Transitioning Broadcast to Cloud

By Yuriy Reznik, Jordi Cenzano, and Bo Zhang

Abstract

We analyze the differences between on-premise broadcast and cloud-based online video delivery workflows and identify means needed for bridging the gaps between them. Specifically, we note differences in the ingest protocols, media formats, signal-processing chains, codec constraints, metadata, transport formats, delays, and means for implementing operations such as ad-splicing, redundancy, and synchronization. To bridge the gaps, we suggest specific improvements in cloud ingest, signal processing, and transcoding stacks. Cloud playout is also identified as a critically needed technology for convergence. Finally, based on all such considerations, we offer sketches of several possible hybrid architectures, with varying degrees of offloading of processing in the cloud, which are likely to emerge in the future.

Keywords

Broadcast systems, cloud-based video streaming, cloud playout, internet media streaming, over the top (OTT) services, video encoding

We are now living in an era where hybrid broadcast + OTT distribution becomes the norm, which leads us to the question of how such hybrid systems can be deployed and operated most efficiently?

Currently, we have two methods:

***On-prem*: Everything, including playout systems, encoders, multiplexers, servers, and other equipment for both broadcast and OTT distribution, is installed and operated on-premises.**

***Cloud-based*: Almost everything is turned into software-based solutions and operated using the infrastructure of cloud service providers, such as AWS, GCP, Azure, and so on.**

Introduction

Historically, terrestrial broadcast TV has been the first and still widely used technology for the delivery of visual information to the masses. Cable and direct-to-home (DTH) satellite TV technologies came next, as highly successful evolutions and extensions of the broadcast TV model.[1,2]

Yet, broadcast has some limits. For instance, in its most basic form, it only enables *linear* delivery. It also provides direct reach to only one category of devices: TV sets. To reach other devices, such as mobiles, tablets, PCs, game consoles, and so on, the most feasible option currently available is to send streams over the top (OTT). The OTT model utilizes Internet Protocol (IP) connections that many such devices already support and *internet streaming* as a delivery mechanism.[3–7] The use of OTT/streaming also makes it possible to implement interactive, nonlinear, time-shifted TV, or digital video recorder (DVR) types of services.

Considering all such benefits and conveniences, many companies in the broadcast ecosystem are now increasingly adding OTT services, complementing their traditional (e.g., terrestrial, cable, satellite) services or distribution models.[8,9] At a broader scale, we must also recognize new standards and industry initiatives such as hybrid broadcast broadband TV (HbbTV),[10] as well as Advanced Television Systems Committee (ATSC) 3.0,[11,12] which are further blurring the boundaries between traditional broadcast and OTT.

We are now living in an era where hybrid broadcast + OTT distribution becomes the norm, which leads us to the question of how such hybrid systems can be deployed and operated most efficiently?

Currently, we have two methods:

- *On-prem*: Everything, including playout systems, encoders, multiplexers, servers, and other equipment for both broadcast and OTT distribution, is installed and operated on-premises.
- *Cloud-based*: Almost everything is turned into software-based solutions and operated using the infrastructure of *cloud* service providers, such as Amazon Web Services (AWS), GCP, Azure, and so on.

The on-prem model is indeed well known. This is how all traditional broadcast systems have been built and operated. The cloud-based approach is a more recent development. It requires considerably different (modular, software-only)

Digital Object Identifier 10.5594/JMI.2021.3106162
Date of publication: XX XXX XXXX

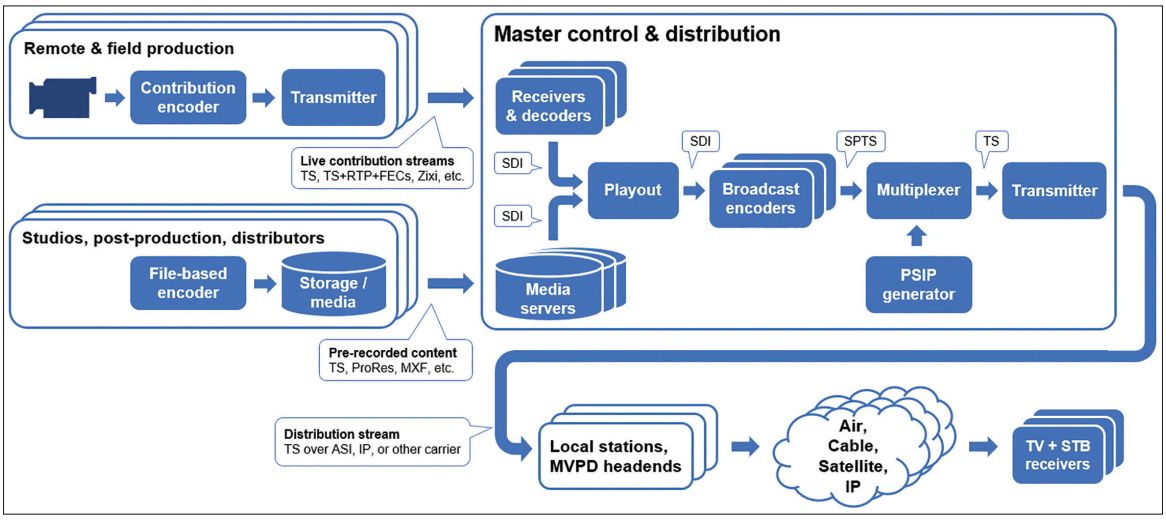

FIGURE 1. Conceptual diagram of processing operations in broadcast distribution.

implementation of the system, but in the end, it brings a number of significant advantages: it minimizes investments in hardware, allows pay-as-you-go operation, simplifies management, upgrades, makes the whole design more flexible and future-proof, and so on.[13–15]

Furthermore, the *use of the cloud has already been proved to be highly scalable, reliable, and cost-effective* for implementing *OTT/streaming delivery*. Today, the cloud already powers the largest online video services, such as YouTube or Netflix, as well as *online video platforms* (OVPs)—Brightcove, Kaltura, thePlatform, and so on.[9,16,17] Besides enabling basic streaming functionality, OVPs also provide means for content management, ad insertions, analytics, client SDKs, and even automatic generators of apps for all major platforms. They provide turn-key solutions for OTT services of all sorts.

However, while the transition of OTT services in the cloud is no longer a challenge, the offload of traditionally on-prem functions of broadcast systems, such as ingest, content management, master control/playout, distribution encoding, and so on—is a topic that we believe deserves additional discussion. As we will show in this paper, there are many important differences in the ways video processing is currently done in the cloud versus on-prem broadcast, as well as technologies that may be needed to bridge the gaps between them.

Processing Chains in Broadcast and Cloud-Based Online Video Systems

In this section, we will study *commonalities and differences* between processing chains in broadcast and online video systems. We focus on functions, formats, means for implementation of certain operations, and overall system characteristics such as *reliability, processing granularity, and delays*.

The idealized chain of processing in broadcast distribution is shown in **Fig. 1**, and the chain of processing in an online video system is shown in **Fig. 2**. Both are indeed conceptual and high level.

Main Functions and Distribution Flows

In broadcast, everything is based on processing and delivery of a set of live streams, visible to end-users as "TV channels." As shown in **Fig. 1**, the selection or scheduling of input feeds that go in each channel is done by *master control* or *playout* systems. Such systems also insert graphics (e.g., channel logos or "bugs"), slots for ads, captions, metadata, and so on.

After playout, all channel streams are subsequently encoded and passed on to a *multiplexer*, which combines them in a *multiprogram transport stream* [also known as MPEG transport stream (TS) or Motion Picture Experts Group (MPEG)-2 TS[14]] intended for distribution. In addition to the channel's media content, the final multiplex TS also carries program and system information (PSIP[18]), SCTE 35 ad markers,[19,20] and other metadata required for broadcast distribution.[21]

As shown in **Fig. 1**, the distribution chain in broadcast systems may have *multiple tiers*—from the main network center to local stations and also multichannel video programming distributors (MVPDs), such as cable or satellite TV companies. At each stage, media streams corresponding to each channel can be extracted, modified (e.g., by adding local content or ads), re-multiplexed into a new set of channels, with new program tables and other metadata inserted, and then again sent down to distribution or next headend.

In other words, broadcast systems are responsible for both the *formation of the content*, turning it into a *set of channels*, and then the *distribution* of content to the end-users.

In contrast, OVPs are used primarily for *distribution*. They assume that content is already fully formed. As shown in **Fig. 2**, live inputs are typically turned into live output streams, and prerecorded media files are

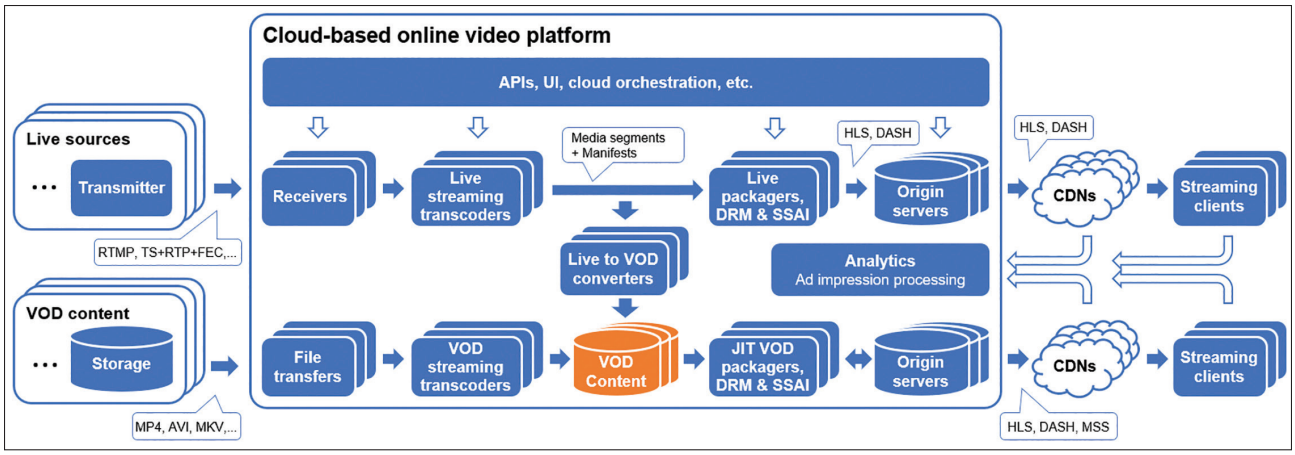

FIGURE 2. Conceptual diagram of processing operations in a cloud-based OVP.

typically published as video-on-demand (VOD) assets. They transcode and repackage inputs into HLS,[5] DASH,[6] or MSS[7] streaming formats and then pass them on to content delivery networks (CDNs) for propagation and delivery to end-user devices (streaming clients). In some cases, OVPs may also be used for live-to-VOD conversions and VOD content management, but not for the formation of live streams.

Another important difference between online video systems and broadcast is the *availability of the feedback chain*. In **Fig. 2**, it is shown by contour arrows connecting players and CDNs to the *analytics* module within OVPs. This module collects playback and CDN usage statistics and turns them into metrics used for ad monetization/billing, operations control, and optimization purposes.[22,23]

Contribution and Ingest

In broadcast, live input streams (or "feeds") originate from remote or field production. They are normally encoded by a *contribution encoder* and delivered to the broadcast center over a certain physical link (dedicated IP, satellite, 4G, etc.). The encoding is always done using one of the standard TV formats (e.g., 480i standard definition (SD) or 1080i high definition (HD), and with a number of codec- and TS-level constraints applied, making such streams compatible with broadcast systems.[24–32] When streams are sent over IP, realtime User Datagram Protocol (UDP)-based delivery protocols are normally used. Examples of such protocols include Real Time Transport Protocol (RTP),[33] SMPTE 2022-1,[34] SMPTE 2022-2,[35] Zixi,[36] and so on.

Prerecorded content usually comes in the form of files, produced by studio encoders. Again, only standard TV/broadcast video formats are used, and specific codec- and container-level restrictions are applied.[37] Moreover, in most cases, the contribution (or the so-called *mezzanine*) encodings are done at rates that are considerably higher than those used for final distribution. This allows broadcast systems to start with "cleaner" versions of the content.

In the case of OVPs, input content generally comes from a *much broader and more diverse set of sources*—from professional production studios and broadcast workflows to user-generated content. Consequently, the bitrates, formats, and quality of such streams can also vary greatly. This forces OVPs to be highly versatile, robust, and tolerant on the ingest end.

The *quality of links* used to deliver content to OVPs may also vary greatly, from dedicated connections to data centers to public internet over some local internet service providers (ISP). UDP may or may not be available.

In such a context, the ingest protocol that has most commonly deployed is Real-Time Messaging Protocol (RTMP).[38] This is an old, Flash-era protocol, with many known limitations, but it works over TCP and remains a popular choice for live to cloud ingest.

Video Formats and Elementary Streams

We next look at the characteristics of video formats used in both broadcast and online video systems. The summary of this comparison is provided in **Table 1**. For simplicity, in this comparison, we only consider SD and HD systems.

As shown in this table, SD systems almost always use *interlaced, bottom-field first (bff)* sampling format.[26,30,47] If the source is progressive (e.g., film), then it is typically converted to the interlaced form by the so-called *telecine process*.[1,2] HD systems also use interlaced formats, but with the *top-field first* (tff) order. HD systems can also carry progressive formats (e.g., 720p). In contrast, in internet streaming, only *progressive* formats are normally used.[48,49]

In terms of color parameters and SARs, streaming video formats are well aligned with high-definition television (HDTV) systems. On the other hand, streaming of SD content requires both color- and sample aspect ratio (SAR)-type conversions.

The primary reason why streaming systems are more restrictive is their *compatibility* with a wide range of possible receiving devices—mobiles, tablets, PCs, and so on.[48] In such devices, graphics stacks are simply not

TABLE 1. Comparison of video formats used in broadcast and internet streaming.

Format characteristic	Broadcast systems	Online platforms
Temporal sampling	SD: interlaced (bff), telecine HD: progressive, interlaced (tff), telecine	Progressive only
Frame rates [Hz]	24000/1001, 24, 25, 30000/1001, 30 ,50, 60000/10001, 60	Same as source, typically capped at 30 Hz or 60 Hz
Display aspect ratio (DAR)	SD: 4:3, 16:9 HD: 16:9	Same as indicated by the source
Sample aspect ratio (SAR)	SD: 1:1, 12:11, 10:11, 16:11, 40:33, 24:11, 32:11, 80:33, 18:11, 15:11, 64:33, 160:99 HD: 1:1 (most common), 4:3 (1440 mode)	1:1 or nearest rounding to 1:1 SARs are usually preferred
Resolutions	SD (NTSC-derived): 480i SD (PAL, SECAM-derived): 576i HD: 720p, 1080i	Same as source + down-scaled versions; additional restrictions may apply[39,40]
Chroma sampling	4:2:0, 4:2:2	4:2:0 only
Primary colors	SD (NTSC-derived): SMPTE-C[41,42] SD (PAL, SECAM-derived): EBU[43,44] HD: ITU-R BT.709[45]	ITU-R BT.709/ sRGB[45,46]
Color matrix	SD: ITU-R BT.601[44] HD: ITU-R BT.709[45]	ITU-R BT.709
Transfer characteristic	SD (NTSC-derived): power-law 2.2 SD (PAL, SECAM-derived): power-law 2.8 HD: ITU-R BT.709[45]	ITU-R BT.709

TABLE 2. Comparison of encoded video streams used in broadcast and internet streaming.

Stream characteristic	Broadcast systems	Online platforms
Number of outputs	Single	Multiple (usually 3-10) as needed to support ABR delivery[4,48,50]
Preprocessing	Denoising, MCTF-type filters[24,51]	Rarely used
Video codecs	MPEG-2,[52] MPEG-4/AVC[53]	MPEG-4/AVC—most deployments, HEVC,[54] AV1[55]—special cases
Codec profiles, levels	Fixed for each format. Applicable standards: ATSC A/53 P4,[26] ATSC A/72 P1,[29] ETSI TS 101 154,[30] SCTE 128[47]	Based on the target set of devices.[48] Applicable guidelines: HLS,[49] ETSI TS 103 285,[40] CTA 5001[56]
GOP length	0.5 sec	2–10 sec
GOP type	Open, closed	Closed
Error resiliency features	Mandatory slicing of I/IDR pictures, etc.	N/A
Encoding modes	CBR, VBR (with statmux) Many additional constraints apply, see ATSC A/53 P4,[26] ATSC A/54A,[27] ATSC A/72 P1,[29] ETSI TS 101 154,[30] SCTE 43,[31] SCTE 128[47]	Capped VBR Max. bitrate is typically capped to 1.1..1.5 * target bitrate; HRD buffer size is typically limited by codec profile+ level constraints;
VUI/HRD parameters	Required	Optional, usually omitted
VUI/ colorimetry data	Required	Optional, usually included
VUI/aspect ratio	Required	Optional, usually included
Picture timing SEI	Maybe required (e.g., in film mode)	Optional, usually omitted
Buffering period SEI	Optional	Optional, usually omitted
ADF/bar data/T.35	Required and carried in video ES	Not used
Closed captions	Required and carried in video ES	Optional, may be carried out of band

designed to properly render interlace, or colors other than sRGB/International Telecommunication Union – Radiocommunication (ITU-R) BT.709, or pixels that are nonsquare. This forces color-, temporal sampling-, and SAR-type conversions.

In **Table 2**, we further analyze the characteristics of *encoded video streams* (or *elementary streams*) used for broadcast and streaming distribution. Again, consideration is limited to SD and HD systems.

First, we notice that the number of encoded streams is different. In broadcast, each channel is encoded as a single stream. In streaming, each input is encoded into several output streams with different resolutions and bitrates. This is needed to accommodate adaptive bitrate (ABR) delivery.

There are also differences in codecs, encoding modes, and codec constraints. For example, in broadcast, the use of constant bitrate (CBR) encoding is most common.[24] It forces the codec to operate at a certain target bitrate, matching the amount of channel bandwidth allocated for a particular channel. The use of variable bitrate (VBR) encoding in broadcast is rare and only allowed in the so-called *statistical multiplexing* (or

statmux) regime,[57,58] where the multiplexer is effectively driving dynamic bandwidth allocation across all channels in a way that the total sum of their bitrates remains constant. In streaming, there is no need for CBR or stamux modes. All streams are typically VBR-encoded with some additional constraints applied on the buffer size and maximum bitrate of the decoder.[49]

Significantly different are also *group of pictures (GOP) lengths*. In broadcast, GOPs are typically 0.5 sec, as required for channel switching. In streaming, GOPs can be 2–10 sec long, typically limited by the lengths of segments used for delivery.

Broadcast streams also carry more *metadata*. They typically include relevant video bitstream verifier (VBV)[52] or hypothetical reference decoder (HRD) parameters,[53] picture structure-, picture timing-, and colorimetry-related information.[53] They also carry *CEA 608/708 closed captions*[59,60] and *active format descriptor (AFD)/bar data* information.[61,62] In streaming, only close captions may be present.

Finally, there are also important differences in preprocessing. Broadcast encoders are famous for the use of denoisers, MCTF filters, and other preprocessing techniques applied to make compression more efficient.[24,51] In streaming, the use of such techniques is only beginning to emerge.

Distribution Formats

As mentioned earlier, in broadcast, distribution is always done using MPEG-2 transport streams.[14] They carry audio and video elementary streams, program and system information,[18] SCTE 35 ad markers,[19] and other metadata as prescribed by relevant broadcast standards and guidelines.[21,25,27,30] TS in cable systems may also carry EBPs[63] and other cable-specific metadata.

In streaming, things are more diverse. There are several streaming formats and standards currently in use. The most prevalent ones, as of the time of writing, are as follows:

- HTTP Live Streaming (HLS)[5]
- Dynamic Adaptive Streaming over HTTP (DASH)[6]
- Microsoft Smooth Streaming (MSS).[7]

There are also several different types of digital rights management (DRM) technologies. The most commonly used ones are as follows:

- FairPlay[64]
- PlayReady[65]
- Widevine Modular. [66]

The support for these technologies varies across different categories of receiving devices. *Hence, to reach all devices, multiple streaming formats and combinations of formats and DRM technologies must be supported*. We show few common choices of such combinations in **Table 3**.

HLS, DASH, as well as MSS use the *multirate, segment-based representation* of media data. Original content is encoded at several different resolutions and bitrates and then split into segments, each starting at the GOP boundary, such that they can be retrieved and decoded separately. Along with media segments [either in TS,[14] ISOBMFF,[67] or Common Media Application Format (CMAF)[68] formats], additional files (usually called *manifests, playlists*, or *MPD* files) are provided, describing locations and properties of all such segments.

TABLE 3. Combinations of streaming formats and DRMs that can be used to reach different devices. Orange tick marks indicate possible, but less commonly used choices.

Device category	Players/platforms	HLS	DASH	HLS + FairPlay	HLS + Widevine	DASH + Widevine	DASH + PlayReady	MSS + PlayReady
PCs/browsers	Chrome	✓	✓	×	✓	✓	×	×
	Firefox	✓	✓	×	✓	✓	×	×
	IE, Edge	✓	✓	×	×	×	✓	✓
	Safari	✓	×	✓	×	×	×	×
Mobiles	Android	✓	✓	×	×	✓	✓	✓
	iOS	✓	×	✓	×	×	×	×
Set-top-boxes	Chromecast	✓	✓	×	×	✓	✓	✓
	Android TV	✓	✓	×	×	✓	✓	✓
	Roku	✓	✓	×	×	✓	✓	✓
	Apple TV	✓	×	✓	×	×	×	×
	Amazon Fire TV	✓	✓	×	×	✓	✓	✓
Smart TVs	Samsung/Tizen	✓	✓	×	×	✓	✓	✓
	LG/webOS	✓	✓	×	✓	×	×	×
	SmartTV Alliance	✓	✓	×	×	×	✓	✓
	Android TV	✓	✓	×	×	✓	✓	✓
Game Cs	Xbox One/360	✓	×	×	×	×	×	✓

Such manifests are used by players (*or streaming clients*) to retrieve and play the content.

The *carriage of metadata* in streaming systems is also more diverse. Some metadata can be embedded in media segments, whereas others may also be embedded in manifests, carried as additional "sidecar" tracks of segment files, as "event" messages,[6] or ID3 tags.[69]

For example, in addition to "broadcast-style" carriage of CEA 608/708[60,61] closed captions in video elementary streams, it is also possible to carry captions as separate tracks of WebVTT[70] or TTML[71] segments, or as IMSC1 timed text data[72] encapsulated in XML or ISOBMFF formats.[39] The preferred way of carriage depends on player capabilities and may vary for different platforms.

The SCTE 35 information is allowed to be carried only at a manifest level in HLS, by either manifest of in-band events in MPEG-DASH or only in-band in MSS.[7,39,73]

To manage such broad diversity of formats, DRMs, and metadata representations, OVPs commonly deploy the so-called *dynamic* or *just-in-time* (JIT) *packaging* mechanisms.[23] This is illustrated by an architecture shown in **Fig. 2**. Instead of proactively generating and storing all possible permutations of packaged streams on the origin server, such a system stores all VOD content in a *single intermediate representation* that allows fast transmux to all desired formats. The origin server works as a cache/proxy, invoking JIT transmuxers to produce each version of content only if there is a client device that requests it. Such logic is commonly accompanied by dynamic manifest generation, matching the choices of formats, DRMs, and metadata representation to the capabilities of devices requesting them. This reduces the amount of cloud storage needed and also increases the efficiency of the use of CDNs when handing multiple content representations.[23]

As easily observed, delivery formats and their support system in the case of OTT/streaming are completely different when compared to broadcast.

Ad Processing

In broadcast systems, there are several types of ad slots, where some are local and anticipated to be filled by local stations, and some are regional or global and are filled earlier in the delivery chain.

In all cases, insertions are done by *splicing ads* in the distribution TS streams, aided by SCTE 35[19] ad markers. Such markers (or cue tones) are inserted earlier, at playout or even production stages.[20] *Ad splicers* subsequently look for SCTE 35 markers embedded in the TS and then communicate with *Ad servers* (normally over SCTE 30[74]) to request and receive ad content that needs to be inserted. Then they update TS streams to insert a segment of ad content. Such TS update is a fairly complex and tedious process, involving re-mux, regeneration of timestamps, and so on. It also requires both main content and ads to be consistently encoded: have the same codec parameters, HDR model, and so on.[75]

In the online/streaming world, ad-related processing is quite different. The ads are usually inserted/*personalized on a per-stream/per-client basis*, and the results of viewers watching the ads (the so-called *ad impressions*) are also registered, collected, and subsequently used for monetization. It is all fully automated and has to work in realtime and at a mass scale.

There are two types for ad insertion that are used in streaming currently: server-side ad insertion (SSAI) and client-side ad insertion (CSAI).[39] In the case of CSAI, most ad-related processing resides in a client. The cloud-only needs to deliver content and SCTE 53 cue tones to the client. This scales well regardless of how cue tones are delivered—both in-band or in-manifest carriage methods are adequate.

In the case of SSAI, most ad-related processing resides in the cloud. To operate it at a high scale and reasonable costs, such processing has to be extremely simple. In this context, in-manifest carriage of SCTE 35 cue tones is strongly preferred, as it allows ad insertions to be done by *manipulation of manifests*.

For example, in the case of HLS, SCTE 35 markers in HLS playlists become substituted with sections containing URLs to ad-content segments, with extra EXT-X-DISCONTINUITY markers added at the beginning and end of such sections.[73] In the case of MPEG-DASH, essentially the same functionality is achieved by using multiple periods.[39] The discontinuity markers or changing periods are effectively forcing clients to reset decoders when switching between the program and ad content. This prevents possible HRD buffer overflows and other decodability issues during playback.

Delays, Random Access, Fault Tolerance, and Signal Discontinuities

In broadcast systems, many essential signal processing operations—format conversions, editing, switching, and so on—are normally done with uncompressed video streams, carried over by SDI,[76,77] or more recently, by SMPTE 2110[78] over IP. This enables all such operations to be performed with extremely short delays and with frame-level temporal precision. When redundant processing chains are employed, the switching between them in the SDI domain also happens seamlessly. When streams are encoded, this increases random access granularity to about 0.5 sec, which is a typical GOP length in broadcast streams.

In streaming, as discussed earlier, the delivery of encoded videos to clients is done using *segments*. Such segments cannot be made arbitrarily small due to CDN efficiency reasons. In practice, 2-, 6-, and 10-sec segments are most commonly used.[49] The *same segmented media representations are also commonly used internally in cloud-based processing workflows*. This simplifies

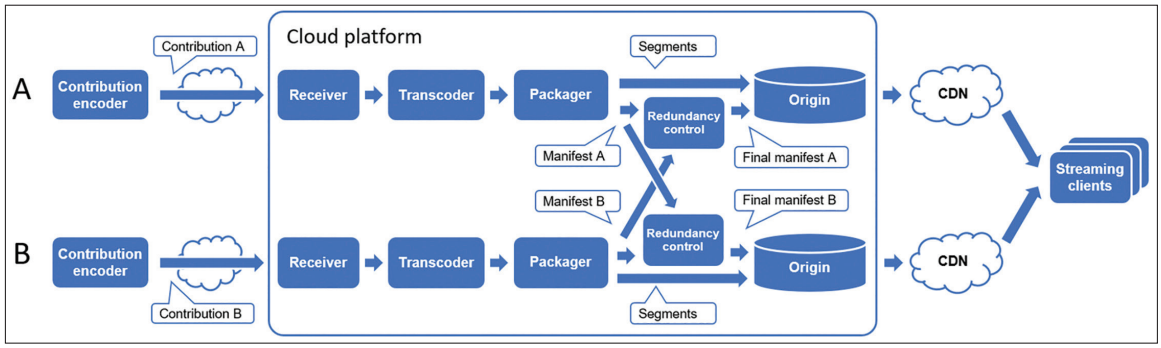

FIGURE 3. Example of cloud-based redundant live streaming workflow.

exchanges, avoids additional transcoding or transmuxing operations, and reduces many basic stream-level operations to manifest updates. However, such a design also makes *random access and delay capabilities in cloud video systems much worse compared to broadcast*.

What also makes things in the cloud complicated is the *distributed and inhomogeneous* nature of processing resources. For instance, physical servers (or cloud "instances") responsible for running video processing tasks may be located in different data centers, have somewhat different characteristics of hardware, non-synchronized local clocks, and so on. The network-induced delays in accessing such instances may also be different. Processing jobs have to be scheduled dynamically and in anticipation of all such possible differences. Moreover, occasionally, *cloud instances may become unstable, nonresponsive, or terminated by the cloud service provider.* These are rare, but relatively "normal" events. Cloud workflows must be designed to be "immune" to such events.

To illustrate how fault tolerance in the cloud may be achieved, **Fig. 3** shows an example of a live streaming system with two-way redundancy introduced. There are two contribution feeds, marked as A and B, respectively, and two processing chains, including ingest, transcoding, and packaging stages. The outputs of packagers are DASH or HLS media segments and the manifests. This system also deploys two *redundancy control* modules. These modules check whether manifest and segments' updates along route A or B are arriving at expected times, and if so—they just leave manifests unchanged. However, if they detect that either of these processing chains become nonfunctional, they update the manifest to include a discontinuity marker and then continue with segments arriving from an alternative path.

As easily observed, this system remains operational in case if either of the chains A or B fails. It also stays operational in the case of failure of one of the redundancy control units. However, what is important to note is that in the case of a failure, *the switch between videos in chains A and B may not be perfectly time-aligned*. The output stream will be decodable, but it may exhibit time-shift *discontinuity* in media content at the time of switch/fallback. This comes as a result of the operation in a distributed system with variable delays and different processing resources that may be utilized along chains A and B. Naturally, with some additional effort, the magnitude of such misalignment could be minimized, but that will necessarily increase the complexity and the delay of the system. Perfect synchronization, in principle, is one of the most challenging problems in the cloud.

The observed differences in delays, random access granularity, and also possible discontinuities in signals coming from cloud-based workflows are among the most critical factors that must be considered in planning the migration of signal processing functionality in the cloud.

Technologies Needed to Support Convergence

We next discuss measures that we believe must be taken to make cloud-based video platforms more compatible with broadcast systems.

Cloud Contribution Links and Protocols

As mentioned earlier, cloud-based video platforms typically use RTMP[38] as a protocol for live ingest. It is an old, Flash-era protocol, performing internal demux and carriage of audio and video data as separate streams sent over TCP.[38] It has no control over latencies, alters PTS/DTS timestamps, and makes it very difficult to carry SCTE 35 and other important metadata. In other words, it is inadequate for integration with broadcast workflows.

Things can be done better. Nowadays, most cloud systems can accept UDP traffic, enabling the use of protocols such as RTP,[33] RTP+SMPTE 2022-1,[34] RTP+SMPTE 2022-2,[35] Zixi,[36] SRT,[79] or RIST.[80] Such protocols can carry unaltered transport streams from contribution encoders or broadcast workflows to the cloud. Some of these protocols can also be used to send information back from the cloud to the broadcast systems.

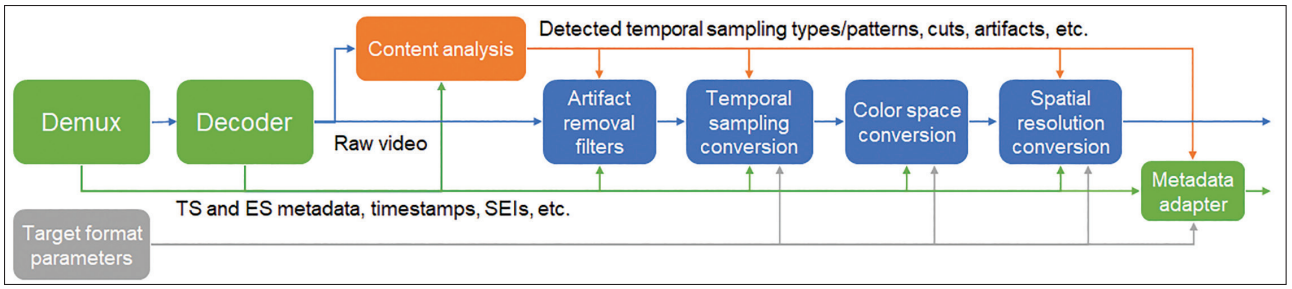

FIGURE 4. Decoding and format conversion chain needed to operate with cable/broadcast content.

The use of *dedicated connections* to cloud data centers also ultimately helps with achieving reliable ingest (as well as other exchanges between broadcast on-prem systems and cloud). Such dedicated links can be established with most major cloud operators (e.g., AWS Direct Connect,[81] Azure ExpressRoute,[82] etc.).

Signal Processing

As mentioned earlier, broadcast workflows may carry videos in progressive, interlaced, or telecine format. Field orders and pulldown patterns may also differ across different sources. When such signals are then edited or combined, this produces output with changing temporal sampling type. If such videos are then encoded and delivered as *interlaced*—they may still look OK on traditional TV sets. However, if one receives such interlace-encoded signals and then "naively" tries to convert them to progressive, the results can be disastrous, for example, a wrong assumption about field order can make videos look jerky, lack of detection of 3:2 pulldowns can produce periodic garbled frames, and so on.

Accumulations of compression and conversion artifacts make broadcast signals difficult to work with. The further down the chain the signal is obtained, the "noisier" it becomes.

To work with such complex signals, a proper processing stack is needed. One possible architecture is illustrated in **Fig. 4**. It includes a *content analysis* module, which performs detection of segment cuts and identifies types of temporal sampling patterns and artifacts in each segment. Such information, along with bitstream metadata, is then passed to a chain of filters, including artifact removal, temporal sampling conversion, color space conversion, and scaling filters.

The artifact removal filters, such as deblocking and denoising operations, are among the most basic techniques needed to work with broadcast signals. Deblocking filters are needed, for example, in working with MPEG-2 encoded content, as MPEG-2 codec[52] does not have in-loop filters and passes all such artifacts to the output. **Figure 5** shows how such artifacts look, along with the cleaned output produced by our deblocking filer. Denoising is also needed, especially when working with older (analog-converted) SD signals. Removal of low-magnitude noise not only makes the signal cleaner,

FIGURE 5. Example of MPEG-2 blocking artifacts (left) and their removal by deblocking filter (right).

but also makes the job of the subsequent encoder easier, enabling it to achieve better quality or lower rate. We illustrate this effect in **Fig. 6**.

The temporal sampling conversion filter in **Fig. 4** performs conversions between progressive, telecine, and interlace formats, as well as temporal interpolation and resampling operations. As discussed earlier, this filter is driven by information from the content analysis module. This way, for example, telecine segment can be properly converted back to progressive, interlaced, properly deinterlaced, and so on.

The quality of temporal sampling conversion operations is very critical. For example, **Fig. 7** shows the outputs of a basic deinterlacing filter (FFMPEG "yadif" filter[83]) and a more advanced optical-flow-based algorithm.[84] It can be seen that a basic deinterlacer cannot maintain continuity of field lines under high motion. The effects of such nature can be very prominent in sports broadcast content.

The use of subsequent filters, such as color space conversion and scaling filters, in **Fig. 5** is driven by possible differences in color spaces, SARs, and resolutions in input and output formats.

All such conversion operations need to be state of the art. Or at least they must be comparable in quality with Teranex,[85] Snell-Willcox/Grass Valley KudosPro,[86] and other standards converter boxes, commonly used in post-production and broadcast.

Broadcast-Compliant Encoding

As discussed earlier, broadcast and streaming workflows use encoders that are significantly different in

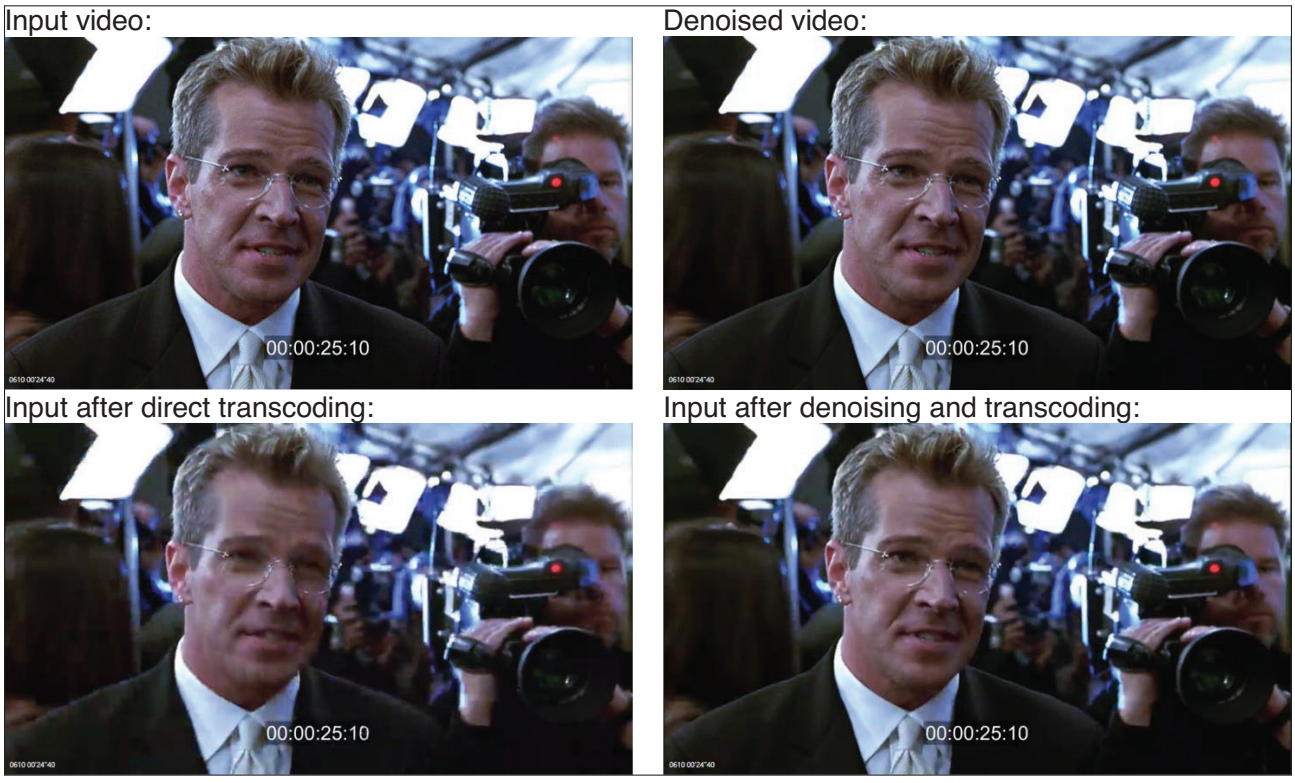

FIGURE 6. Example of using denoising filter for improving quality of the final transcoded signal.

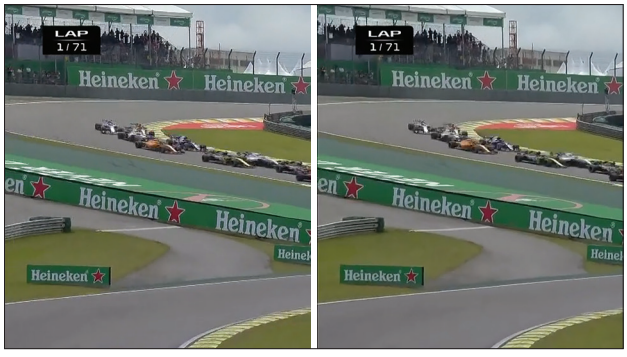

FIGURE 7. Comparison of outputs of basic (left) and advanced (right) deinterlacing filters.

their feature sets and tuning/stream constraints. Perhaps, the most extreme example of such differences is a statmux regime, where encoders are operating under the control of a multiplexer, an operating regime that has no parallel in streaming.

Consequently, if cloud workflows are intended to be used for producing streams going back to broadcast distribution, the tuning or upgrade of existing cloud encoders will be needed. For the implementation of statmux, the multiplexer should also be natively implemented in the cloud and integrated with encoders.

Cloud Playout

The last and the most important technology that is needed to enable convergence is a high-quality, cloud-based implementation of the playout system.

The design of such a system is a nontrivial task. As we discussed earlier, current cloud-based video workflows typically use HLS/DASH-type segmented media formats, causing them to operate with significant delays and random-access limitations. One cannot build a broadcast-grade playout system based on such architecture. Even the so-called *ultralow-delay* versions of HLS, DASH, or CMAF[87–89] are inadequate. For most master control operations, such as previews, nonlinear editing, switching, and so on, frame-level access accuracy is an essential requirement.

Figure 8 shows one possible cloud playout system architecture that can be suggested. To enable flame-level random access, this system uses an *internal intra-only mezzanine format*. Such a format could use any image or video codec operating in an intracoding mode, along with pulse-code modulation (PCM) audio, and index enabling access to each frame. Both live videos and the prerecorded content are then converted into an internal mezzanine format and placed in cloud storage. All subsequent operations, such as previews, nonlinear editing, as well as selection and mix of content producing channel outputs are done by accessing media in such mezzanine format. The final stream selections, the addition of logos, transitions, and so on are done by "stream processor" elements.

In addition to enabling frame-level-accurate processing operations, the use of intra-only mezzanine format also minimizes the impacts of possible failures in the system. All signal processing blocks shown in **Fig. 8** can

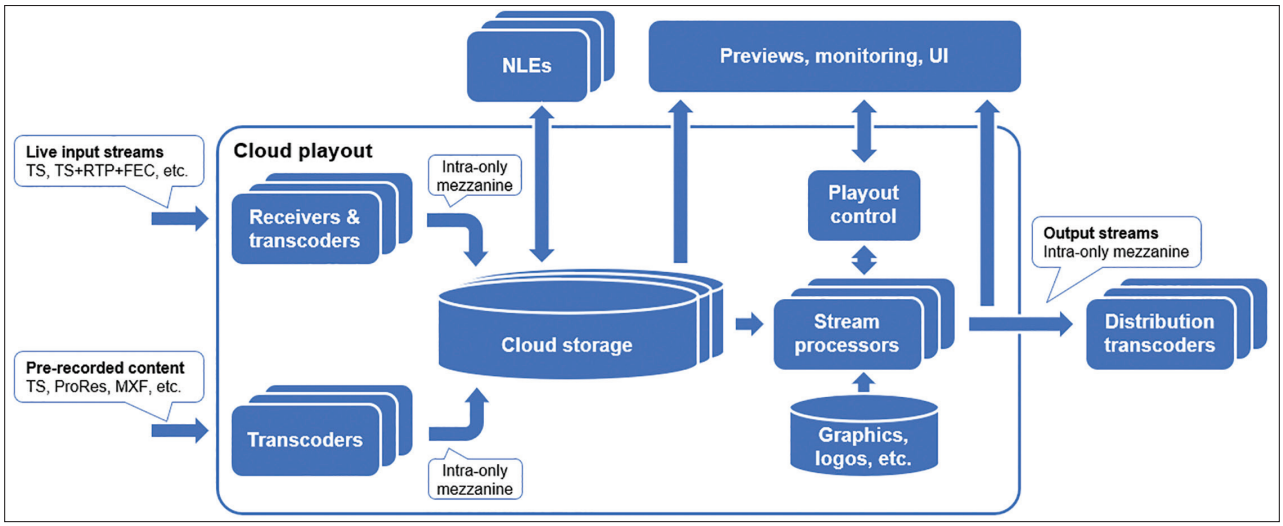

FIGURE 8. Example architecture of cloud playout system.

be run in a redundant fashion, with checks and switches added to ensure frame-accurate fault tolerance.

Transitioning Broadcast to Cloud

In this section, we finally discuss possible ways in which broadcast and cloud-based video workflows may evolve in the future. We offer three examples of possible hybrid architectures, with different degrees of migration of processing to the cloud.

Cloud-Based OTT System

Figure 9 shows a hybrid architecture where the cloud is used only to implement OTT/streaming services. Everything else stays on-prem. This is the easiest possible example of integration.

To route streams to the cloud, broadcast workflow produces contribution streams, one for each channel, and then sends them over IP (e.g., using RTP+FEC or Zixi) to the cloud.

The cloud platform receives such streams, performs necessary conversions, transcodes them, and distributes them over CDNs to clients. As shown in **Fig. 9**, the cloud platform may also be used to implement DVR or time-shift TV-type functionality, DRM protection, SSAI, analytics, and so on. All standard techniques for optimizing multiformat/multiscreen streaming delivery (dynamic packaging, dynamic manifest generation, optimized profiles, etc.[23]) can also be employed in this case.

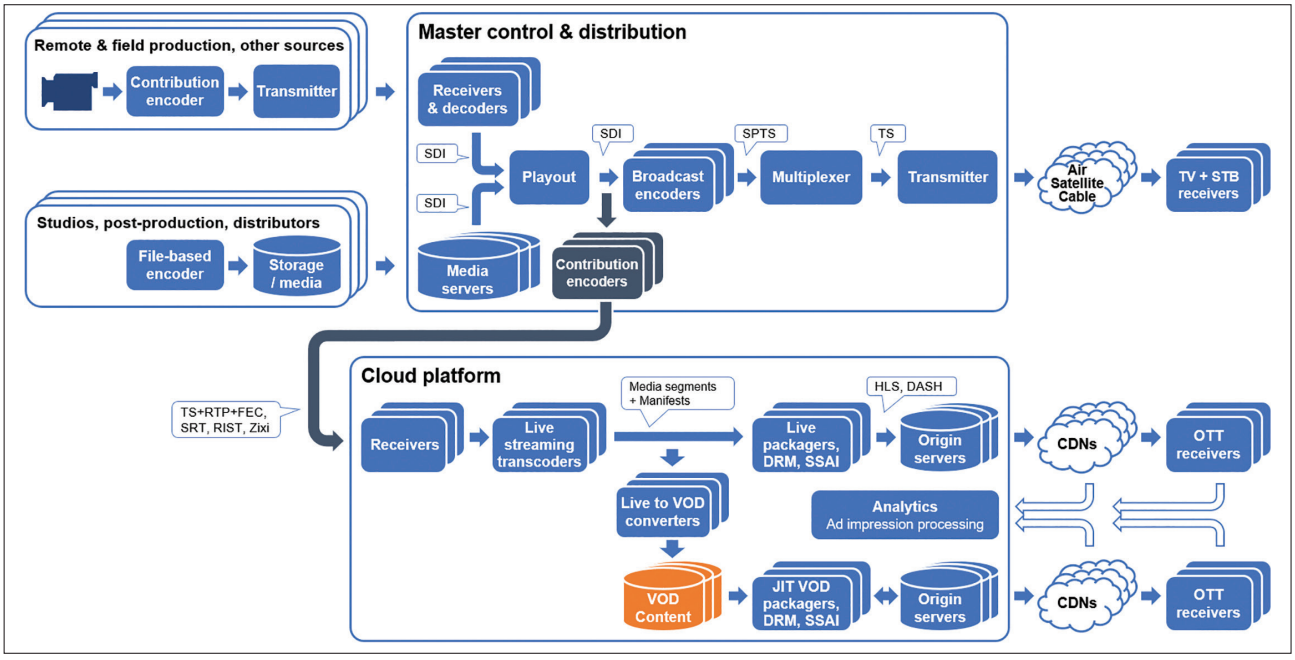

FIGURE 9. Hybrid architecture with OTT/streaming workflow offloaded to cloud.

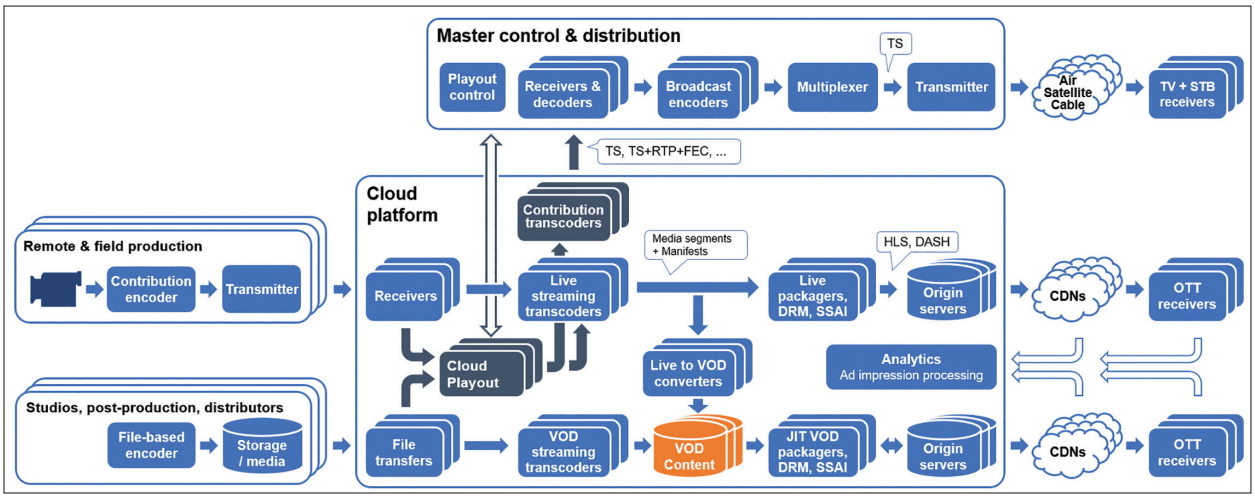

FIGURE 10. Hybrid architecture with ingest, playout, and OTT/streaming workflow offloaded to cloud.

To make such a system works well, the main technologies/improvements that are needed include:

- Reliable realtime ingest, for example, using RTP+FEC, SRT, RIST, or Zixi protocols and/or a dedicated link, such as AWS Direct connect.[81]
- Improvements in signal processing stack—achieving artifact-free conversion of broadcast formats to ones used in OTT/streaming.
- Improvements in metadata handling, including full pass-through of SCTE 35 and compliant implementation of SSAI and CSAI functionality based on it.

But generally, hybrid architectures of this kind have already been deployed and proved to be effective in practice. Some of the above-mentioned close-gap technologies have also been implemented. For example, using RTP, SMPTE 2022-1, SMPTE 2022-2, or SRT for cloud ingest, improvements in support of SCTE 35 and improvements in encoding stack were among recent updates in the Brightcove VideoCloud system.[90]

Cloud-Based Ingest, Playout, and OTT Delivery System

Figure 10 shows a more advanced architecture, in which not only OTT delivery, but also ingest, media asset management, and playout are offloaded to the cloud.

As can be immediately grasped, the move of playout functionality to the cloud also enables the use of the cloud platform for *ingest*. This is particularly helpful on a global scale, as major cloud systems have data centers in all major regions, and so the contribution link is only needed to deliver content to the nearest local datacenter. Media asset management also naturally moves to the cloud in this case.

With cloud-based playout, there will still be a need in a control room with monitors, switch panels, and so on. But it all will be reduced to a function of a thin client. All storage, redundancy management, media processing, and so on will happen in the cloud, significantly reducing required investments in hardware and operating costs.

In the system depicted in **Fig. 10**, the broadcast distribution encoding, multiplexer, and all subsequent operations stay on-prem without any changes. This way, broadcasters can operate all current ATSC equipment until ATSC 3.0 matures or there is some other serious need to replace it. This is another hybrid cloud + on-prem architecture, which we believe will make sense in practice.

To make such a system works, in addition to all improvements mentioned earlier, what is further needed is as follows:

- Cloud-based broadcast-grade playout service
- Direct link connection to cloud ensuring low latency monitoring and realtime responses in the operation of cloud playout system
- Improvements in cloud-run encoders, specifically those acting as contribution transcoders sending broadcast-compliant streams back to the on-prem system.

Cloud-Based Broadcast and OTT Delivery System

Finally, **Fig. 11** shows an architecture where pretty much all signal processing, transcoding, and multiplexing operations are moved to the cloud.

In addition to running playout, this system also runs broadcast transcoders and the multiplexer in the cloud. The final multiplex TS is then sent back to the on-prem distribution system, but mostly only to be relayed to modulators and amplifiers or (via IP or asynchronous serial interface (ASI) to next-tier stations or MVPD headends.

To make this system work, in addition to all improvements mentioned earlier, what is further needed is as follows:

- Broadcast-grade transcoders and multiplexers should be natively implemented in the cloud.

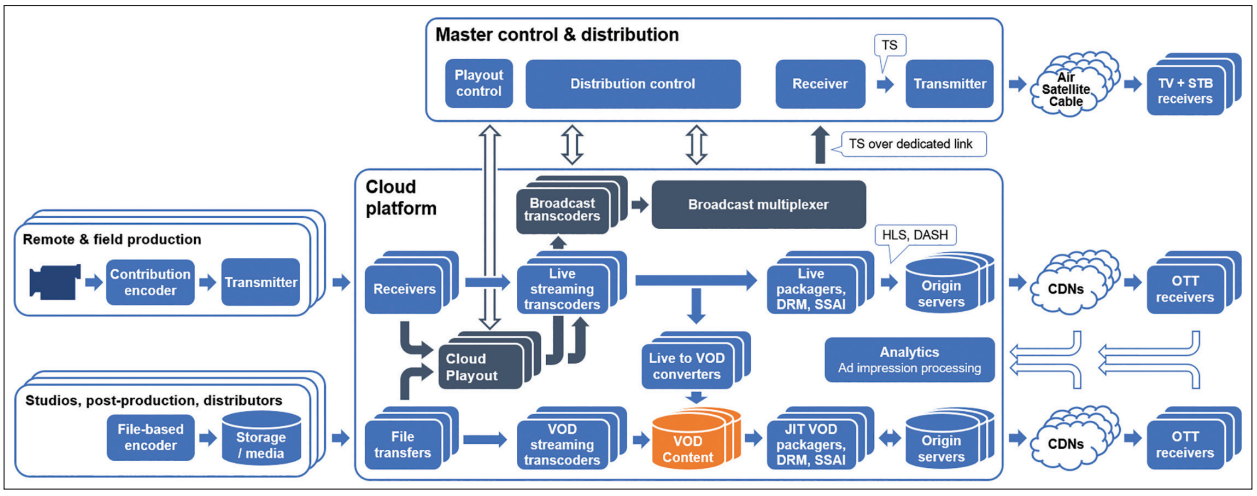

FIGURE 11. Hybrid cloud-based broadcast + OTT system architecture.

- This includes implementation of the statmux capability, generation, and insertion of all related program and system information, possible addition of datacast service capability, and so on.

This architecture is indeed an extreme example, where pretty much all data- and processing-intensive operations are migrated to the cloud. It is most technically challenging to implement, but is also most promising, as it enables the best utilization of the cloud and all the benefits it brings.

Conclusions

In this paper, we studied the differences between on-premise broadcast and cloud-based online video delivery workflows and identified the means needed to bridge the gaps between them. These include improvements in cloud ingest, signal processing stacks, transcoder capabilities, and most importantly, a broadcast-grade cloud playout system. To implement a cloud playout system, we have suggested an architecture employing intra-only mezzanine format and associated processing blocks that can be easily replicated and operated in a fault-tolerant fashion. We finally considered possible evolutions of broadcast and cloud-based video systems and suggested several possible hybrid architectures, with different degrees of offloading of processing in the cloud, which are likely to emerge in the future.

References

1. S. Pizzi and G. Jones, *A Broadcast Engineering Tutorial for Non-Engineers*, 4th ed., National Association Broadcasters (NAB), Focal Press, Taylor & Francis Group: New York and London, 2014, 354p.
2. A. Luther and A. Inglis, *Video Engineering*, 3rd ed., McGraw-Hill: New York, 1999, 549p.
3. D. Wu et al., "Streaming Video Over the Internet: Approaches and Directions," *IEEE Trans. Circuits Syst. Video Technol.*, 11(3):282–300, Mar. 2001.
4. G. J. Conklin et al., "Video Coding for Streaming Media Delivery on the Internet," *IEEE Trans. Circuits Syst. Video Technol.*, 11(3):269–281, Mar. 2001.
5. R. Pantos and W. May, "HTTP Live Streaming, RFC 8216," *IETF*, Aug. 2017. Accessed: Sept. 8, 2021. [Online]. Available: https://tools.ietf.org/html/rfc8216
6. International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) 23009-1:2014, "Information Technology—Dynamic Adaptive Streaming Over HTTP (DASH)—Part 1: Media Presentation Description and Segment Formats," Dec. 2014.
7. Microsoft Smooth Streaming. [Online]. Available: https://www.iis.net/downloads/microsoft/smooth-streaming
8. T. Evens, "Co-Opetition of TV Broadcasters in Online Video Markets: A Winning Strategy?," *Int. J. Digital Television*, 5(1):61–74(14), Mar. 2014.
9. Nielsen Holdings plc, "Total Audience Report," Feb. 2020. [Online]. Available: https://www.nielsen.com/us/en/client-learning/tv/nielsen-total-audience-report-february-2020/. See also [Online]. Available: https://techcrunch.com/2020/02/11/nielsen-streaming-wars-total-audience-report/
10. European Telecommunications Standards Institute (ETSI) TS 102 796, "Hybrid Broadcast Broadband TV," ETSI, Aug. 2016.
11. Advanced Television Systems Committee (ATSC) A/331:2020, "Signaling, Delivery, Synchronization, and Error Protection," *ATSC,* Jan. 2020.
12. T. Stockhammer et al., "Dash in ATSC 3.0: Bridging the Gap Between OTT and Broadcast," *IET Conf. Proc.*, pp. 24–24, 2016.
13. T. Fautier, "Cloud Technology Drives Superior Video Encoding," *SMPTE 2019*, Los Angeles, CA, pp. 1–9, 2019.
14. 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.
15. Amazon Web Services (AWS) Elemental, "Video Processing and Delivery Moves to the Cloud," e-book, 2018. [Online]. Available: https://www.elemental.com/resources/white-papers/e-book-video-processing-delivery-moves-cloud/
16. Sandvine, "The Global Internet Phenomena Report," Sep. 2019. [Online]. Available: https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/Internet%20Phenomena/Internet%20Phenomena%20Report%20Q32019%2020190910.pdf
17. Frost & Sullivan, "Analysis of the Global Online Video Platforms Market," *Frost & Sullivan*, Jun. 2014. [Online]. Available: https://store.frost.com/analysis-of-the-global-online-video-platforms-market.html. See also: [Online]. Available: https://www.eweek.com/small-business/online-video-platform-market-to-top-800-million-by-2019

18. Advanced Television Systems Committee (ATSC) A/65B, "Program and System Information Protocol for Terrestrial Broadcast and Cable (PSIP)," *ATSC*, Mar. 18, 2003.
19. American National Standards Institute/Society of Cable and Telecommunications Engineers (ANSI/SCTE) 35 2007, "Digital Program Insertion Cueing Message for Cable," *SCTE*, 2007.
20. American National Standards Institute/Society of Cable and Telecommunications Engineers (ANSI/SCTE) 67 2010, "Recommended Practice for SCTE 35 Digital Program Insertion Cueing Message for Cable," *SCTE*, 2010.
21. B. J. Lechner et al., "The ATSC Transport Layer, Including Program and System Information (PSIP)," *Proc. IEEE*, 94(1):77–101, Jan. 2006.
22. Y. Reznik et al., "Optimal Design of Encoding Profiles for ABR Streaming," *Proc. Packet Video Workshop*, Amsterdam, The Netherlands, Jun. 12, 2018.
23. Y. Reznik et al., "Optimizing Mass-Scale Multi-Screen Video Delivery," *Proc. 2019 NAB Broadcast Engineering and Information Technology Conference*, Las Vegas, NV, Apr. 6–11, 2019.
24. G. A. Davidson et al., "ATSC Video and Audio Coding," *Proc. IEEE*, 94(1):60–76, Jan. 2006.
25. Advanced Television Systems Committee (ATSC) A/53 Part 1: 2009, "ATSC Digital Television Standard: Part 1—Digital Television System," *ATSC*, 7 Aug. 2009.
26. Advanced Television Systems Committee (ATSC) A/53 Part 4:2009, "ATSC Digital Television Standard: Part 4—MPEG-2 Video System Characteristics," *ATSC*, Aug. 7, 2009.
27. Advanced Television Systems Committee (ATSC) A/54A, "Recommended Practice: Guide to the Use of the ATSC Digital Television Standard," *ATSC*, Dec. 4, 2003.
28. Advanced Television Systems Committee (ATSC) A/72 Part 1:2015, "Video System Characteristics of AVC in the ATSC Digital Television System," *ATSC*, May 19, 2015.
29. Advanced Television Systems Committee (ATSC) A/72 Part 2:2014, "AVC Video Transport Subsystem Characteristics," *ATSC*, Feb. 18, 2014.
30. European Telecommunications Standards Insitute (ETSI) TS 101 154, "Digital Video Broadcasting (DVB): Implementation Guidelines for the Use of MPEG-2 Systems, Video and Audio in Satellite, Cable and Terrestrial Broadcasting Applications," Doc. ETSI TS 101 154 V1.7.1, Annex B, Jun. 2005.
31. American National Standards Institute/Society of Cable and Telecommunications Engineers (ANSI/SCTE) 43 2015, "Digital Video Systems Characteristics Standard for Cable Television," *SCTE*, 2015.
32. American National Standards Institute/Society of Cable and Telecommunications Engineers (ANSI/SCTE) 128 2010-a, "AVC Video Systems and Transport Constraints for Cable Television," *SCTE*, 2010.
33. H. Schulzrinne et al., "RTP: A Transport Protocol for Real-Time Applications," *RFC 1889*, IETF, Jan. 1996.
34. SMPTE, ST 2022-1:2007, "Forward Error Correction for Real-Time Video/Audio Transport Over IP Networks," *ST 2022-1*, SMPTE, May 2007.
35. SMPTE, ST 2022-2:2007, "Unidirectional Transport of Constant Bit Rate MPEG-2 Transport Streams on IP Networks," *ST 2022-2*, SMPTE, May 2007.
36. Zixi, LLC, "Streaming Video Over the Internet and Zixi," May 2015. [Online]. Available: http://www.zixi.com/PDFs/Adaptive-Bit-Rate-Streaming-and-Final.aspx
37. OpenCableSpecifications(OC-SP-)MEZZANINE-C01-161026, "Mezzanine Encoding Specification," Cable Television Laboratories, Inc., Oct. 26, 2016.
38. Adobe Systems, "Real-Time Messaging Protocol (RTMP) Specification. Version 1.0," Dec. 2012. [Online]. Available: https://www.adobe.com/devnet/rtmp.html
39. Dynamic Adaptive Streaming over HTTP-Industry Forum (DASH-IF), "DASH-IF Interoperability Points, v4.3," Nov. 2018. [Online]. Available: https://dashif.org/docs/DASH-IF-IOP-v4.3.pdf
40. ETSI TS 103 285 v1.2.1, "Digital Video Broadcasting (DVB); MPEG-DASH Profile for Transport of ISO BMFF Based DVB Services Over IP Based Networks," Feb. 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/103200_103299/103285/01.03.01_60/ts_103285v010301p.pdf
41. SMPTE, RP 145, "Color Monitor Colorimetry," *SMPTE*, 1987.
42. SMPTE, 170M, "Composite Analog Video Signal—NTSC for Studio Applications," *SMPTE*, 1994.
43. European Broadcasting Union (EBU) Tech. 3213-E, "E.B.U. Standard for Chromaticity Tolerances for Studio Monitors," *EBU*, 1975.
44. International Telecommunication Union – Radiocommunication (ITU-R), Recommendation BT.601, "Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide Screen 16:9 Aspect Ratios," *ITU-R*, Mar. 2011.
45. International Telecommunication Union – Radiocommunication (ITU-R), Recommendation BT.709, "Parameter Values for the HDTV Standards for Production and International Programme Exchange," *ITU-R*, Jun. 2015.
46. International Electrotechnical Commission (IEC) 61966-2-1:1999, "Multimedia Systems and Equipment—Colour Measurement and Management—Part 2-1: Colour Management—Default RGB Colour Space—sRGB," *IEC*, Oct. 1999.
47. Open Cable Specifications (OC-SP)-CEP3.0-I05-151104, "Content Encoding Profiles 3.0 Specification," Cable Television Laboratories, Inc., Nov. 4, 2015.
48. J.Ozer,"EncodingforMultipleDevices,"*StreamingMediaMagazine*, Mar. 2013. [Online]. Available: http://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=88179&fb_comment_id=220580544752826_937649
49. Apple,"HLSAuthoringSpecificationforAppleDevices,"Jun.2019. [Online]. Available: https://developer.apple.com/documentation/http_live_streaming/hls_authoring_specification_for_apple_devices
50. J. Ozer, "Encoding for Multiple-Screen Delivery," *Streaming Media East*, 2013. [Online]. Available: https://www.streamingmediablog.com/wp-content/uploads/2013/07/2013SMEast-Workshop-Encoding.pdf
51. N. Brydon, "Saving Bits—The Impact of MCTF Enhanced Noise Reduction," *SMPTE J.*, 111(1):23–28, Jan. 2002.
52. International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) 13818-2:2013, "Information Technology—Generic Coding of Moving Pictures and Associated Audio Information—Part 2: Video," *ISO/IEC*, Oct. 2013.
53. 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.
54. 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 Eoding," *ISO/IEC*, Dec. 2013.
55. AOM AV1, "AV1 Bitstream & Decoding Process Specification, v1.0.0," *Alliance for Open Media*, Jan. 2019. [Online]. Available: https://aomediacodec.github.io/av1-spec/av1-spec.pdf
56. Consumer Technology Association (CTA) 5001, "Web Application Video Ecosystem—Content Specification," *CTA WAVE*, Apr. 2018. [Online]. Available: https://cdn.cta.tech/cta/media/media/resources/standards/pdfs/cta-5001-final_v2_pdf.pdf
57. M. Perkins and D. Arnstein, "Statistical Multiplexing of Multiple MPEG-2 Video Programs in a Single Channel," *SMPTE J.*, 104(9):596–599, Sept. 1995.

58. L. Boroczky, A. Y. Ngai, and E. F. Westermann, "Statistical Multiplexing Using MPEG-2 Video Encoders," *IBM J. Res. Dev.*, 43(4):511–520, Jul. 1999.
59. Consumer Electronics Association (CEA)-608-B, "Line 21 Data Services," *Consumer Electronics Association*, Apr. 1, 4, 2008.
60. Consumer Electronics Association (CEA)-708-B, "Digital Television (DTV) Closed Captioning," *Consumer Electronics Association*, Aug. 1, 2008.
61. Consumer Electronics Association (CEA) CEB16, "Active Format Description (AFD) & Bar Data Recommended Practice," *CEA*, Jul. 31, 2006.
62. SMPTE, 2016-1, "Standard for Television—Format for Active Format Description and Bar Data," *SMPTE*, 2007.
63. OC-SP-EP-I01-130118, "Encoder Boundary Point Specification," Cable Television Laboratories, Inc., Jan. 2013.
64. FairPlay Streaming. [Online]. Available: https://developer.apple.com/streaming/fps/
65. PlayReady. [Online]. Available: https://www.microsoft.com/playready/overview/
66. Widevine. [Online]. Available: https://www.widevine.com/solutions/widevine-drm
67. 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.
68. 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," Dec. 2018.
69. ID3 Tagging System. [Online]. Available: http://www.id3.org/id3v2.3.0
70. Worldwide Web Consortium (W3C) WebVTT, "The Web Video Text Tracks," W3C, Mar. 2018. [Online]. Available: http://dev.w3.org/html5/webvtt/
71. Worldwide Web Consortium (W3C)TTML1, "Timed Text Markup Language 1," *W3C*, Nov. 2019. [Online]. Available: https://www.w3.org/TR/2018/REC-ttml1-20181108/
72. Worldwide Web Consortium (W3C) IMSC1, "TTML Profiles for Internet Media Subtitles and Captions 1.0," *W3C*, Aug. 2015. [Online]. Available: https://dvcs.w3.org/hg/ttml/raw-file/tip/ttml-ww-pro-33 files/ttml-ww-profiles.html
73. Apple, "Incorporating Ads into a Playlist." [Online]. Available: https://developer.apple.com/documentation/http_live_streaming/example_playlists_for_http_live_streaming/incorporating_ads_into_a_playlist
74. American National Standards Institute/Society of Cable and Telecommunications Engineers (ANSI/SCTE) 30 2017, "Digital Program Insertion Splicing API," *SCTE*, 2017.
75. American National Standards Institute/Society of Cable and Telecommunications Engineers (ANSI/SCTE) 172 2011, "Constraints on AVC Video Coding for Digital Program Insertion," *SCTE*, 2011.
76. SMPTE, 259:2008, "SMPTE Standard—For Television—SDTV—Digital Signal/Data—Serial Digital Interface," *SMPTE*, Jan. 2008.
77. SMPTE, 292-1:2018: "SMPTE Standard—1.5 Gb/s Signal/Data Serial Interface," *SMPTE*, Apr. 2018.
78. SMPTE, 2110-20:2017, "SMPTE Standard—Professional Media Over Managed IP Networks: Uncompressed Active Video," *SMPTE*, Nov. 2017.
79. Haivision, "Secure Reliable Transport (SRT)," Sep. 2019. [Online]. Available: https://github.com/Haivision/srt
80. Video Services Forum (VSF) TR-06-1, "Reliable Internet Stream Transport (RIST) Protocol Specification—Simple Profile," *Video Services Forum*, Oct. 2018. [Online]. Available: http://vsf.tv/download/technical_recommendations/VSF_TR-06-1_2018_10_17.pdf
81. Amazon Web Services (AWS) Direct Connect. [Online]. Available: https://aws.amazon.com/directconnect/
82. Azure ExpressRoute. [Online]. Available: https://azure.microsoft.com/en-us/services/expressroute/
83. FFMPEG filter documentation. [Online]. Available: https://ffmpeg.org/ffmpeg-filters.html
84. R. Vanam and Y. Reznik, "Frame Rate Up-Conversion Using Bi-Directional Optical Flows With Dual Regularization," *Proc. 2020 IEEE Int. Conf. Image Processing (ICIP)*, 2020, pp. 558–562.
85. Teranex Standards Converters. [Online]. Available: https://www.blackmagicdesign.com/products/teranex
86. Grass Valley KudosPro Converters. [Online]. Available: https://www.grassvalley.com/products/kudospro/
87. Apple, "Protocol Extension for Low-Latency HLS (Preliminary Specification)," Apple Inc., Feb. 2020. [Online]. Available: https://developer.apple.com/documentation/http_live_streaming/protocol_extension_for_low-latency_hls_preliminary_specification
88. Dynamic Adaptive Streaming over HTTP-Industry Forum (DASH-IF), "DASH-IFATSC3.0IOP," Jun. 2012. [Online]. Available: https://dashif.org/docs/DASH-IF-IOP-for-ATSC3-0-v1.1.pdf
89. W. Law, "Ultra Low Latency with CMAF." [Online]. Available: https://www.akamai.com/us/en/multimedia/documents/white-paper/low-latency-streaming-cmaf-whitepaper.pdf
90. Brightcove VideoCloud System. [Online]. Available: https://www.brightcove.com/en/online-video-platform.

About the Authors

Yuriy Reznik is a technology fellow and the vice president of research at Brightcove, Inc., Boston, MA. Previously, he held engineering and management positions at InterDigital, San Diego, CA, from 2011 to 2016, Qualcomm, San Diego, 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 also been involved in the work of the ITU-T SG16 and MPEG standards committees and has made contributions to several multimedia coding and delivery standards, including ITU-T H.264/MPEG-4 AVC, MPEG-4 ALS, ITU-T G.718, ITU-T H.265/MPEG HEVC, and MPEG-DASH. Several technologies, standards, and products that he has helped to develop (RealAudio/RealVideo, ITU-T H.264/MPEG-4 AVC, Zencoder, and Brightcove CAE) have been recognized by the NATAS Technology & Engineering Emmy Awards. He holds a PhD degree in computer science from Kyiv University, Kyiv, Ukraine. He is a senior member of IEEE and SPIE and a member of the ACM, AES, and SMPTE. He is a co-author of over 120 conference and journal papers and co-inventor of over 70 granted US patents.

Jordi Cenzano is an engineer specializing in broadcast and online media. He is currently working on the live ingest pipeline optimization at Facebook. He previously worked as a director of engineering at Brightcove, Inc., Boston, MA, and was a principal architect of Brightcove Live

streaming workflow. For over 20 years, he has been working in the media space, starting on the broadcast side, and moving to the online media. He holds an MS degree in information and communication technologies and a BS degree in telecommunications from Universitat Politécnica de Catalunya, Catalunya, Spain, in 2013 and 1997, respectively.

Bo Zhang is a video systems engineer at the Brightcove Research team. He works on researching video delivery and playback technologies and building high-quality, smooth, scalable, and low-latency video streaming software. He has published several research papers in the domains of video streaming and wireless communications. He has a PhD degree from George Mason University, Fairfax, VA, an MS degree from the University of Cincinnati, Cincinnati, OH, and a BS degree from Huazhong University of Science and Technology, Wuhan, China, all in computer science. He was the recipient of the best paper award from ACM MSWiM 2011.

This paper first appeared in the Proceedings of the NAB 2020 Broadcast Engineering and Information Technology (BEIT) Conference. Reprinted here with permission of the author(s) and the National Association of Broadcasters, Washington, DC.