Optimizing Mass-Scale Multi-Screen Video Delivery

Yuriy Reznik, Xiangbo Li, Karl Lillevold, Robert Peck, Thom Shutt, and Radoslav Marinov
Brightcove, Inc.
Seattle, WA
{yreznik, xli, klillevold, rpeck, tshutt, rmarinov}@brightcove.com

Abstract - *We propose a combination of tools and techniques for improving OTT delivery to devices with different decoding, rendering, and connection capabilities. These tools and techniques include: dynamic packaging, dynamic devices detection, dynamic manifest generation, rules engine, analytics engine, and context aware encoding (CAE). We explain how such tools and techniques can be implemented using current cloud computing and CDN platforms and tuned to achieve optimal end-to-end performance, delivering best possible user experience and minimizing transcoding-, bandwidth- and storage costs.*

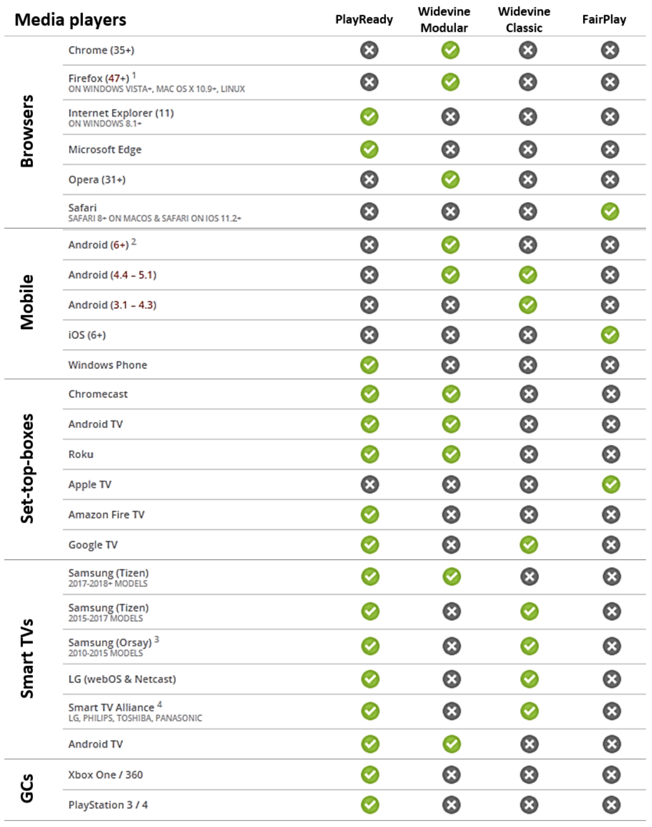
INTRODUCTION

Over the course of the last 2 decades OTT streaming has evolved from a pioneering concept to a mainstream technology used to deliver media content. Important steps in this evolution included:

- the invention of the concept of adaptive bitrate (ABR) streaming [1,2],
- the emergence of CDNs and HTTP-based delivery model are most practical method of mass-scale delivery,
- the emergence of suitable standard codecs, file formats and system specifications (e.g. H.264 [3], HEVC [4], HLS [5], DASH [6], and most recently CMAF [7],
- consolidation of DRMs to fewer recognized systems (e.g. FairPlay, Widevine, PlayReady) that are broadly supported, and
- improvements in client technologies, such as MSE [8] and EME [9] functions supported by most popular web browsers.

However, despite all these improvements and consolidations to fewer choices of codecs, formats, and DRMs, modern-days OTT media delivery systems still face fragmentation in ways different client devices support them.

For example, most existing devices can decode H.264, including streams encoding using H.264 Main and High profiles. Some newer devices, most notably Apple devices with iOS 11 or later, can also decode HEVC. However, many older devices, including Android devices with versions prior to 6.0, most likely can only play H.264 baseline. Likewise, it is not a secret that to send streams to Apple devices, one has to use HLS, while DASH is preferred for Androids and SmartTVs. Moreover, some older TVs and game consoles can only play Smooth streaming, and some other legacy devices that can only play progressive download streams. The support of different types of DRMs across different devices is also fragmented, as illustrated in Figure 1.



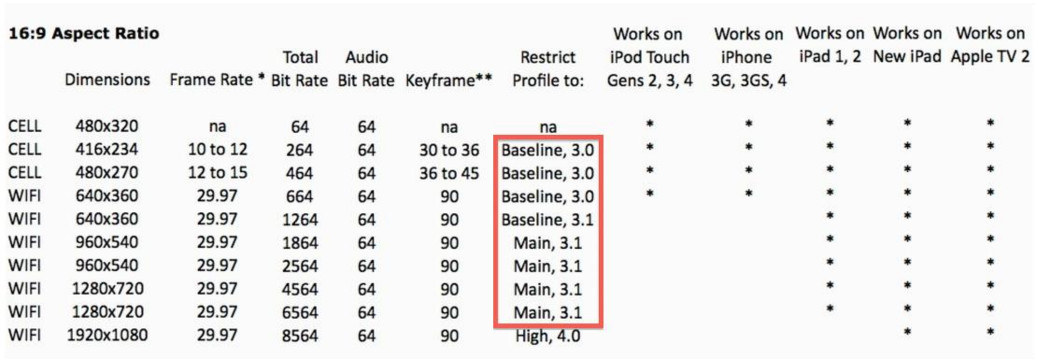
Media players		PlayReady	Widevine Modular	Widevine Classic	FairPlay
Browsers	Chrome (35+)	✗	✓	✗	✗
	Firefox (47+) [1] ON WINDOWS VISTA+, MAC OS X 10.9+, LINUX	✗	✓	✗	✗
	Internet Explorer (11) ON WINDOWS 8.1+	✓	✗	✗	✗
	Microsoft Edge	✓	✗	✗	✗
	Opera (31+)	✗	✓	✗	✗
	Safari SAFARI 8+ ON MACOS & SAFARI ON IOS 11.2+	✗	✗	✗	✓
Mobile	Android (6+) [2]	✗	✓	✗	✗
	Android (4.4 – 5.1)	✗	✓	✓	✗
	Android (3.1 – 4.3)	✗	✗	✓	✗
	iOS (6+)	✗	✗	✗	✓
	Windows Phone	✓	✗	✗	✗
Set-top-boxes	Chromecast	✓	✓	✗	✗
	Android TV	✓	✓	✗	✗
	Roku	✓	✓	✗	✗
	Apple TV	✗	✗	✗	✓
	Amazon Fire TV	✓	✗	✗	✗
	Google TV	✓	✗	✓	✗
Smart TVs	Samsung (Tizen) 2017-2018+ MODELS	✓	✓	✗	✗
	Samsung (Tizen) 2015-2017 MODELS	✓	✗	✓	✗
	Samsung (Orsay) [3] 2010-2015 MODELS	✓	✗	✓	✗
	LG (webOS & Netcast)	✓	✗	✓	✗
	Smart TV Alliance [4] LG, PHILIPS, TOSHIBA, PANASONIC	✓	✗	✓	✗
	Android TV	✓	✓	✗	✗
GCs	Xbox One / 360	✓	✗	✗	✗
	PlayStation 3 / 4	✓	✗	✗	✗

FIGURE. 1: SUPPORT OF DRMS ACROSS DIFFERENT DEVICES.

Known methods of streaming delivery considering fragmentation of existing codecs and formats include:

- creation of *separate copies of streams*, packaged specifically to different delivery formats (HLS, DASH, Smooth, etc.) and DRMs (PlayReady, FairPlay, Widevine, etc.),
- *dynamic transmuxing* and *dynamic encryption* of streams encoded and stored in some intermediate format to match requirement of final delivery formats and DRMs,
- creation of *separate encoding profiles* (and ABR stacks of streams) using each codec (e.g. H.264, HEVC, H.264/baseline), and/or
- creation of *mixed encoding profiles*, where low-bitrate streams are targeted to legacy devices and hence encoded using H.264 baseline, while higher bitrate streams are encoded using H.264 main and High profiles (see Figure 2).

Naturally, creating many versions of encoded streams for different codecs, protocols and DRMs dramatically increases transcoding, storage and CDN costs. It also affects efficiency



16:9 Aspect Ratio

	Dimensions	Frame Rate *	Total Bit Rate	Audio Bit Rate	Keyframe**	Restrict Profile to:	Works on iPod Touch Gens 2, 3, 4	Works on iPhone 3G, 3GS, 4	Works on iPad 1, 2	Works on New iPad	Works on Apple TV 2
CELL	480x320	na	64	64	na	na	*	*	*	*	*
CELL	416x234	10 to 12	264	64	30 to 36	Baseline, 3.0	*	*	*	*	*
CELL	480x270	12 to 15	464	64	36 to 45	Baseline, 3.0	*	*	*	*	*
WIFI	640x360	29.97	664	64	90	Baseline, 3.0	*	*	*	*	*
WIFI	640x360	29.97	1264	64	90	Baseline, 3.1			*	*	*
WIFI	960x540	29.97	1864	64	90	Main, 3.1			*	*	*
WIFI	960x540	29.97	2564	64	90	Main, 3.1			*	*	*
WIFI	1280x720	29.97	4564	64	90	Main, 3.1			*	*	*
WIFI	1280x720	29.97	6564	64	90	Main, 3.1			*	*	*
WIFI	1920x1080	29.97	8564	64	90	High, 4.0				*	*

FIGURE. 2: TYPICAL ENCODING PROFILES USED FOR HLS STREAMING (SOURCE APPLE TECH NOTE TN2224,2004).

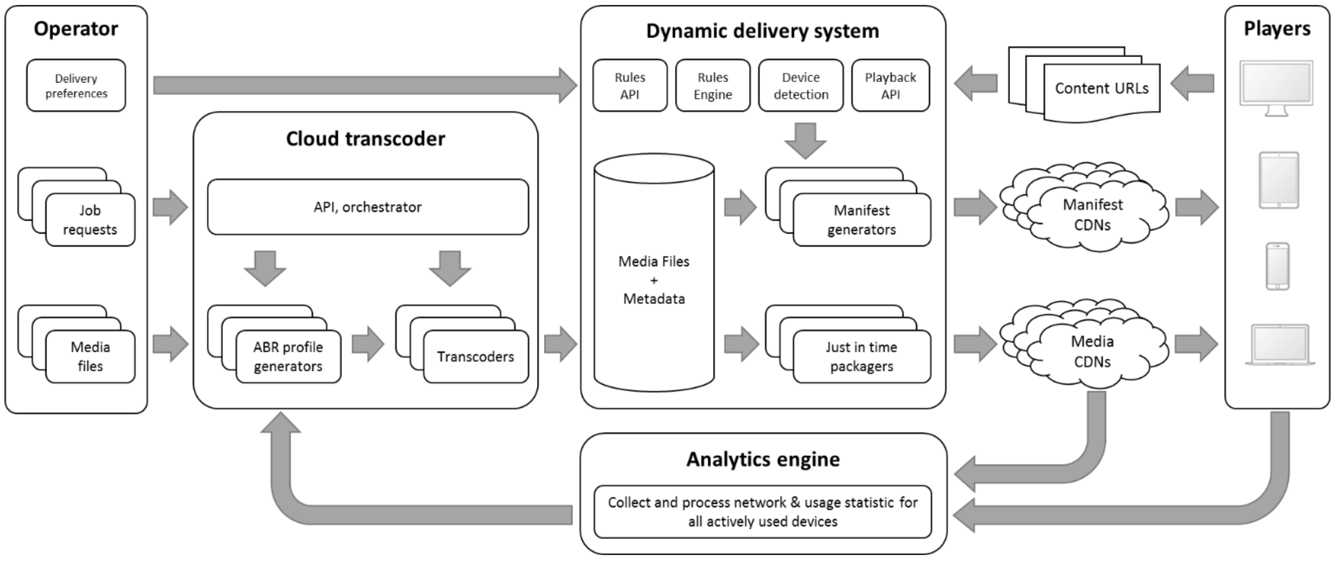


FIGURE 3: HIGH-LEVEL ARCHITECTURE OF CLOUD BASED VIDEO DELIVERY SYSTEM.

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

Similarly, the use of separate encoding profiles for different codecs or codec profiles/level combinations sounds wasteful. It may create more streams than are actually needed to enable delivery. The use of mixed encoding profiles is a better idea, but it is complicated by the fact that some players may not be able to switch between different codecs (e.g. H.264 and HEVC), and that depending on the operator or the region, the distribution of video playback devices and their codec support capabilities may be very different. Hence ideally, encoding profiles should be generated customarily, accounting for the context of each operator or region.

And finally, what also makes video encoding and delivery challenging, is that video content by itself is highly variable, rendering static (pre-configured) ABR encoding profiles suboptimal for some video sequences or over time. To address this issue, in recent years several techniques have been proposed based on the concept of dynamic generation of encoding profiles for each content item. These includes so-called "per-title" [14], "content-aware" [15], or "context-aware" [12] encoding techniques. However, in most cases, such techniques have been developed to work only with a single codec (e.g. H.264 or HEVC). The use of multiple codecs and specifics of fragmentation of their support have not yet been incorporated in problem definition addressed by these techniques.

Summarizing all of the above, we note that while many effective techniques and standards for enabling mass-scale multi-screen video delivery have already been developed, there are still many areas where additional work can be done, and where additional improvements could possibly be obtained. Such areas include improvements in overall delivery system architecture design, coupling and joint optimization of different modules (e.g. device detection, transmuxing, dynamic manifest generation, encoding profile generation, etc.), and end-to-end system performance optimizations.

The objective of this paper is to offer some results on the above-mentioned topics. In the next section, we will describe our proposed multi-screen delivery system architecture, highlighting commonly known and some unique elements, and reasoning behind them. We then focus on explaining tools developed for end to end optimizations. We next present examples of system statistics and explain performance gains that have been achieved. The last section offers concluding remarks.

Architecture of Cloud-Based Multi-Screen Video Delivery System

I. System overview

In Figure 3, we present a high-level architecture of the proposed cloud-based video delivery system. It consists of several functional blocks, and where all exchanges, as common for cloud-based systems, are done by means of RESTful APIs.

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

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

The *dynamic delivery system* is essentially a layer performing selective *transmuxing*, *encryption*, and passage of final streams to CDNs for the purpose of delivery. It is also responsible for *manifest generation*. This module is implemented as a highly distributed system, allowing such operations to be performed sufficiently close to the players.

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

In next sections we describe operations of elements of this system during video delivery process.

II. Playback initiation

When *dynamic delivery* system receives a playback request for a particular media content, it generates a list of manifest URLs, representing all possible combinations of supported delivery protocols, formats, and DRMs. This list of URLs is subsequently presented to a player. If player recognizes any of the formats in this list, it then tries to load the corresponding manifest based on URL provided. Such manifest, in turn, may or may not be present in the manifest CDN. If it is absent, which happens the first time a content in some particular format is requested, this results in CDN cache miss, bringing control back to dynamic delivery system, and its *device detection* and *manifest generation* modules.

III. Device Detection

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

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

Table. 1: Properties that device detection seeks to establish.

IV. Manifest generation and rules engine

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

The other function of the manifest generator is to apply certain additional rules defined by operators. For example, based on geo location and some other parameters, an operator may decide to use a different CDN, or limit maximum resolution, bitrate, etc. The corresponding blocks providing for such functionality in Figure 3 are *rules API* and *rules engine*.

V. Just-in-time packaging

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

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

VI. Cascaded CDN architecture

To reduce frequency of at which just-in-time packagers are invoked the delivery system uses 2-layer CDN architecture. The first-layer CDN, which interfaces with packagers, is used for initial caching and propagation of the content to different regions. The second level CDNs, are then added in each region according to operator preferences, and provide edge caching as needed for delivery to end users.

As it follows from the above description, the proposed architecture is designed specifically to minimize transcoding, transmuxing, storage, and CDN delivery costs while supporting plurality of existing codecs, formats, and DRMs as needed for multi-screen delivery.

Measuring and Tuning Overall System Performance

I. *Bandwidth and usage statistics*

The collection of bandwidth, usage, and other relevant statistic in the system shown in Figure 3 is done by the *analytics engine*. It collects information from two sources: players and CDNs. Player's data are needed for understanding of which content segments have been played, as well as to measure buffering and start-up latencies. CDN statistics show which segments have been delivered to the device and at which speed. By pooling such data that analytics engine is able to collect variety of statistics, including information about usage and bandwidth distributions related to different categories of client devices.

Examples bandwidth distributions as measured for three different OTT operators are shown in Figures 4, 5, and 6. The associated device usage and average bandwidth statistics are showing in Tables 2, 3, and 4, respectively.

Device type	Usage [%]	Average bandwidth [Mbps]
PC	0.004	7.5654
Mobile	94.321	3.2916
Tablet	5.514	3.8922
TV	0.161	5.4374
All devices	**100**	**3.3283**

Table 2: Usage and average bandwidth statistics for Operator 1.

Device type	Usage [%]	Average bandwidth [Mbps]
PC	63.49	14.720
Mobile	6.186	10.609
Tablet	9.165	12.055
TV	21.15	24.986
All devices	100	16.393

Table 3: Usage and average bandwidth statistics for Operator 2.

Device type	Usage [%]	Average bandwidth [Mbps]
PC	0.0	N/A
Mobile	0.0	N/A
Tablet	0.0	N/A
TV	100	35.7736
All devices	100	35.7736

Table 4: Usage and average bandwidth statistics for Operator 3.

Based on information in the above tables and figures, it follows that bandwidth and usage statistics in these 3 cases are very different. The operator 1 streams predominantly to mobiles, and its effective average bandwidth across all devices is only 3.3Mbps. The operator 2 has a mixed distribution to PCs, mobiles, tablets, and TV screens. Its effective average bandwidth across all devices is about 16.393Mbps. The operator 3 streams only to TVs and average bandwidth in this case is much higher – around 35.77 Mbps.

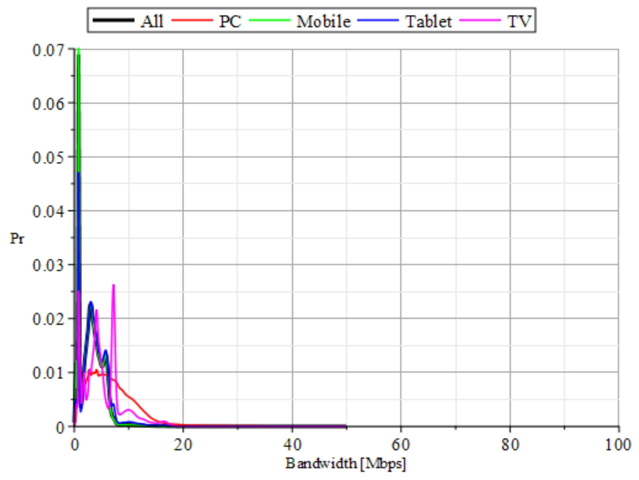


Figure 4: Bandwidth histograms measured for operator 1.

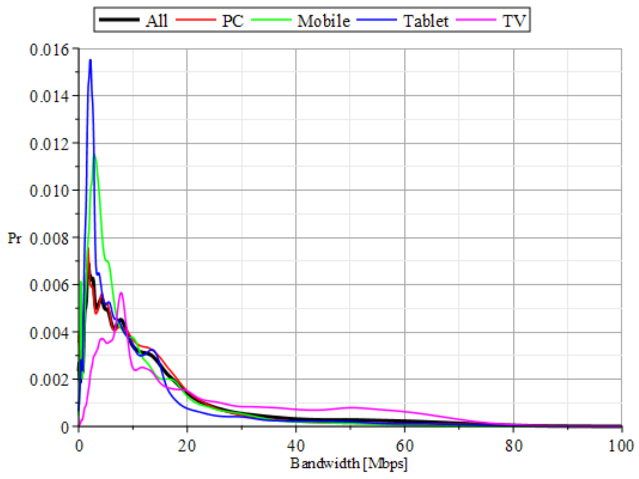


Figure 5: Bandwidth histograms measured for an operator 2.

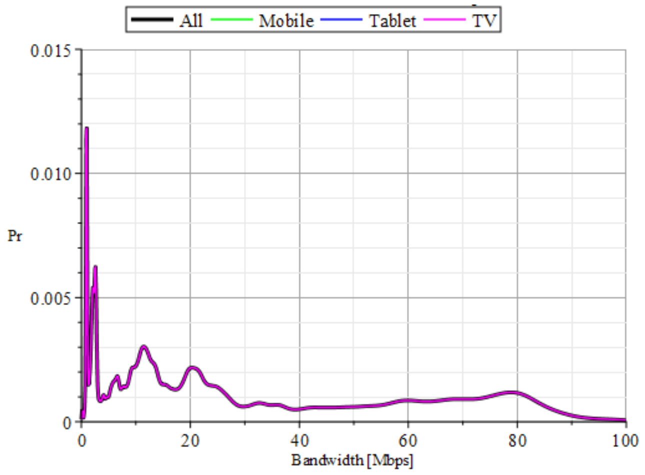


Figure 6: Bandwidth histograms measured for operator 3. This operator streams only to connected TVs.

II. *Playback statistics*

The analytics engine module also collects and reports variety of statistics related to *quality of experience* (QOE) during the playback. Examples of such statistics collected from above 3 operators are presented in Tables 5-7. In all cases, statistics are based on streaming of same test content, encoded using 9 renditions as prescribed by standard HLS ladder for H.264 codec [13]. This ladder is shown in Table 8.

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

TABLE 5: PLAYBACK STATISTICS FOR OPERATOR 1.

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

TABLE 6: PLAYBACK STATISTICS FOR OPERATOR 2

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

TABLE 7: PLAYBACK STATISTICS FOR OPERATOR 3.

First 9 rows in the above tables list relative frequencies of loading of each rendition. This is followed by buffering probability, start-up latency (in seconds), average bandwidth (in Kbps), and then average resolution (in frame heights) and average encoding quality (in SSIM [13]) as delivered by a combination of streams pulled by streaming clients.

The reported SSIM values are initially computed during encoding stage, and then retrieved and aggregated into final average value based on stream load statistics. Such statistics are provided separately to each category of devices, as well as in combined form, averaged across all devices (last column).

Rendition	**Profile**	**Resolution**	**Framerate**	**Bitrate**	**SSIM**
1	High	416x234	23.976	145	0.92231
2	High	640x360	23.976	365	0.94337
3	High	768x432	23.976	730	0.95776
4	High	768x432	23.976	1100	0.96788
5	High	960x540	23.976	2000	0.97148
6	High	1280x720	23.976	3000	0.96931
7	High	1280x720	23.976	4500	0.9753
8	High	1920x1080	23.976	6000	0.96861
9	High	1920x1080	23.976	7800	0.97217
Storage				25640	

TABLE 8: STANDARD HLS ENCODING LADDER & SSIM QUALITY LEVELS ACHIEVED FOR CONTENT USED IN THE ABOVE EXPERIMENT.

As easily noticed, despite the fact that content is identically encoded, the quality of experience delivered by these 3 operators is very different. The operator 3 pulls mostly top bitrate 1080p rendition (with probability of about 0.83), delivering on average about 1003 lines of resolution and encoding quality of about 0.971 SSIM. In case of operator 2, the probability of loading of top-most rendition drops to about 0.62 considering all devices, and just about 0.43 for mobiles. This results in lower resolutions (about 938 lines across all devices and only 867 lines for mobiles), as well as lower encoding quality, as ladder shown in Table 8 progressively drops encoding quality for lower resolutions. Finally, in the case of operator 1, the situation is even worse. Renditions 3, 5, 6, and 7 become most commonly used, resulting average delivered resolution of about 631 lines (628 for mobiles), and encoding quality of about 0.966 SSIM.

III. Means for tuning the system

The system depicted in Figure 3 includes several tools and means by which it can be adjusted or tuned to achieve best performance given each operator's context and needs.

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

The use of *rules API* and *rules engine* allows operators to select local CDNs and distribute traffic between them dynamically without disruption of operations. It also allows operators to impose limits and effectively add or remove some streams that can be delivered. The addition of streams, especially low-bitrate ones, may be considered as means for reducing buffering probability or load times. On the other hand, removal of some streams may be considered for reducing bandwidth usage or for improving CDN cache performance.

Finally, the system in Figure 3 also allows *encoding profile generation* for each new content item to be done *dynamically*, account for both characteristics of the content as well as existing bandwidth, usage, and playback statistics for each operator. We call such profile generation and encoding process *context-aware encoding* or CAE. This step effectively *closes the feedback loop* provided by the analytics engine, and allows encoding of new content to be done better, accounting for current context (delivery and playback statistics) of each operator.

IV. Context-aware profile generation

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

Importantly, in such optimization process, the quality estimates for each possible resolution and bitrate come from prior content analysis, and the estimates of stream load probabilities at each rate come from bandwidth statistics measured for each client. In computing final optimization cost expression, CAE generator aggregates estimates obtained for each type of client according to usage distribution, also provided by the analytics module. *In other words, CAE profile generation is really an end-to-end optimization process for multi-device /multi-screen delivery.*

The formal mathematical description of this optimization problem can be found in [12]. Reference [17] extends it to a case of designing profiles using multiple codecs and fragmentation of their support across different devices.

EXAMPLES OF OPTIMIZATIONS

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

I. Adaptations to different networks and devices

Let us now again consider 3 operators with bandwidth and usage statistics as presented in Figures 4-6 and Tables 2-4, respectively. Same test video sequence, is used in all cases. CAE encoding profiles generated for this sequence given statistics from each of the operators are shown in Tables 9-11.

In all cases, the CAE profile generator was given the same overall constraints that generally match characteristics of the HLS reference encoding ladder (see Table 8). This includes constraints on minimum and maximum bitrates, constraint on the maximum number of renditions and maximum change between bitrates in the encoding ladder, constraints on aspect ratios, framerates, and set of resolutions that can be used, etc.

However, as can be observed in Tables 9-11, CAE-generated profiles for each operator are somewhat different. For operator 1, it generated 7 renditions, with high density of points around 0-1 Mbps range. For operator 2, it also generated 7 renditions, however with faster rump towards higher resolutions and bitrates. Notice, specifically, that instead of selecting 540p resolution at 4th rendition, it selects 576p. Finally, in cases of operator 3, CAE generated only 5 renditions, which are even more sparsely placed apart. Such use of fewer renditions leads to lower transcoding costs and better CDN efficiency.

Besides the changes in the numbers of renditions, we also notice significant changes in total bitrates occupied by composition of all renditions in encoding profiles. Thus, all CAE generated profiles require significantly lower amount of storage.

Rendition	Profile	Resolution	Framerate	Bitrate	SSIM
1	Baseline	320x180	30	125	0.93369
2	Baseline	480x270	30	223.08	0.93793
3	Main	640x360	30	398.11	0.94636
4	Main	960x540	30	774.78	0.94953
5	Main	1280x720	30	1549.5	0.95637
6	High	1600x900	30	2765.3	0.96105
7	High	1920x1080	30	4935.1	0.96576
Storage				10771	

TABLE 9: CAE-GENERATED ENCODING LADDER FOR OPERATOR 1.

Rendition	Profile	Resolution	Framerate	Bitrate	SSIM
1	Baseline	320x180	30	125	0.93338
2	Baseline	480x270	30	239.71	0.94122
3	Main	640x360	30	469.54	0.95202
4	Main	1024x576	30	939.08	0.95221
5	Main	1280x720	30	1568.8	0.95658
6	High	1600x900	30	2765.3	0.96105
7	High	1920x1080	30	4935.1	0.96576
Storage				11026	

TABLE 10: CAE-GENERATED ENCODING LADDER FOR OPERATOR 2.

Rendition	Profile	Resolution	Framerate	Bitrate	SSIM
1	Baseline	320x180	30	125	0.93447
2	Baseline	512x288	30	307.42	0.94855
3	Main	960x540	30	803.59	0.95050
4	Main	1280x720	30	1727.8	0.95864
5	High	1920x1080	30	5050.7	0.96599
Storage				8014.6	

TABLE 11: CAE-GENERATED ENCODING LADDER FOR OPERATOR 3.

One extra notable difference between CAE profiles and reference HLS profile (Table 8) is that CAE uses mixed set of H.264 profiles, starting with Baseline, followed by Main and High profiles. In contrast, HLS ladder, recommended in Apple deployment guidelines [13], uses only High profile across all renditions. CAE generated profiles can therefore reach a much broader set of playback devices, including those that can only decode H.264 baseline.

Next, in Tables 12-14 we present playback statistics as measured for CAE encoded content after delivery across all three operators. The summary of relative differences between these statistics and ones obtained for reference HLS profile (cf. Tables 5-7) are presented in Table 15.

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

TABLE 12 PLAYBACK STATISTICS FOR OPERATOR 1 AFTER CAE OPTIMIZATION.

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

TABLE 13 PLAYBACK STATISTICS FOR OPERATOR 2 AFTER CAE OPTIMIZATION.

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

TABLE 14 PLAYBACK STATISTICS FOR OPERATOR 3 AFTER CAE OPTIMIZATION.

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

TABLE 15 EFFECTS OF CAE OPTIMIZATION FOR 3 OPERATORS.

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

Based on information presented in Table 15, it can be observed that the use of CAE optimizations resulted in significant savings of resources in all 3 cases. The number of renditions, and consequently transcoding/compute costs were reduced by 22.2 to 44.4%. The amount of storage was reduced by 56.9 to 68.7%, reducing cloud storage and bandwidth costs. The changes in average bandwidth use are also significant, but more depended on operator's context. For example, for operators 2 and 3, which deliver mostly over high speed networks, the bandwidth savings ranged from 31.3 to 33.9%. For operator 1, delivering over very slow connections (with average bandwidth around 3.3 Mbps) the reductions in average bandwidth use were more modest – about 8.44%. However, the average resolution delivered to this operator become over 27% higher (804 lines on average vs 631), and average start-up latency also got decreased by over 5.7%. In other words, the use of CAE optimizations for operator 1 have resulted in the *increase of quality of experience, in addition to savings in bandwidth, storage, and compute costs.*

All such optimizations become possible by tuning encoding profiles to each of the operator's network distributions and distributions of playback time between different categories of receiving devices.

II. Adaptations to different types of content

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

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

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

TABLE 16: AVERAGE SAVINGS AS MEASURED FOR DIFFERENT CONTENT CATEGORIES, OPERATOR 2.

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

The results are summarized in Table 16. All numbers represent relative changes between respective statistics obtained for encodings produced using default HLS ladder (Table 8) vs CAE. For compactness of presentation, only the changes in renditions, storage, bandwidth, and resolution are presented. The changes in other statistics were minor (<2%). By looking at data in Table 16, it can be observed, that CAE improvements are significant across all categories of content. We also note, that for some categories of content, such as "Interviews" or "Golf", the changes in bandwidth are extremely high (we see savings of about 74%), while for some other categories, such as "Swimming" or "Hockey", such savings are considerably lower (19-22%). The savings in storage are more consistent across all categories of content. The changes in the numbers of renditions are also more consistent across all categories of content.

The above study was produced using H.264 encoder, and for SDR content. In our experience, we also noted that CAE savings when using HEVC encoders are generally similar in magnitude and have same general dependency on the characteristics of the content.

III. *Multi-codec profile optimizations*

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

The use of multi-codec profile generation leads to *additional savings in the total number of renditions and quality gains achievable by clients that can switch between the codecs.*

For example, let us consider a set of mixed H.264+HEVC ladder points presented in Figure 7. Here, by green and red lines we plot quality-rate functions achievable for a given content by HEVC and H.264 codecs respectively. The set of H.264 streams is connected by an orange line, forming a "staircase" of quality levels achievable by the H.264-only client. Similarly, the set of HEVC streams is connected by gray line, forming another "staircase" representing quality levels achievable by HEVC-only clients. The dotted blue line is used to connect points that may be used by clients that can switch between both codecs. By following the shape of this blue staircase, it becomes immediately obvious that *hybrid/switchable clients should be able to achieve better performance* than the other clients, as they effectively operate with a finer-grain ladder, delivering progressively better quality. But naturally, to enables such improvements, the locations of H.264 and HEVC streams need to be chosen carefully, and in consideration of quality-rate characteristics of both codecs for given content.

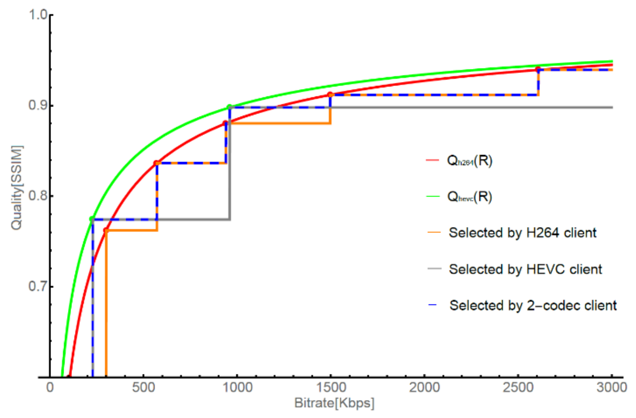


FIGURE 7: H.264 AND HEVC-ENCODING LADDERS AND QUALITY LEVELS ACHIEVABLE BY DIFFERENT TYPES OF CLIENT DEVICES.

Additional results and discussion about how multi-codec profiles can be optimally generated will be presented in [17].

CONCLUSIONS

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

REFERENCES

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

[2] G. J. Conklin, G. S. Greenbaum, K. O. Lillevold, A. F. Lippman, and Y. A. Reznik, "Video coding for streaming media delivery on the internet," IEEE Transactions on Circuits and Systems for Video Technology, vol. 11, no. 3, pp. 269–281, March 2001.

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

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

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

[6] ISO/IEC 23009-1:2012, "Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats," February 2012.

[7] ISO/IEC 23000-19, Information technology - Coding of audio-visual objects - Part 19: Common media application format (CMAF) for segmented media. https://www.iso.org/standard/71975.html

[8] Media Source Extensions, W3C, https://www.w3.org/TR/media-source/

[9] Encrypted Media Extensions, W3C, https://www.w3.org/TR/encrypted-media/

[10] A. Zambelli, Smooth streaming technical overview, Microsoft Corp., Redmond,WA, USA, Tech. Rep. [Online]. Available: http://www.iis.net/learn/media/on-demand-smooth-streaming/smooth-streamingtechnical-overview

[11] 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

[12] Y. A. Reznik, K. O. Lillevold, A. Jagannath, J. Greer, and J. Corley, "Optimal design of encoding profiles for ABR streaming," in Proc. 23rd Packet Video Workshop (PV'2018), Amsterdam, The Netherlands, June 12, 2018, pp. 43–47.

[13] Apple Inc., "HLS authoring specification for Apple devices," https://developer.apple.com/documentation/http_live_streaming/hls_authoring_specification_for_apple_devices, September 2018.

[14] A. Aaron et al., "Per-title encode optimization," https://medium.com/netflix-techblog/per-title-encode-optimization-7e99442b62a2, December 15 2015, Netflix technology blog.

[15] Ultra HD Forum, "Ultra HD Forum phase B guidelines," https://ultrahdforum.org/wp-content/uploads/Ultra-HD-Forum-Phase-B-Guidelines-v1.0.pdf, April 2018.

[16] Z. Wang, L. Lu, and A. C. Bovik, "Video quality assessment based on structural distortion measurement," Signal Processing: Image Communication, vol. 19, no. 2, pp. 121 – 132, 2004.

[17] Y. A. Reznik, X. Li, K. O. Lillevold, A. Jagannath, and J. Greer, "Optimal design of multi-codec profiles for ABR streaming," in Proc. IEEE Int. Conf. Multimedia and Expo (ICME'2019), Shanghai, China, July 8-12, 2019 – submitted.