}}(R) = \max_{\substack{i=1,..,n_1 \\ R_1^i \le R}} R_1^i, \text{ and } R_2^{\text{selected}}(R) = \max_{\substack{i=1,..,n_2 \\ R_2^i \le R}} R_2^i.$$

Assuming monotonically increasing quality-rate models, this also implies that

$$Q_1^{\text{selected}}(R) = \max_{\substack{i=1,..,n_1 \\ R_1^i \le R}} Q_1\left(R_1^i\right), \text{ and } Q_2^{\text{selected}}(R) = \max_{\substack{i=1,..,n_2 \\ R_2^i \le R}} Q_2\left(R_2^i\right).$$

Average Quality Achievable by the Streaming System

Next, given all the above definitions, we write expressions for the *average quality* values achievable by clients/decoders of each kind.

For instance, the *average quality* achieved by client 1 when working with a ladder $\mathcal{L}_1$ and network with bandwidth distribution $p(R)$ becomes

$$\bar{Q}_1 = \int_0^\infty Q_1^{\text{selected}}(R)\, p(R)\, dR.$$

Similarly, the *average quality* achieved by client 2, working with a ladder $\mathcal{L}_2$, becomes

$$\bar{Q}_2 = \int_0^\infty Q_2^{\text{selected}}(R)\, p(R)\, dR.$$

Finally, for client 3, which is capable of working with both ladders $\mathcal{L}_1$ and $\mathcal{L}_2$, including switching between all such streams, the expression for the average quality becomes

$$\bar{Q}_3 = \int_0^\infty \max\left(Q_1^{\text{selected}}(R), Q_2^{\text{selected}}(R)\right) p(R)\, dR.$$

The max() function in the above expression reflects the capability of a dual-codec client to select the best streams across renditions encoded by both codecs.

The average quality achieved by the entire population of clients becomes

$$\bar{Q}_\Sigma = \pi_1 \bar{Q}_1 + \pi_2\, \bar{Q}_2 + \pi_3\, \bar{Q}_3$$

where π_1, π_2, π_3 denote normalized $(\pi_1 + \pi_2 + \pi_3 = 1)$ population counts of clients of each kind.

Optimal Multicodec Ladder Design

By considering all the definitions provided, and observing that average quality value $\bar{Q}_\Sigma$ can be understood as a function of network bandwidth density $p(R)$, client distribution π, the number of ladder points n_1, n_2, and the sets of rates used in the ladder, we are now ready to define the following ladder optimization problem.

Given:

- the total number of ladder points n;
- limits for all rate points: $R_{\min}$, $R_{\max}$;
- maximum limits for first-rate points: $R_{\max}^1$;
- quality-rate functions for both codecs and content $Q_1(R)$, $Q_2(R)$;
- network bandwidth density $p(R)$; and
- distribution of clients π.

Find:

- numbers $\hat{n}_1, \hat{n}_2$, such that $\hat{n}_1 + \hat{n}_2 = n$ and
- ladder rates $\hat{R}_1^1, \ldots, \hat{R}_1^{\hat{n}_1}$ and $\hat{R}_2^1, \ldots, \hat{R}_2^{\hat{n}_2}$

such that overall quality $\bar{Q}_£$ delivered by the streaming system is maximal

$$\bar{Q}_\Sigma\left(p, \pi, n, \hat{R}_1^1, \ldots, \hat{R}_1^{\hat{n}_1}, \hat{R}_2^1, \ldots, \hat{R}_2^{\hat{n}_2}\right)$$
$$= \max_{\substack{n_1 + n_2 = n \\ R_{\min} \le R_1^1 \le \ldots \le R_1^{n_1} \le R_{\max} \\ R_{\min} \le R_2^1 \le \ldots \le R_2^{n_2} \le R_{\max} \\ R_1^1, R_2^1 \le R_{\max}^1}} \bar{Q}_\Sigma(p, \pi, n, R_1^1, \ldots, R_1^{n_1}, R_2^1, \ldots, R_2^{n_2}). \quad (*)$$

As easily noticed, this problem folds into a class of nonlinearly constrained optimization problems. The details about the design of a numerical solver for this problem and examples of solutions can be found in Ref. 13.

We note that all constraints introduced in the problem setting (*) are essential in practice. For example, the maximum rate limit $R_{\max}$ is needed to prevent the allocation of bitrates beyond those that are physically achievable. The minimum rate limit $R_{\min}$ is usually related to the minimum quality level at which streaming as a service is even feasible. The limit on the maximum first-rate in the ladder $R^1_{\max}$ is typically used to limit start-up time and/or buffering the probability of clients, and so on. In practice, several additional constraints may also be introduced.

We also note that while (*) does not explicitly operate with the choices of *video resolutions* for each stream, it is assumed that the best choices of resolutions are already absorbed in the definition of quality-rate models for each codec. In other words, given a set of allowed resolutions $\mathcal{S}$ and quality-rate models $Q_1(S,R)$, obtained for each specific resolution $S \in \mathcal{S}$, we will assume that the final quality rate model $Q_1(R)$ is defined such that

$$Q_1(R) = \sup_{S \in \mathcal{S}} Q_1(S,R).$$

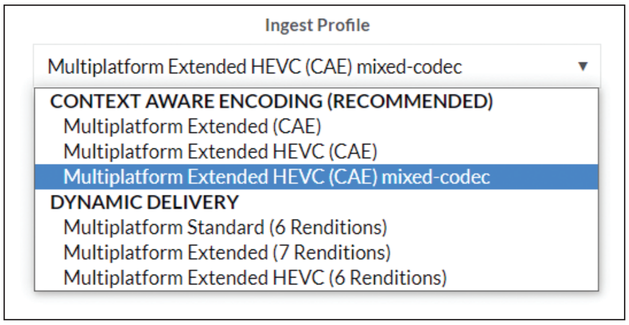
This way, the ladder optimization problem becomes effectively reduced to the choice of bitrates needed for each rendition.

Practical Implementation

The described mathematical problem and its solution finder form the basis for the design of the profile generator in the Brightcove CAE system.[25]

As shown in **Fig. 9**, the Brightcove CAE system includes several "ingest profiles," with the "Multiplatform Extended HEVC (CAE) mixed-codec" profile corresponding to a mode where both H.264 and HEVC codecs will be used in final generated streams. Our example profiles, shown earlier in **Tables 1** and **2**, have been generated by using this system.

When operating this system, the users can customize some overall ladder parameters—such as the limits on the numbers of renditions, ranges of resolutions and bitrates, assumed network and usage distributions, and so on. But the final choices for the number of renditions, and their parameters (resolutions, bitrates, codec HRD controls, etc.) are all done automatically by the CAE system.


Ingest Profile

Multiplatform Extended HEVC (CAE) mixed-codec ▾

CONTEXT AWARE ENCODING (RECOMMENDED)
- Multiplatform Extended (CAE)
- Multiplatform Extended HEVC (CAE)
- Multiplatform Extended HEVC (CAE) mixed-codec

DYNAMIC DELIVERY
- Multiplatform Standard (6 Renditions)
- Multiplatform Extended (7 Renditions)
- Multiplatform Extended HEVC (6 Renditions)

FIGURE 9. Multicodec streaming options available in Brightcove VideoCloud system.[24,25]

Using Multicodec Features of HLS and DASH

In designing multicodec streaming systems, particular attention must also be paid to the proper generation of HLS and DASH manifests.

One of the important components of such design is the inclusion of *quality attributes* that may be needed to better guide the selection of renditions by the streaming clients. This is important because in mixed-codec ladders, the increase in stream bitrates may no longer mean an increase in quality. In our example ladder in **Fig. 6**, the monotonicity of bitrate-quality relations is preserved by proper choices of codecs and bitrates, but more generally, it may not be the case. An HEVC stream with a lower bitrate may have a better quality than H.264/AVC stream using more bits. Hence quality-related attributes are important.

Additionally, there is always the need to make sure manifests are generated in a backward-compatible fashion such that older-generation clients, which can only recognize a subset of declarations, would still operate properly.

And, there are also certain new and advanced features of both HLS[16] and DASH[17] standards, as well as limitations imposed by HLS deployment guidelines[18] and DASH-IF interoperability requirements[20] that become important to account in such designs.

To explain some of these nuances, in **Figs. 10** and **11**, we show sketches of the HLS and DASH manifests constructed for a composition of H.264 and HEVC streams from the ABR ladder in **Table 2**.

As shown in these figures, in the HLS system, all HEVC and H.264/AVC renditions can be included in the natural order in the master playlist. However, in MPEG DASH, they must be listed separately, in different *adaptation sets*, defined independently for each codec. To enable switching between HEVC and AVC renditions, the "adaptation-set-switching:2016" *SupplementalProperty* descriptors must be included in each adaptation set.

Such separate placement of renditions in DASH is needed to improve compatibility with legacy players. It is required by DASH-IF Interoperability Guidelines.[19] This practice also enables conformance with CTA WAVE content specification,[20] which defines its media profiles on a per-codec level.

By looking at **Figs. 10** and **11**, we next note that the means for adding quality annotations in HLS and DASH are quite different.

In HLS, quality-related parameters are called *SCORE* attributes, with higher values indicating better quality. In MPEG-DASH, they are called *qualityRanking* attributes, but now with lower values indicating better quality. Furthermore, since HEVC and AVC renditions in DASH become split across different adaptation sets, the "qualityRanking" attributes should be produced to indicate correct and unique relative rankings values for the entire set of renditions and the "qr-equivalence:2019" *SupplementalProperty* descriptor must be included at the period

```
#EXTM3U
…
#EXT-X-STREAM-INF:BANDWIDTH=267868,CODECS="avc1.4d401e",RESOLUTION=384x216,SCORE=1, ...
Rendition1.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=307200,CODECS="hvc1.1.6.L90.90",RESOLUTION=512x288,SCORE=2, ...
Rendition2.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=525864,CODECS="avc1.4d401e",RESOLUTION=512x288,SCORE=3, ...
Rendition3.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=75378,CODECS="hvc1.1.6.L90.90",RESOLUTION=768x432,SCORE=4, ...
Rendition4.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=1048954,CODECS="avc1.4d401e",RESOLUTION=768x432,SCORE=5, ...
Rendition5.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=1194014,CODECS="hvc1.1.6.L93.90",RESOLUTION=1024x576,SCORE=6, ...
Rendition6.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=2125527,CODECS="avc1.640028",RESOLUTION=1280x720,SCORE=7, ...
Rendition7.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=2419445,CODECS=”hvc1.1.6.L120.90",RESOLUTION=1600x900,SCORE=8, ...
Rendition8.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=4303902,CODECS="avc1.640028",RESOLUTION=1920x1080,SCORE=9, ...
Rendition9.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=4304332,CODECS="hvc1.1.6.L120.90",RESOLUTION=1920x1080,SCORE=10, ...
Rendition10.m3u8
…
```

FIGURE 10. HLS master playlist (.m3u8) file created for ABR ladder from Table 2.

```
MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT1.500S" type="static“ … >
<Period duration="PT0H12M14.167S">
<SupplementalProperty schemeIdUri="urn:mpeg:dash:qr-equivalence:2019" value="1,2" />
<AdaptationSet id="1">
  <SupplementalProperty schemeIdUri="urn:mpeg:dash:adaptation-set-switching:2016" value="2" />
  <Representation id="1" mimeType="video/mp4" codecs="avc1.42001e" bandwidth="267868"   qualityRanking=”10” …/>
  <Representation id="2" mimeType="video/mp4" codecs="avc1.42001e" bandwidth="5525864" qualityRanking=”8” …/>
  <Representation id="3" mimeType="video/mp4" codecs="avc1.42001e" bandwidth="1048954" qualityRanking=”6” …/>
  <Representation id="4" mimeType="video/mp4" codecs="avc1.640028" bandwidth="2125527" qualityRanking=”4” …/>
  <Representation id="5" mimeType="video/mp4" codecs="avc1.640028" bandwidth="4303902" qualityRanking=”2” …/>
</AdaptationSet>
<AdaptationSet id="2"">
  <SupplementalProperty schemeIdUri="urn:mpeg:dash:adaptation-set-switching:2016" value="1" />
  <Representation id="1" mimeType="video/mp4" codecs="hvc1.1.6.L90.90" bandwidth="307200"   qualityRanking=”9” …/>
  <Representation id="2" mimeType="video/mp4" codecs="hvc1.1.6.L90.90" bandwidth="75378"     qualityRanking=”7” …/>
  <Representation id="3" mimeType="video/mp4" codecs="hvc1.1.6.L93.90" bandwidth="1194014" qualityRanking=”5” …/>
  <Representation id="4" mimeType="video/mp4" codecs="hvc1.1.6.L120.90" bandwidth="2419445" qualityRanking=”3” ./>
  <Representation id="5" mimeType="video/mp4" codecs="hvc1.1.6.L120.90" bandwidth="4304332" qualityRanking=”1” ./>
</AdaptationSet>
</Period>
</MPD>
```

FIGURE 11. DASH MPD file created for ABR ladder from Table 2.

level. This extra descriptor advises the client that "qualityRanking" attributes represent correct relative quality values considering the complete set of all renditions.

All other standard content-related requirements and constraints as specified in HLS authoring specification[18] and DASH-IF interoperability guidelines[20] must also be considered in the design of encoding profiles and manifests for multicodec streaming.

Additional Practical Considerations

While the above-described declarations in HLS and DASH manifest should ensure interoperability across all proper implementations of HLS and DASH clients, in practice, there may still be a risk that some legacy clients may be confused by the presence of extra renditions or manifest descriptors and will not start play of the stream or will not play it well.

Sometimes suboptimal behaviors may also be encountered with newer devices/clients, which attempt to support new codecs or features of HLS or DASH, but their implementations are not yet mature.

To minimize the likelihood of running in such situations, advanced streaming systems may employ *device detection* and *manifest filtering* logic at the edge when processing requests from players to deliver HLS or DASH manifests. Such logic may use the "user_agent" string in HTTP request headers to detect the type of the device, OS, or web browser and then dynamically decide which renditions or attributes to retain in the manifest.

For example, if the client device is classified as one that can only decode H.264/AVC streams, then only H.264/AVC streams can be retained in the manifest and the rest of them are pruned.

Such logic may generally help increase the streaming system's robustness, maintainability, and cross-platform reach. In practice, such extra logic is easily combinable with existing edge-level functions that advanced streaming systems employ for CDN selection, multiformat management, redundancy management, and other advanced delivery functions.[15,27]

Conclusion

We considered the problem of fragmentation of codec support across streaming devices. We have shown that it can be practically addressed by creating multicodec sets of renditions for HLS or DASH streaming systems with the specifically optimized design of the encoding profiles, proper generation of manifests, and manifest filtering functions of the streaming platform.

We have also shown several interesting phenomena achievable with multicodec deployments. For example, we have demonstrated that devices/clients that can decode and switch between different codecs can deliver better quality of experience than "nonswitching" clients even if they use a better codec. Another critical factor is that the total number of streams needed to support the operation of such "switching" clients can be much smaller than for clients operating with just a single codec.

These factors suggest that with growing support for codec-switching functionality in streaming clients, the concept of multicodec streaming should become increasingly more practical, cost-efficient, and deployable at a mass scale.

References

1. International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) 14496-10:2003, "Information Technology—Coding of Audio-Visual Objects—Part 10: Advanced Video Coding," ISO/IEC, Dec. 2003.
2. "Can I Use" Web Service. Mar. 18, 2022. Accessed: Apr. 14, 2023. [Online]. Available: https://caniuse.com/
3. 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," ISO/IEC, Dec. 2013.
4. AV1, "AV1 Bitstream and Decoding Process Specification, Version 1.0.0," Alliance for Open Media, Jan. 18, 2019.
5. T. Laude, Y. G. Adhisantoso, J. Voges, M. Munderloh, and J. Ostermann, "A Comparison of JEM and AV1 with HEVC: Coding Tools, Coding Efficiency and Complexity," *2018 Picture Coding Symp. (PCS)*, pp. 36–40, 2018.
6. P. Topiwala, M. Krishnan, and W. Dai, "Performance Comparison of VVC, AV1, and HEVC on 8-Bit and 10-Bit Content," *Proc. SPIE 10752, Appl. of Digital Image Process. XLI*, Sep. 17, 2018.
7. D. Grois et al., "Performance Comparison of Emerging EVC and VVC Video Coding Standards With HEVC and AV1," *SMPTE Motion Imag. J.*, 130(4):1–12, May 2021.
8. International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) 23090-3:2021, "Information Technology—Coded Representation of Immersive Media—Part 3: Versatile Video Coding," ISO/IEC, Feb. 2021.
9. RethinkTV Research, "Media & Entertainment Transcoding Workload and Device Royalty Forecast 2020-2030," Apr. 30, 2021. Accessed: Apr. 14, 2023. [Online]. Available: https://rethinkresearch.biz/reports-category/rethink-tv/#media-entertainment-transcoding-workload-and-device-royalty-forecast-2020-2030
10. A. Aaron, Z. Li, M. Manohara, J. De Cock, and D. Ronca, "Per-Title Encode Optimization," Dec. 15, 2015. Accessed: Apr. 14, 2023. [Online]. Available: https://medium.com/netflixtechblog/per-title-encode-optimization-7e99442b62a2
11. C. Chen, Y. Lin, S. Benting, and A. Kokaram, "Optimized Transcoding for Large Scale Adaptive Streaming Using Playback Statistics," *2018 25th Proc. IEEE Int. Conf. on Image Process.*, pp. 3269–3273, Oct. 2018.
12. Y. Reznik, K. O. Lillevold, A. Jagannath, J. Greer, and J. Corley, "Optimal Design of Encoding Profiles for ABR Streaming," *Proc. Packet Video Workshop*, Amsterdam, NL, pp. 43–47, Jun. 12, 2018.
13. Y. Reznik, X. Li, K. O. Lillevold, A. Jagannath, and J. Greer, "Optimal Multi-Codec Adaptive Bitrate Streaming," *Proc. IEEE Int. Conf. Multimedia & Expo*, Shanghai, China, pp. 348–353, 2019.
14. Y. Reznik, "Average Performance of Adaptive Streaming," *Proc. Data Compression Conf. (DCC'21)*, Snowbird, UT, Mar. 23–26, 2021.
15. Y. Reznik, X. Li, K. O. Lillevold, R. Peck, T. Shutt, and P. Howard, "Optimizing Mass-Scale Multi-Screen Video Delivery," *SMPTE Motion Imag. J.*, 129(3):26–38, Apr. 2020.
16. R. Pantos and W. May, "HTTP Live Streaming, RFC 8216," Internet Engineering Taskforce (IETF), 2017.
17. International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) 23009-1:2019, "Information Technology—Dynamic Adaptive Streaming Over HTTP (DASH)—Part 1: Media Presentation Description and Segment Formats," ISO/IEC, Aug. 2019.
18. Apple, "HTTP Live Streaming (HLS) Authoring Specification for Apple Devices," Apple, Nov. 12, 2021. Accessed: Apr. 14, 2023. [Online]. Available: https://developer.apple.com/documentation/http_live_streaming/http_live_streaming_hls_authoring_specification_for_apple_devices
19. DASH-IF, "DASH-IF Interoperability Guidelines, v.3," DASH-IF, Nov. 15, 2018. Accessed: Apr. 14, 2023. [Online]. Available: https://dashif.org/docs/DASH-IF-IOP-v4.3.pdf
20. CTA-5001-A, "Web Application Video Ecosystem—Content Specification," CTA, Dec. 2018. Accessed: Apr. 14, 2023. [Online]. Available: https://cdn.cta.tech/cta/media/media/resources/standards/pdfs/cta-5001-a-final.pdf
21. D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, "Streaming Video Over the Internet: Approaches and Directions," *IEEE Trans. CSVT*, 11(3):282–300, 2001.

22. B. Girod, M. Kalman, Y. J. Liang, and R. Zhang, "Advances in Channel-Adaptive Video Streaming," *Wireless Commun. Mobile Comput.*, 2(6):573–584, 2002.
23. 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. CSVT*, 11(3):269–281, 2001.
24. Brightcove VideoCloud Platform. Accessed: Apr. 14, 2023. [Online]. Available: https://www.brightcove.com/en/onlinevideo-platform
25. Brightcove Context-Aware Encoding. Accessed: Apr. 14, 2023. [Online]. Available: https://apis.support.brightcove.com/general/overview-context-aware-encoding.html
26. International Telecommunication Union—Radiocommunication (ITU-R) Rec. BT.500, "Methodology for the Subjective Assessment of the Quality of Television Pictures," ITU-R, Oct. 2019.
27. Y. Reznik, J. Cenzano, and B. Zhang, "Transitioning Broadcast to Cloud," *Proc. 2020 NAB Broadcast Eng. and Inf. Technol. Conf.*, Las Vegas, NV, May 13–14, 2020.

About the Author

Yuriy A. Reznik is a technology fellow and the vice president of research at Brightcove Inc., Boston, MA. Previously, he held engineering and management positions with InterDigital, San Diego, CA, from 2011 to 2016; Qualcomm, San Diego, CA, from 2005 to 2011; and RealNetworks, Seattle, WA, from 1998 to 2005. In 2008, he was a visiting scholar at Stanford University, Stanford, CA. Since 2001, he has been involved in the work of ITU-T SG16 and MPEG standards committees and made contributions to several multimedia coding and delivery standards, including ITU-T H.264/MPEG-4 Advanced Video Coding (AVC), MPEG-4 ALS, ITU-T G.718, ITU-T H.265/MPEG-HEVC, and MPEG-Dynamic Adaptive Streaming over HTTP (DASH). Several technologies, standards, and products that he has helped to develop (RealAudio/RealVideo, ITU-T H.264/MPEG-4 AVC, Zencoder, Brightcove CAE, and MPEG-DASH) have been recognized by the NATAS Technology and Engineering Emmy Awards. He received a PhD degree in computer science from Kyiv University, Kyiv, Ukraine. He is a Senior Member of IEEE and the SPIE, and a Member of the ACM, AES, and SMPTE. He is a coauthor of over 150 conference papers and journal articles, and a coinventor of over 70 granted U.S. patents.

This article first appeared in the Proceedings of the NAB 2022 Broadcast Engineering and Information Technology (BEIT) Conference. Reprinted here with permission of the author(s) and the National Association of Broadcasters, Washington, D.C.