Transitioning Broadcast to Cloud

Yuriy Reznik, Jordi Cenzano, and Bo Zhang
Brightcove, Inc.
Boston, MA, USA

Abstract – We analyze the differences between on-premise broadcast and cloud-based online video delivery workflows and identify technologies needed for bridging the gaps between them. Specifically, we note differences in 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 critically needed technology for convergence. Finally, based on all such considerations, we offer sketches of several possible hybrid architectures, with different degrees of offloading of processing in cloud, that are likely to emerge in the future.

Introduction

Terrestrial broadcast TV has been historically the first and still broadly used technology for 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, etc. – the most practical option currently available is to send streams Over the Top (OTT). The OTT model utilizes IP-connections that many of such devices already have, and *internet streaming* as a delivery mechanism [3-7]. The use of OTT/streaming also makes it possible to implement interactive, non-linear, time-shifted TV, or 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 broader scale, we must also recognize new standards and industry initiatives such as HbbTV [12], as well as ATSC 3.0 [13,14], which are further blurring the boundaries between traditional broadcast and OTT.

In other words, we are now living in an era where *hybrid broadcast + OTT distribution becomes a norm*, and this brings us to a question of *how such hybrid systems can be deployed and operated most efficiently?*

At present time, the two extreme choices are:

- **on-prem:** everything, including playout systems, encoders, multiplexers, servers, and other equipment for both broadcast and OTT distribution is installed and operated on premises, and
- **cloud-based:** almost everything is turned into software-based solutions, and operated using infrastructure of *cloud* service providers, such as AWS, GCP, Azure, etc.

On-prem model is indeed well-known. This is how all traditional broadcast systems have always been built and operated. Cloud-based approach is a more recent development. It requires considerably different (modular, software only) 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 whole design more flexible and future-proof, etc. [16,17].

Furthermore, the *use of cloud has already been proven to be highly scalable, reliable, and cost-effective* for implementing *OTT/streaming delivery.* Today, cloud already powers mass-scale online video services, such as YouTube and Netflix, as well as *online video platforms* (OVPs) – Brightcove, Ooyala,

Kaltura, etc. [9-11]. Besides enabling basic streaming functionality, OVPs also provide means for content management, Dynamic Ad-Insertions (DAI), analytics, client SDKs, and even automatic generators of apps for all major platforms. They basically provide turn-key solutions for OTT services of all sorts.

However, while transition of OTT services to 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, etc. – is a topic that we believe deserves additional discussion.* As we will show in this paper, there are many important differences in ways video processing is currently done in cloud vs 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 Figure 1, and the chain of processing in online video system is shown in Figure 2. Both are indeed conceptual and high-level.

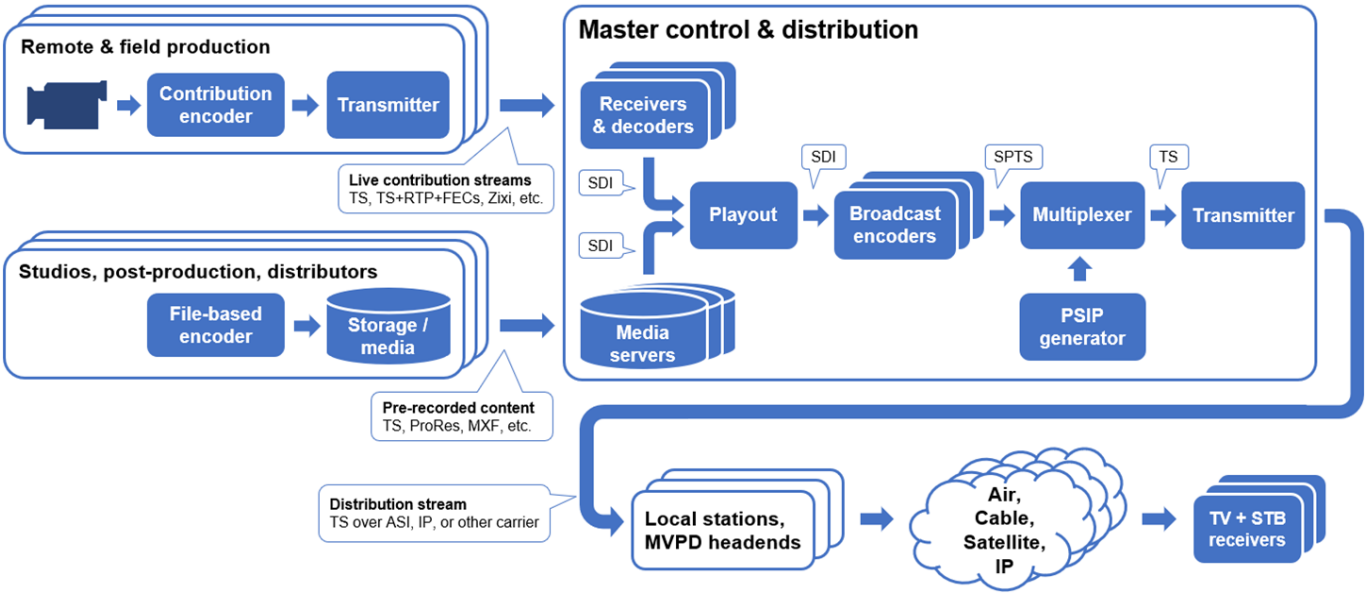

FIGURE 1: CONCEPTUAL DIAGRAM OF PROCESSING OPERATIONS IN BROADCAST DISTRIBUTION.

Main functions and distribution flows

In broadcast, everything is based around processing and delivery of a set of live streams, visible to end users as “TV channels”. As shown in Figure 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”, overlays or “lower thirds”, etc.), slots for ads, captions, metadata, etc.

After playout, all channels are subsequently encoded and passed to a *multiplexer*, which combines them in a *multi-program transport streams* (aka TS or MPEG-2 TS [17]) intended for distribution. In addition to channel’s media content, the final multiplex TS also carries program and system information (PSIP [18]), SCTE 35 cue messages [19,20], and other metadata required for broadcast distribution [21].

As further shown in Figure 1, the distribution chain in broadcast systems may have *multiple tiers* – from main network center to local stations and also MVPDs (multichannel video programming distributors), 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, and then again sent down to distribution or next headend.

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

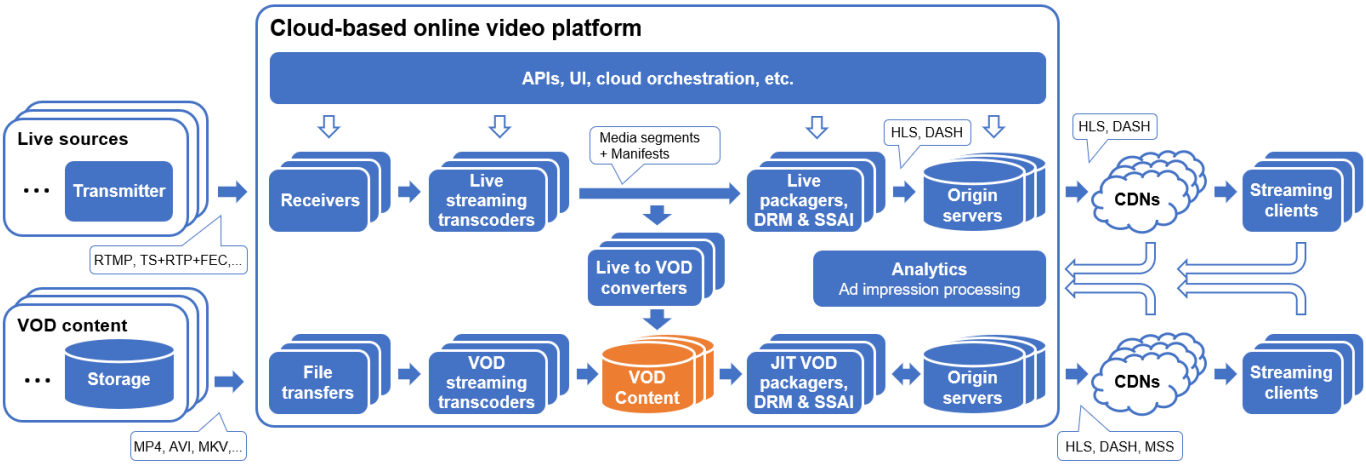

FIGURE 2: CONCEPTUAL DIAGRAM OF PROCESSING OPERATIONS IN CLOUD-BASED ONLINE VIDEO PLATFORM.

In contrast, online video platforms are used primarily for *distribution*. They assume that content is already fully formed. As shown in Figure 2, live inputs are typically turned into live output streams, and pre-recorded media files are typically published as VOD assets. They transcode and repackage inputs into HLS [5], DASH [6], or MSS [7] streaming formats, and then pass them to 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 formation of live streams.

Another important difference between online video systems and broadcast is the *availability of the feedback chain*. In Figure 2 it is shown by contour arrows connecting players and CDNs to the *analytics* module within OVP. 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 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 SD or 1080i 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, real-time UDP-based delivery protocols are normally used. Examples of such protocols include RTP [35], SMPTE 2022-1 [36], SMPTE 2022-2 [37], Zixi [38], etc.

Pre-recorded content usually comes in 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 (see e.g. [34]). Moreover, in most cases, the contribution (or so-called “mezzanine”) encodings are done at rates that are considerably higher than rates used for final distribution. This allows broadcast systems to start with “cleaner” versions of the content.

In case of online video platforms, 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 datacenters, to public Internet over some local ISPs. UDP may or may not be available.

In such a context, the live ingest protocol that become most commonly used, remarkably enough, is RTMP [41]. 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 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 TV systems.

Format characteristic	Broadcast systems	Online platforms / streaming
Temporal sampling	SD: interlaced (bff), telecine HD: progressive, interlaced (tff), telecine	progressive only
Framerates [Hz]	24000/1001, 24, 25, 30000/1001, 30 ,50, 60000/10001, 60	same as source, typically capped at 30Hz or 60Hz
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, 1080p	same as source + down-scaled versions; additional restrictions may apply [45,46]
Chroma sampling	4:2:0, 4:2:2 (contribution, mezzanine feeds)	4:2:0 only
Primary colors	SD (NTSC-derived): SMPTE-C [48,49] SD (PAL, SECAM-derived): EBU [50,51] HD: ITU-R BT.709 [52]	ITU-R BT.709 / sRGB [52, 53]
Color matrix	SD: ITU-R BT.601 [51] HD: ITU-R BT.709 [52]	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 [52]	ITU-R BT.709

TABLE 1: COMPARISON OF VIDEO FORMATS USED IN BROADCAST AND INTERNET STREAMING.

As shown in this table, SD systems almost always use *interlaced*, *bottom-field first (bff)* sampling format [26,30,33]. If the source is progressive (e.g. film) it is typically converted to interlaced form by so-called telecine process [1,2]. HD systems also use interlaced formats, but with *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 [42,44].

In terms of color parameters and SARs, streaming video formats are well aligned with HDTV systems. On the other hand, streaming of SD content requires both color- and SAR-type conversions.

The primary reason why streaming systems are more restrictive is *compatibility* with wide range of possible receiving devices – mobiles, tablets, PCs, etc. [42]. In many such devices, graphics stacks are simply not designed to properly render interlace, or colors other than sRGB / ITU-R BT.709, or pixels that are non-square. This forces color-, temporal sampling type-, and SAR-type conversions.

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

Stream characteristic	Broadcast systems	Online platforms / streaming
Number of outputs	single	multiple (usually 3-10) as needed to support ABR delivery [4,42,43]
Preprocessing	denoising, MCTF-type filters [24,54]	rarely used
Video codecs	MPEG-2 [55], MPEG-4/AVC [56]	MPEG-4/AVC – most deployments HEVC [57], AV1 [58] – special cases
Codec profiles, levels	fixed for each format. applicable standards:	based on target set of devices [42]

	ATSC A/53 P4 [26], ATSC A/72 P1 [29], ETSI TS 101 154 [30], SCTE 128 [33]	guidelines: Apple HLS [44], ETSI TS 103 285 [46], CTA 5001 [47]
GOP length	0.5 sec	2-10 sec
GOP type	closed (most deployments), open	closed
Error resiliency features	mandatory slicing of I / IDR pictures	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 [33]	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	required in some cases (e.g. in film mode)	optional, usually omitted
Buffering period SEI	required in some cases (see [33])	optional, usually omitted
AFD / bar data	required and carried in video ES	not used
Closed captions	required and carried in video ES	optional, maybe carried out-of-band

TABLE 2: COMPARISON OF ENCODED VIDEO STREAMS USED IN BROADCAST AND INTERNET STREAMING.

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 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 so-called *statistical multiplexing* (or *statmux*) regime [59,60], where 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 decoder buffer size and maximum bitrate [44].

Significantly different are also *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 lengths of segments used for delivery.

Broadcast streams also carry more *metadata.* They typically include relevant VBV (video bitstream verifier) [55] or HRD (hypothetical reference decoder) parameters [56], picture structure-, picture timing-, and colorimetry-related information [56]. They also carry *CEA 608/708 closed captions* [61,62] and *AFD (active format descriptor)* / *bar data* information [63,64]. For streaming most may be omitted.

Finally, there are also important differences in pre-processing. Broadcast encoders are famous for the use of denoisers, MCTF-filters, and other pre-processing techniques applied to make compression more efficient [24,54]. 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 [17]. They carry audio and video elementary streams, program and system information [18], SCTE 35 ad markers [19], time signals, content identification, and other metadata as prescribed by relevant broadcast standards and guidelines [21,25,27,30]. TS in cable systems may also carry EBPs [67] 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 time of writing, are:

1) HTTP Live Streaming (HLS) [5],
2) Dynamic Adaptive Streaming over HTTP (DASH) [6], and

3) Microsoft Smooth Streaming (MSS) [7].

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

1) FairPlay [66],
2) PlayReady [67], and
3) Widevine Modular [68].

The support for these technologies varies across different categories of receiving devices. *Hence, in order 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.

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	✓	×	×	×	×	×	✓

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.

HLS, DASH, as well as MSS use *multi-rate, segment-based representation* of media data. Original content is encoded at several different resolutions and bitrates, and then split in segments, each starting at GOP boundary, such that they can be retrieved and decoded separately. Along with media segments (either in TS [17], ISOBMFF [69], or CMAF [70] formats) additional files (usually called *manifests*, *playlists*, or *MPD* files) are provided, describing locations and properties of all such segments. 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, while others may also be embedded in manifests, carried as additional “sidecar” tracks of segment files, or as “event” messages [6], or ID3 tags [71].

For example, in addition to “broadcast-style” carriage of CEA 608/708 [62,63] closed captions in video elementary streams, it is also possible to carry captions as separate tracks of WebVTT [72] or TTML [73] segments, or as IMSC1 timed text data [74] encapsulated in XML or ISOBMFF formats [45]. 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 manifest level in HLS, by either manifest of in-band events in MPEG-DASH, and only in-band in MSS [75,45,7].

To manage such broad diversity of formats, DRMs, and metadata representations, online video platforms are commonly deploying so-called *dynamic* or *just-in-time* (JIT) *packaging* mechanisms [23]. This is illustrated by an architecture shown in Figure 2. Instead of proactively generating and storing all possible permutations of packaged streams on origin server, such 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 capabilities of devices requesting them. This reduces amount of cloud storage needed and also increases the efficiency of use of CDNs when handing multiple content representations [23].

As easily observed, delivery formats and their support system in case of OTT/streaming is completely different as 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 messages) are inserted earlier – at playout or content production stages [20]. A*d splicers* subsequently look for SCTE 35 markers embedded in the TS, and then communicate with *Ad servers* (normally over SCTE 30 [76]) to request and receive ad content that needs to be inserted. Then they update TS streams to insert segments of ad content. Such TS update is a fairly tedious process, involving re-mux, regeneration of timestamps, etc. It also requires both main content and ads to be consistently encoded: have the same exact codec parameters, HDR model, etc. (cf. [77]).

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 (so-called *ad-impressions*) are also registered, collected, and subsequently used for monetization. It is all fully automated and has to work in real-time and at mass scale.

There are two models for ad-insertion that are used in streaming currently: server-side ad-insertion (SSAI) and client-side ad insertion (CSAI) [45]. In case of CSAI, most ad-related processing resides in a client. The cloud only needs to deliver content and SCTE 35 cue messages to the client. This scales well regardless of how cue messages are delivered – both in-band, or in-manifest carriage methods are adequate.

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

For example, in 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 beginning and end of such sections [75]. In case of MPEG DASH, essentially the same functionality is achieved by using multiple periods [45]. The discontinuity markers or changing periods are effectively forcing clients to reset decoders when switching between 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, etc. are normally done with uncompressed video streams, carried over by SDI [78,79], or more recently – by SMPTE 2110 [80] 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 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 can't be made arbitrary small due to CDN efficiency reasons. In practice, 2-, 6-, and 10-second segments are most commonly used [44]. *Same segmented media representations are also commonly used internally in cloud-based processing workflows*. This simplifies exchanges, avoids additional transcoding or transmuxing operations, and reduces many basic stream-level operations to manifest updates. However, such design also makes *random access and delay capabilities in cloud video systems much worse compared to broadcast.*

What also makes things in cloud complicated, is a *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 datacenters, have somewhat different characteristics of hardware, non-synchronized local clocks, etc. The network-induced delays in accessing such instances may also be different. Processing jobs thus have to be scheduled dynamically and in anticipation of all such possible differences. Moreover, occasionally *cloud instances may become unstable, non-responsive, 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 cloud may be achieved, in Figure 3 we show an example of a live streaming system with 2-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 manifests. This system also deploys two *redundancy control* modules. These modules check if 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 non-functional, they update manifest to include a discontinuity marker, and then continue with segments arriving from an alternative path.

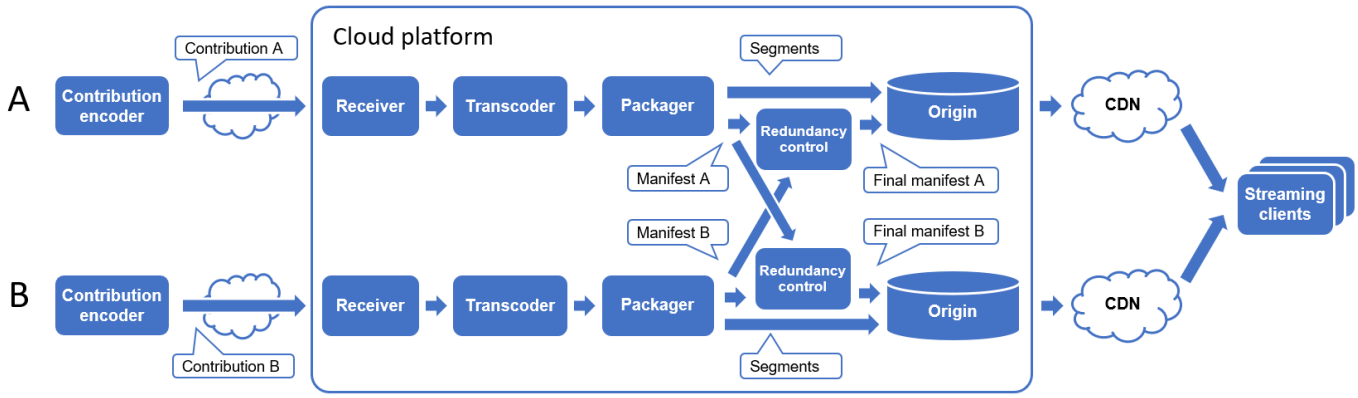

FIGURE 3: AN EXAMPLE OF CLOUD-BASED REDUNDANT LIVE STREAMING WORKFLOW.

As easily observed, this system remains operational in case if either of the chains A or B fails. It also stays operational in case of failure of one of the redundancy control units. However, what is important to note, is that in 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 time of switch/fallback. This comes as a result of 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 complexity and 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 most critical factors that must be considered in planning migration of signal processing functionality in 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 [41] 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 [41]. 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 basically 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 [35], RTP+SMPTE 2022-1 [36], RTP+SMPTE 2022-2 [37], Zixi [38], SRT [39] or RIST [40]. 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 cloud to the broadcast systems.

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

Signal processing

As also mentioned earlier, broadcast workflows may carry videos in progressive, interlaced, or telecine formats. Field orders and pulldown patterns may also differ across different sources. When such signals are then edited or combined together, 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 in and then “naively” tries to convert them to progressive – the results can be disastrous. E.g. a wrong assumption about field order can make videos look jerky, lack of detection of 3:2 pulldowns can produce periodic garbled frames, etc.

What also makes broadcast signals difficult to work with – are accumulations of compression and conversion artifacts. The further down the chain the signal is obtained, the mode “noisier” it becomes.

To work with such complex signals, a proper processing stack is needed. One possible architecture is illustrated in Figure 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.

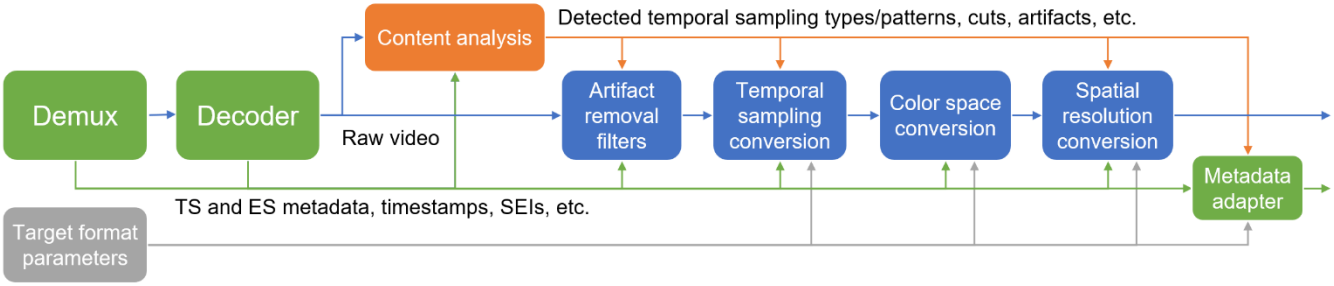

FIGURE 4: DECODING AND FORMAT CONVERSION CHAIN NEEDED TO OPERATE WITH CABLE/BROADCAST CONTENT.

FIGURE 5: EXAMPLE OF MPEG-2 BLOCKING ARTIFACTS (LEFT) AND THEIR REMOVAL BY DEBLOCKING FILTER (RIGHT).

Input video:

Denoised video:

Input after direct transcoding:

Input after denoising and transcoding:

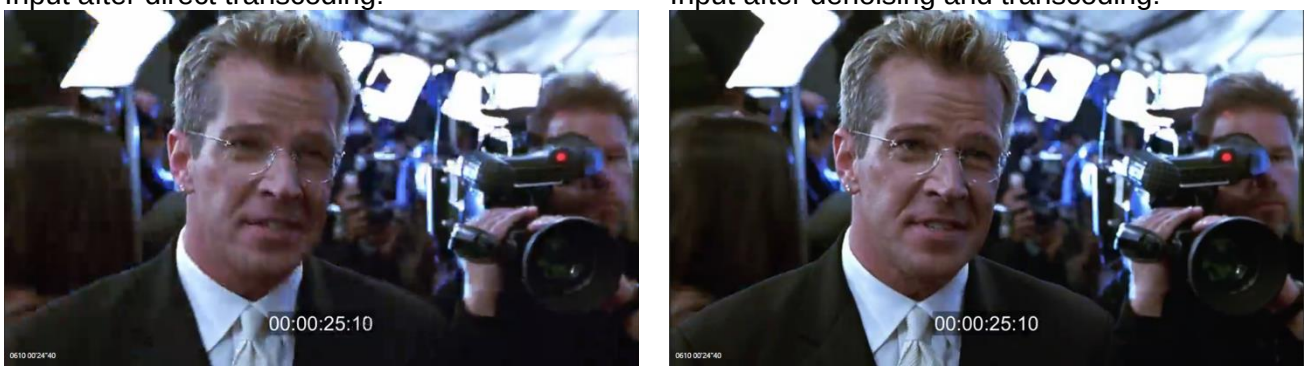

FIGURE 6: EXAMPLE OF USING DENOISING FILTER FOR IMPROVING QUALITY OF FINAL TRANSCODED SIGNAL.

FIGURE 7: COMPARISON OF OUTPUTS OF BASIC (LEFT) AND ADVANCED (RIGHT) DEINTERLACING FILTERS.

The artifact removal filters, such as deblocking and denoising operations are among most basic techniques needed to work with broadcast signals. Deblocking filters are needed, e.g., in working with MPEG-2 encoded content, as MPEG-2 codec [55] does not have in-loop filters, and passes all such artifacts to the output. In Figure 5, we show how such artifacts look, along with 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 signal cleaner, but also makes the job of the subsequent encoder easier, enabling it to achieve better quality or lower rate. We illustrate this effect in Figure 6.

Temporal sampling conversion filter in Figure 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, e.g. telecine segment can be properly converted back to progressive, interlaced – properly deinterlaced, etc.
The quality of temporal sampling conversion operations is very critical. For example, in Figure 7, we show the outputs of a basic deinterlacing filter (FFMPEG “yadif” filter [83]) and 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 in Figure 5, such as color space conversion and scaling filters, 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 their feature sets and tuning / stream constraints. Perhaps the most extreme example of such differences is a statmux regime, where encoders are operating under 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 implementation of statmux, the multiplexer should also be natively implemented in cloud and integrated with encoders.

Cloud playout

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

The design of such a system is a non-trivial 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 can't build a broadcast-grade playout system based on such architecture. Even so-called *ultra-low-delay* versions of HLS, DASH, or CMAF [87-89] are inadequate. For most master control operations, such as previews, non-linear editing, switching, etc., *frame-level access accuracy is an essential requirement.*

In Figure 8, we show 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 format could use any image or video codec operating in Intra-coding mode, along with PCM audio, and index enabling access to each frame. Both input live feeds and pre-recorded content are then converted in such internal mezzanine format and placed on cloud storage. All subsequent operations – such as previews, non-linear 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, addition of logos, transitions, etc. are done by "stream processor" elements.

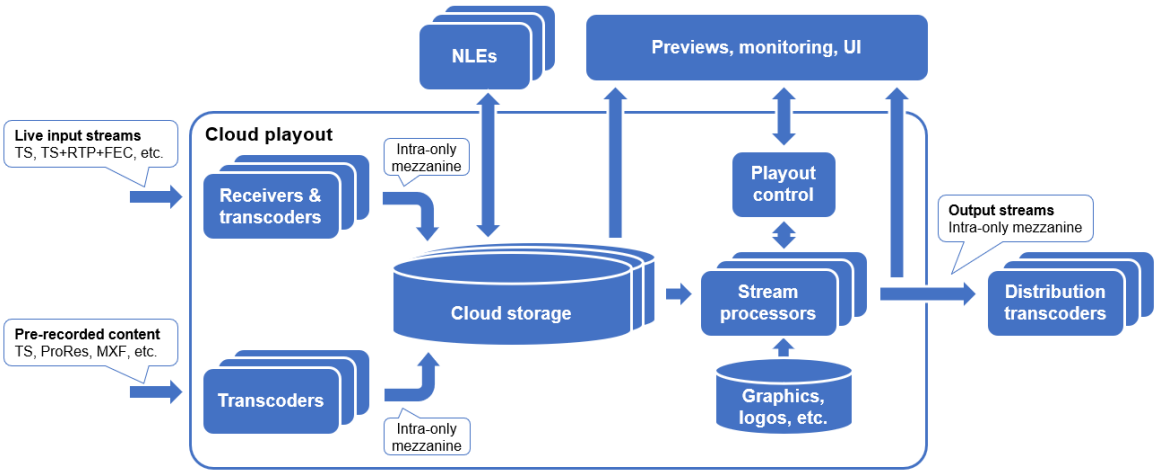

FIGURE 8: EXAMPLE ARCHITECTURE OF CLOUD PLAYOUT SYSTEM.

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

Cloud-based OTT system

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

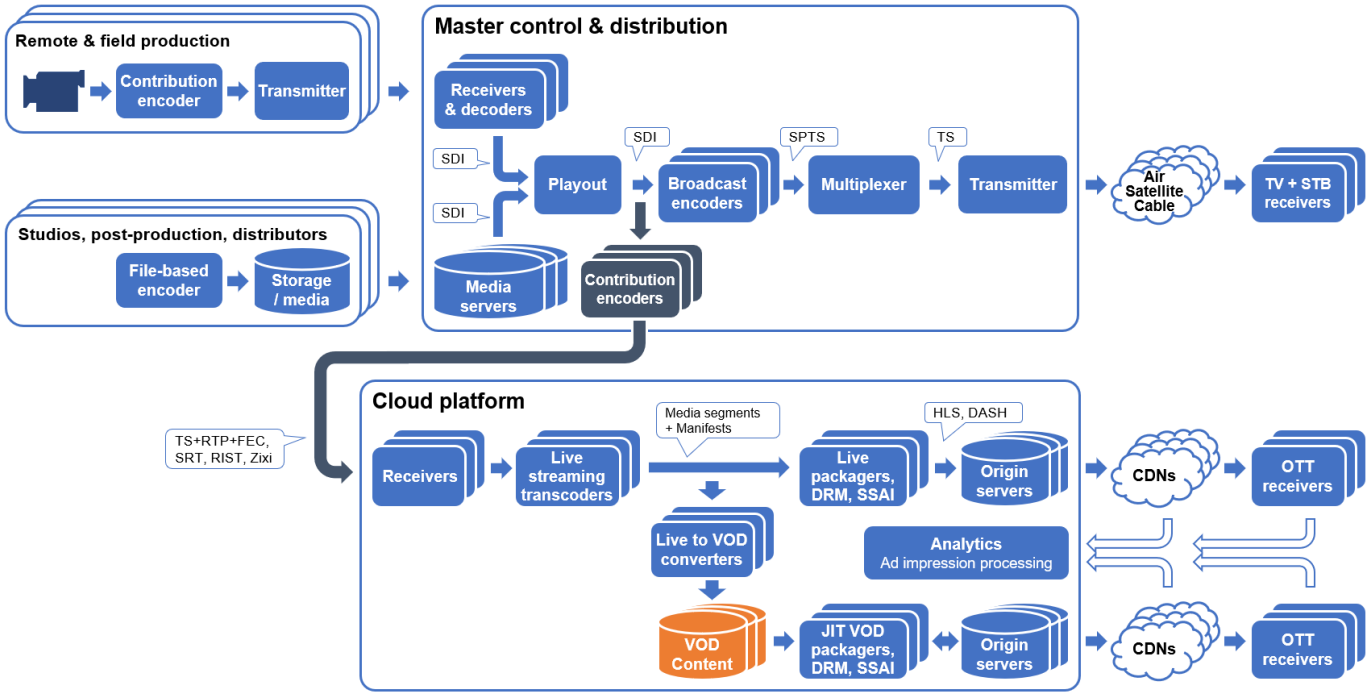

FIGURE 9: HYBRID ARCHITECTURE WITH OTT/STREAMING WORKFLOW OFFLOADED TO CLOUD.

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

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

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

- reliable real-time ingest, e.g. using RTP+FEC, SRT, RIST, or Zixi- type of 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 proven to be effective in practice. Some of the above-mentioned close-gap technologies have also been implemented. For instance, cloud ingest using protocols such as RTP, SMPTE 2022-1, SMPTE 2022-2, or SRT, improvements in SCTE 35 pass-through and use for dynamic ad-insertions, and improvements in the encoding stack were among recent updates in Brightcove VideoCloud system [90].

Cloud-based ingest, playout, and OTT delivery system

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

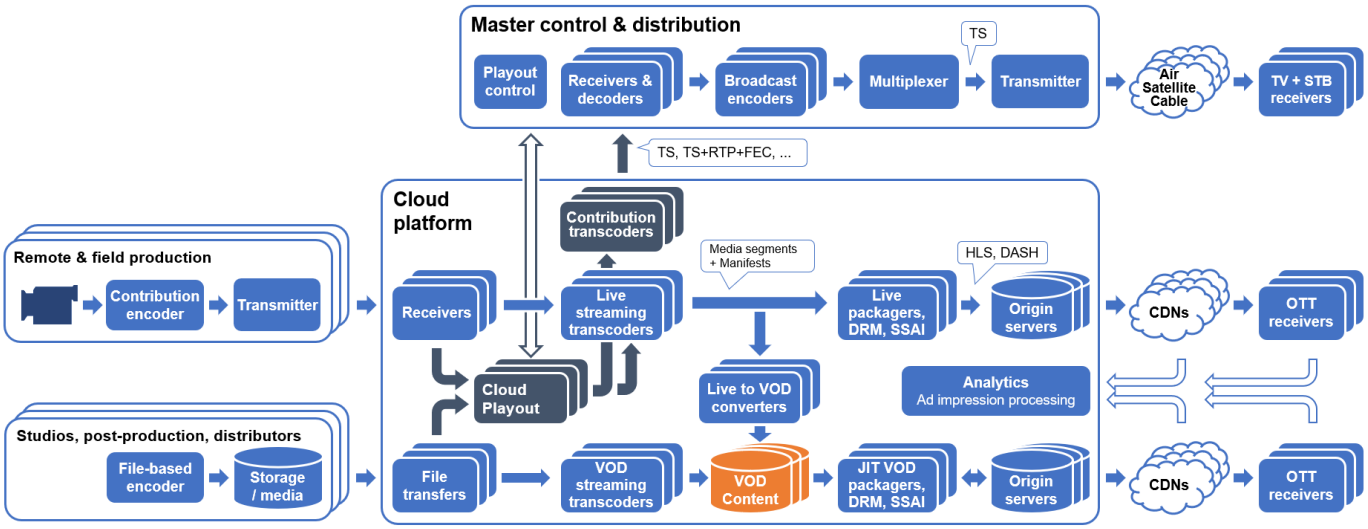

FIGURE 10: HYBRID ARCHITECTURE WITH INGEST, PLAYOUT AND OTT/STREAMING WORKFLOW OFFLOADED TO CLOUD.

As it 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 cloud in this case.

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

In the system depicted in Figure 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 system work, in addition to all improvements mentioned earlier, what further needed is:

- cloud-based broadcast-grade playout system,
- direct link connection to cloud ensuring low latency monitoring and real-time responses in 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, in Figure 11, we show an architecture, where pretty much all signal processing, transcoding, and multiplexing operations are moved to cloud.

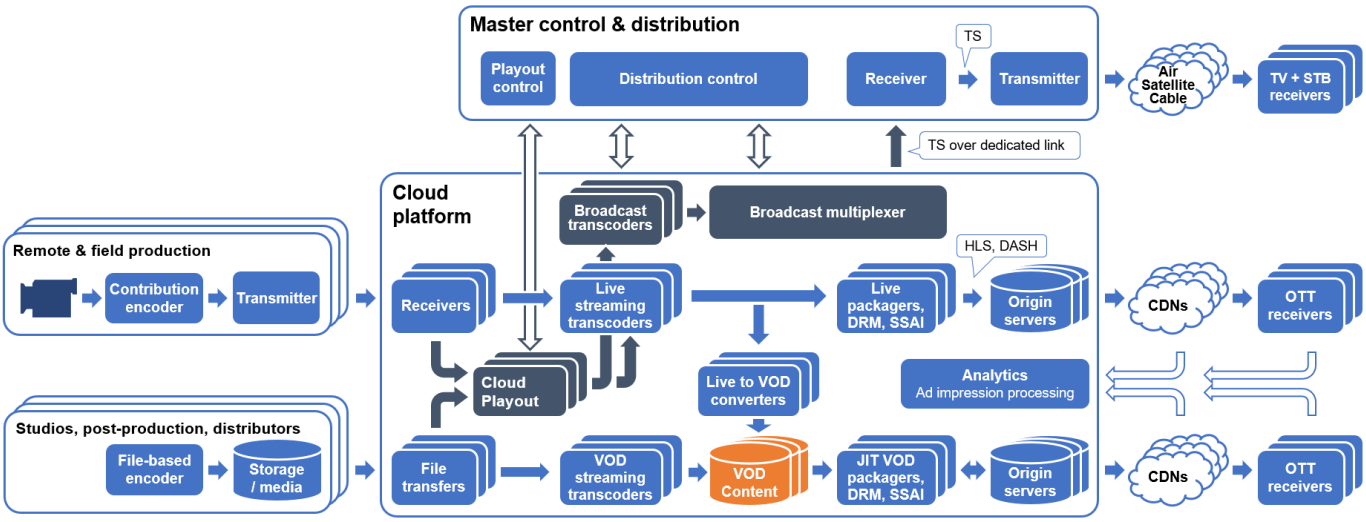

FIGURE 11: HYBRID CLOUD-BASED BROADCAST + OTT SYSTEM ARCHITECTURE.

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

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

- broadcast-grade transcoders and multiplexer should be natively implemented in cloud
- this includes implementation of statmux capability, generation and insertion of all related program and system information, possible addition of datacast service capability, etc.

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

Conclusions

In this paper, we have studied the differences between on-premise broadcast and cloud-based online video delivery workflows and identified means needed for bridging the gaps between them. Such means include: improvements in cloud ingest, signal processing stacks, transcoder capabilities, and most importantly – broadcast-grade cloud playout system. To implement 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 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 cloud, that are likely to emerge in the future.

References

[1] S. Pizzi, G. Jones, *A Broadcast Engineering Tutorial for Non-Engineers, 4th Edition*, National Association Broadcasters (NAB), 2014, 354p.

[2] A. Luther, A. Inglis, *Video Engineering, 3rd Edition*, McGraw-Hill, NY, 1999, 549p.

[3] D. Wu, Y.T. Hou, W. Zhu, Y-Q. Zhang, and J.M. Peha, “Streaming video over the internet: approaches and directions,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 282–300, 2001.

[4] 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 Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 269–281, 2001.

[5] R. Pantos and W. May, “HTTP live streaming, RFC 8216,” *IETF*, https://tools.ietf.org/html/rfc8216, August 2017.

[6] ISO/IEC 23009-1:2020, “Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats”, 4th edition.

[7] Microsoft Smooth Streaming, https://www.iis.net/downloads/microsoft/smooth-streaming

[8] T. Evens, “Co-opetition of TV broadcasters in online video markets: A winning strategy?”, *International Journal of Digital Television*, vol. 5, no 1, 2014, pp. 61-74(14).

[9] Nielsen Holdings plc, “Total Audience Report,” February 2020 https://www.nielsen.com/us/en/client-learning/tv/nielsen-total-audience-report-february-2020/. See also https://techcrunch.com/2020/02/11/nielsen-streaming-wars-total-audience-report/

[10] Sandvine, “The Global Internet Phenomena Report,” September 2019 https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/Internet%20Phenomena/Internet%20Phenomena%20Report%20Q32019%2020190910.pdf

[11] Frost & Sullivan, “Analysis of the Global Online Video Platforms Market”, *Frost & Sullivan*, June 2014. https://store.frost.com/analysis-of-the-global-online-video-platforms-market.html. See also: https://www.eweek.com/small-business/online-video-platform-market-to-top-800-million-by-2019

[12] ETSI TS 102 796, “Hybrid Broadcast Broadband TV,” *ETSI*, August 2016.

[13] ATSC A/331:2020, “Signaling, Delivery, Synchronization, and Error Protection,” *ATSC*, January 2020.

[14] T. Stockhammer, I. Sodagar, W. Zia, S. Deshpande, S. Oh, M-L. Champel, “Dash in ATSC 3.0: bridging the gap between OTT and broadcast,” IET Conference Proceedings, 2016, p. 24-24.

[15] AWS Elemental, “Video Processing and Delivery Moves to the Cloud,” e-book, 2018. https://www.elemental.com/resources/white-papers/e-book-video-processing-delivery-moves-cloud/

[16] T. Fautier, "Cloud Technology Drives Superior Video Encoding," SMPTE 2019, Los Angeles, California, 2019, pp. 1-9.

[17] ISO/IEC 13818-1:2019, “Information technology — Generic coding of moving pictures and associated audio information: Systems — Part 1: Systems”, *ISO/IEC,* June 2019.

[18] ATSC A/65B, “Program and System Information Protocol for Terrestrial Broadcast and Cable (PSIP),” *ATSC*, Mar. 18, 2003.

[19] ANSI/SCTE 35 2007, “Digital Program Insertion Cueing Message for Cable,” *SCTE*, 2007.

[20] ANSI/SCTE 67 2010, “Recommended Practice for SCTE 35 Digital Program Insertion Cueing Message for Cable,” *SCTE*, 2010.

[21] B. J. Lechner, R. Chernock, M. Eyer, A. Goldberg, and M. Goldman, “The ATSC transport layer, including Program and System Information (PSIP),” *Proc. IEEE,* vol. 94, no. 1, pp. 77–101, Jan. 2006

[22] Y. Reznik, K. Lillevold, A. Jagannath, J. Greer, and J. Corley, "Optimal design of encoding profiles for ABR streaming," Proc. Packet Video Workshop, Amsterdam, The Netherlands, June 12, 2018

[23] Y. Reznik, X. Li, K. Lillevold, R. Peck, T. Shutt, R. Marinov, "Optimizing Mass-Scale Multi-Screen Video Delivery," *Proc. 2019 NAB Broadcast Engineering and Information Technology Conference*, Las Vegas, NV, April 6-11, 2019.

[24] G. A. Davidson, M. A. Isnardi, L. D. Fielder, M. S. Goldman and C. C. Todd, "ATSC Video and Audio Coding," *Proceedings of the IEEE*, vol. 94, no. 1, pp. 60-76, Jan. 2006.

[25] ATSC A/53 Part 1: 2009, "ATSC Digital Television Standard: Part 1 – Digital Television System," *ATSC*, 7 August 2009.

[26] ATSC A/53 Part 4:2009, "ATSC Digital Television Standard: Part 4 – MPEG-2 Video System Characteristics," *ATSC*, 7 August 2009

[27] ATSC A/54A, "Recommended practice: Guide to the use of the ATSC digital television standard," *ATSC*, Dec. 4, 2003.

[28] ATSC A/72 Part 1:2015, "Video System Characteristics of AVC in the ATSC Digital Television System," *ATSC*, 19 May 2015.

[29] ATSC A/72 Part 2:2014, "AVC Video Transport Subsystem Characteristics," *ATSC*, 18 February 2014.

[30] 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, June 2005.

[31] ANSI/SCTE 43 2015, "Digital Video Systems Characteristics Standard for Cable Television," *SCTE*, 2015.

[32] ANSI/SCTE 128 2010-a, "AVC Video Systems and Transport Constraints for Cable Television," SCTE 2010.

[33] OC-SP-CEP3.0-I05-151104, "Content Encoding Profiles 3.0 Specification," Cable Television Laboratories, Inc., November 4, 2015.

[34] OC-SP-MEZZANINE-C01-161026, "Mezzanine Encoding Specification," Cable Television Laboratories, Inc., October 26, 2016.

[35] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," *RFC 1889*, IETF, Jan. 1996

[36] SMPTE ST 2022-1:2007, "Forward Error Correction for Real-Time Video/Audio Transport Over IP Networks," *ST 2022-1*, SMPTE, May 2007.

[37] SMPTE ST 2022-2:2007, "Unidirectional Transport of Constant Bit Rate MPEG-2 Transport Streams on IP Networks," *ST 2022-2*, SMPTE, May 2007.

[38] Zixi, LLC, "Streaming video over the internet and Zixi", May 2015 http://www.zixi.com/PDFs/Adaptive-Bit-Rate-Streaming-and-Final.aspx

[39] Haivision, "Secure Reliable Transport (SRT)", Sept 2019, https://github.com/Haivision/srt

[40] VSF TR-06-1, “Reliable Internet Stream Transport (RIST) Protocol Specification – Simple Profile”, *Video Services Forum*, Oct. 2018. http://vsf.tv/download/technical_recommendations/VSF_TR-06-1_2018_10_17.pdf

[41] Adobe Systems, “Real-Time Messaging Protocol (RTMP) specification. Version 1.0,” Dec. 2012. https://www.adobe.com/devnet/rtmp.html

[42] J. Ozer, “Encoding for multiple devices,” *Streaming Media Magazine,* March 2013. http://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=88179&fb_comment_id=220580544752826_937649

[43] J. Ozer, “Encoding for multiple-screen delivery”, *Streaming Media East,* 2013. https://www.streamingmediablog.com/wp-content/uploads/2013/07/2013SMEast-Workshop-Encoding.pdf

[44] Apple, “HLS Authoring Specification for Apple Devices,” June 2019. https://developer.apple.com/documentation/http_live_streaming/hls_authoring_specification_for_apple_devices

[45] DASH-IF, “DASH-IF Interoperability Points, v4.3”, Nov. 2018. https://dashif.org/docs/DASH-IF-IOP-v4.3.pdf

[46] 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”, February 2020. https://www.etsi.org/deliver/etsi_ts/103200_103299/103285/01.03.01_60/ts_103285v010301p.pdf

[47] CTA 5001, “Web Application Video Ecosystem – Content Specification”, *CTA WAVE*, April 2018. https://cdn.cta.tech/cta/media/media/resources/standards/pdfs/cta-5001-final_v2_pdf.pdf

[48] SMPTE RP 145, “Color Monitor Colorimetry,” *SMPTE*, 1987.

[49] SMPTE 170M, "Composite Analog Video Signal – NTSC for Studio Applications," *SMPTE*, 1994.

[50] EBU Tech. 3213-E, “E.B.U. Standard for Chromaticity Tolerances for Studio Monitors,” *EBU*, 1975.

[51] ITU-R Recommendation BT.601, “Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios,” *ITU-R*, March 2011.

[52] ITU-R Recommendation BT.709, “Parameter values for the HDTV standards for production and international programme exchange,” *ITU-R*, June, 2015.

[53] IEC 61966-2-1:1999, “Multimedia systems and equipment - Colour measurement and management - Part 2-1: Colour management - Default RGB colour space – sRGB”, *IEC*, October 1999.

[54] N. Brydon, "Saving Bits—The Impact of MCTF Enhanced Noise Reduction," *SMPTE Journal*, vol. 111, no. 1, pp. 23-28, Jan. 2002.

[55] ISO/IEC 13818-2:2013, “Information technology — Generic coding of moving pictures and associated audio information — Part 2: Video”, *ISO/IEC,* Oct. 2013.

[56] ISO/IEC 14496-10:2003, “Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding,” *ISO/IEC,* December 2003.

[57] ISO/IEC 23008-2:2013, “Information technology –High efficiency coding and media delivery in heterogeneous environments – Part 2: High efficiency video coding,” *ISO/IEC,* December 2013.

[58] AOM AV1, “AV1 Bitstream & Decoding Process Specification, v1.0.0”, *Alliance for Open Media*, Jan 2019, https://aomediacodec.github.io/av1-spec/av1-spec.pdf

[59] M. Perkins and D. Arnstein, “Statistical multiplexing of multiple MPEG-2 video programs in a single channel,” SMPTE Journal, pp. 596–599, Sept. 1995.

[60] L. Boroczky, A. Y. Ngai and E. F. Westermann, "Statistical multiplexing using MPEG-2 video encoders," *IBM Journal of Research and Development*, vol. 43, no. 4, pp. 511-520, July 1999.

[61] CEA-608-B, “Line 21 data services,” *Consumer Electronics Association*, April 1, 04, 2008.

[62] CEA-708-B, “Digital television (DTV) closed captioning,” *Consumer Electronics Association*, August 01, 2008.

[63] CEA CEB16, “Active Format Description (AFD) & Bar Data Recommended Practice,” *CEA*, 31 July 2006.

[64] SMPTE 2016-1, “Standard for Television – Format for Active Format Description and Bar Data,” *SMPTE*, 2007.

[65] OC-SP-EP-I01-130118, “Encoder Boundary Point Specification,” Cable Television Laboratories, Inc., January 2013.

[66] FairPlay Streaming, https://developer.apple.com/streaming/fps/

[67] PlayReady, https://www.microsoft.com/playready/overview/

[68] Widevine, https://www.widevine.com/solutions/widevine-drm

[69] ISO/IEC 14496-12:2015, “Information technology -- Coding of audio-visual objects -- Part 12: ISO base media file format”, 2015.

[70] ISO/IEC 23000-19:2018, “Information technology - Coding of audio-visual objects - Part 19: Common media application format (CMAF) for segmented media”. Dec 2018.

[71] ID3 Tagging System, http://www.id3.org/id3v2.3.0

[72] W3C WebVTT, “The Web Video Text Tracks”, W3C, March 2018, http://dev.w3.org/html5/webvtt/

[73] W3C TTML1, “Timed Text Markup Language 1”, *W3C*, Nov, 2019, https://www.w3.org/TR/2018/REC-ttml1-20181108/

[74] W3C IMSC1, “TTML Profiles for Internet Media Subtitles and Captions 1.0”, *W3C*, August 2015, https://dvcs.w3.org/hg/ttml/raw-file/tip/ttml-ww-pro-33 files/ttml-ww-profiles.html

[75] Apple, “Incorporating Ads into a Playlist,” https://developer.apple.com/documentation/http_live_streaming/example_playlists_for_http_live_streaming/incorporating_ads_into_a_playlist

[76] ANSI/SCTE 30 2017, “Digital Program Insertion Splicing API,” *SCTE,* 2017.

[77] ANSI/SCTE 172 2011, “Constraints on AVC Video Coding for Digital Program Insertion,” SCTE 2011.

[78] SMPTE 259:2008, “SMPTE Standard - For Television — SDTV - Digital Signal/Data — Serial Digital Interface”, *SMPTE*, Jan 2008.

[79] SMPTE 292-1:2018: “SMPTE Standard - 1.5 Gb/s Signal/Data Serial Interface”, *SMPTE*, April 2018.

[80] SMPTE 2110-20:2017, “SMPTE Standard - Professional Media Over Managed IP Networks: Uncompressed Active Video”, *SMPTE*, Nov. 2017.

[81] AWS Direct Connect, https://aws.amazon.com/directconnect/

[82] Azure ExpressRoute, https://azure.microsoft.com/en-us/services/expressroute/

[83] FFMPEG filter documentation, https://ffmpeg.org/ffmpeg-filters.html

[84] R. Vanam, Y. Reznik, “Temporal Sampling Conversion using Bi-directional Optical Flows with Dual Regularization”, in Proc. IEEE International Conference on Image Processing (ICIP), October 2020 – submitted.

[85] Teranex standards converters, https://www.blackmagicdesign.com/products/teranex

[86] Grass Valley KudosPro converters, https://www.grassvalley.com/products/kudospro/

[87] Apple, “Protocol Extension for Low-Latency HLS (Preliminary Specification)”, Apple Inc, February 2020, https://developer.apple.com/documentation/http_live_streaming/protocol_extension_for_low-latency_hls_preliminary_specification

[88] DASH-IF, “DASH-IF ATSC3.0 IOP”, June 2012, https://dashif.org/docs/DASH-IF-IOP-for-ATSC3-0-v1.1.pdf

[89] W. Law, “Ultra Low latency with CMAF,” https://www.akamai.com/us/en/multimedia/documents/white-paper/low-latency-streaming-cmaf-whitepaper.pdf

[90] Brightcove VideoCloud system, https://www.brightcove.com/en/online-video-platform