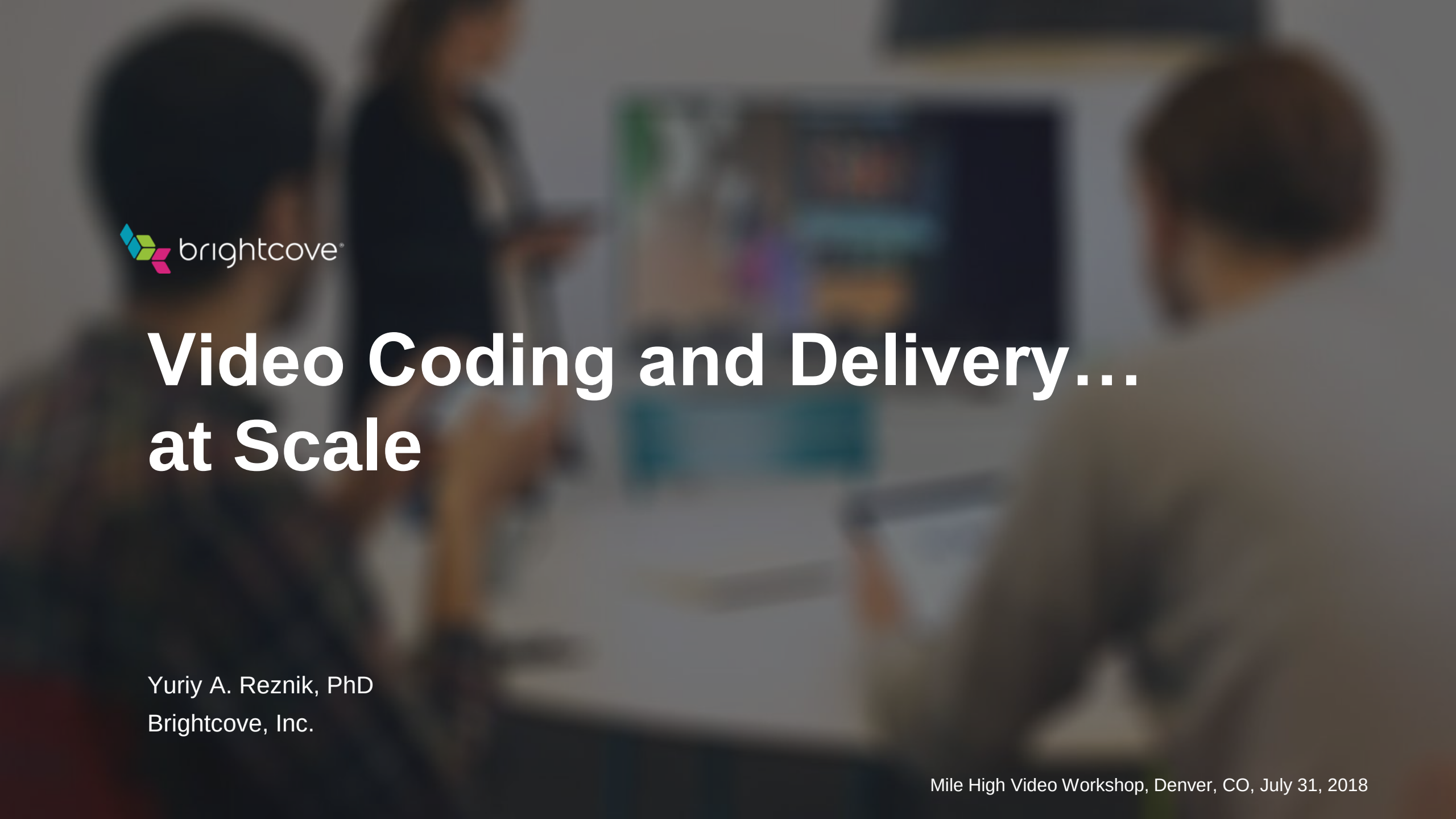


Video Coding and Delivery… at Scale

Yuriy A. Reznik, PhD
Brightcove, Inc.

Mile High Video Workshop, Denver, CO, July 31, 2018

About Brightcove

FOUNDED

2004

IPO 2012

HEADQUARTERS

BOSTON, MASSACHUSETTS

500+ EMPLOYEES

REVENUE

$150M+

IN 2017

CUSTOMERS

4,500+

IN MORE THAN 70 COUNTRIES

SCALE

2.7B+

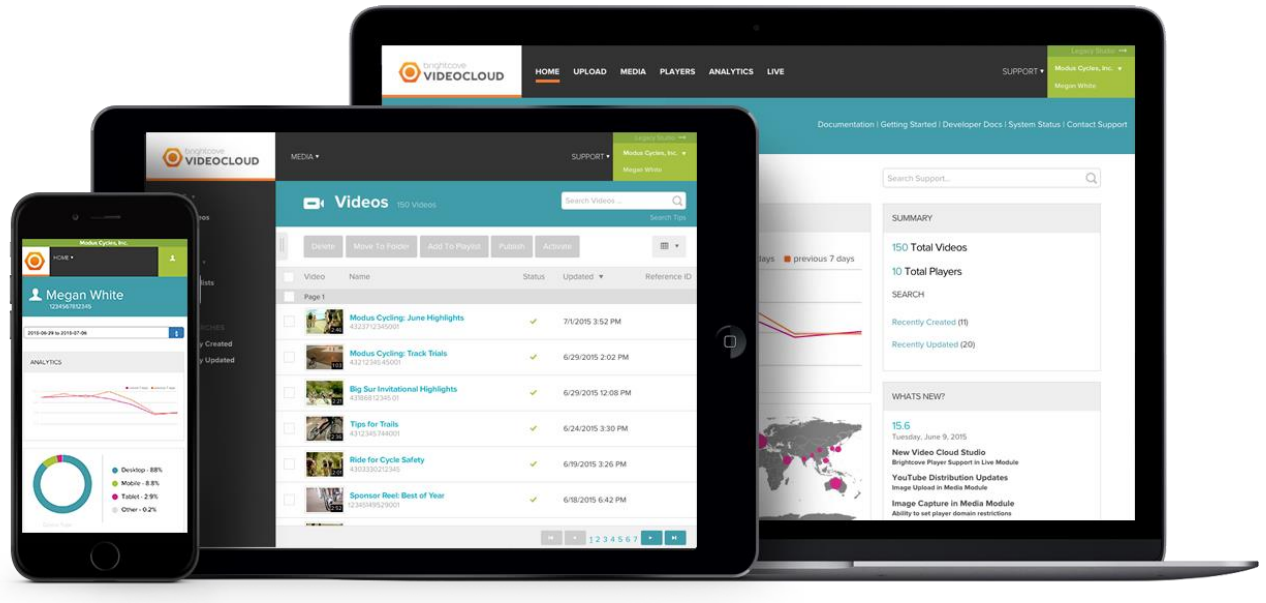
AVERAGE MONTHLY STREAMS

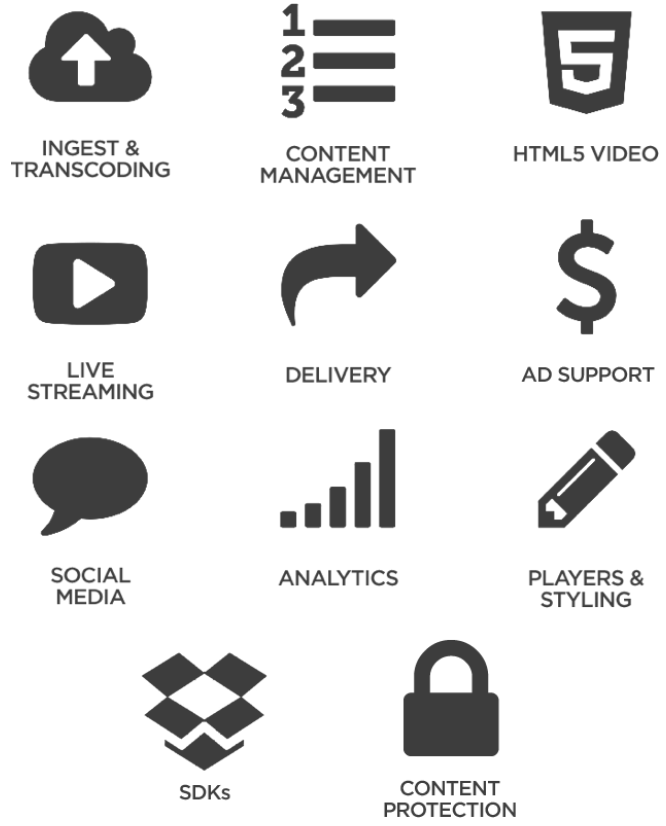
CORE OFFERINGS

- brightcove VIDEO CLOUD™
- brightcove ZENCODER™
- brightcove LIVE™
- brightcove OTT FLOW™ powered by accedo
- brightcove PLAYER™

Global Presence & Reach

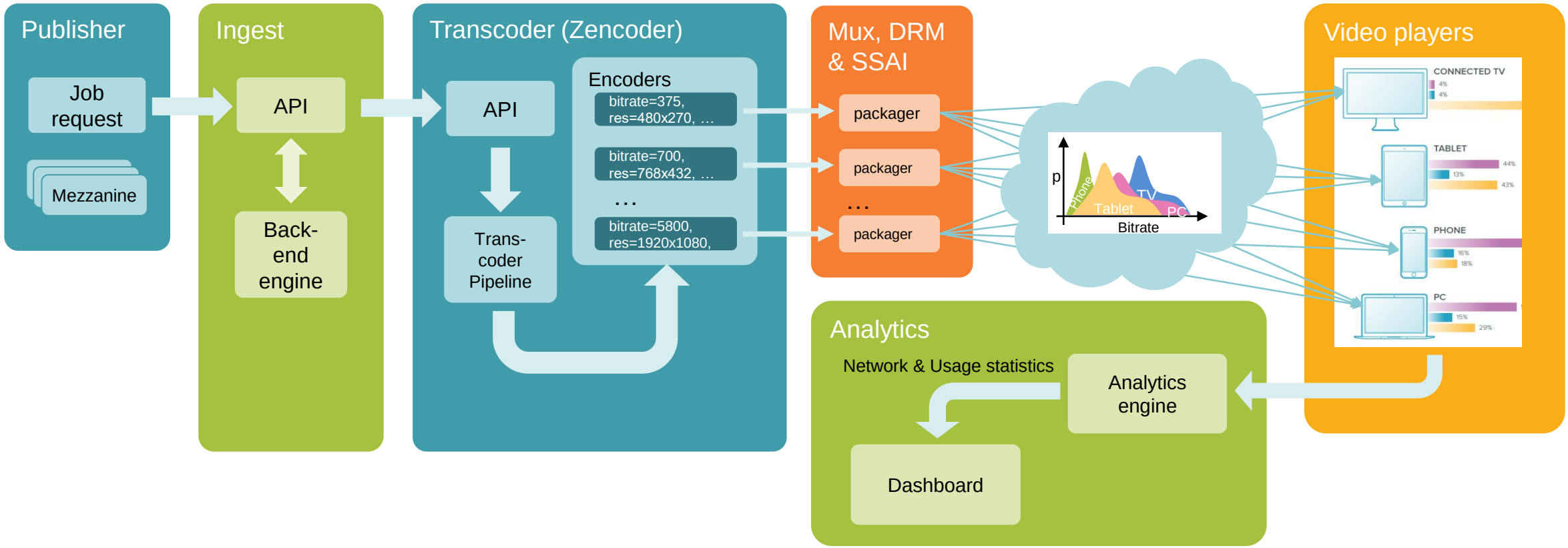
brightcove
VIDEOCLOUD





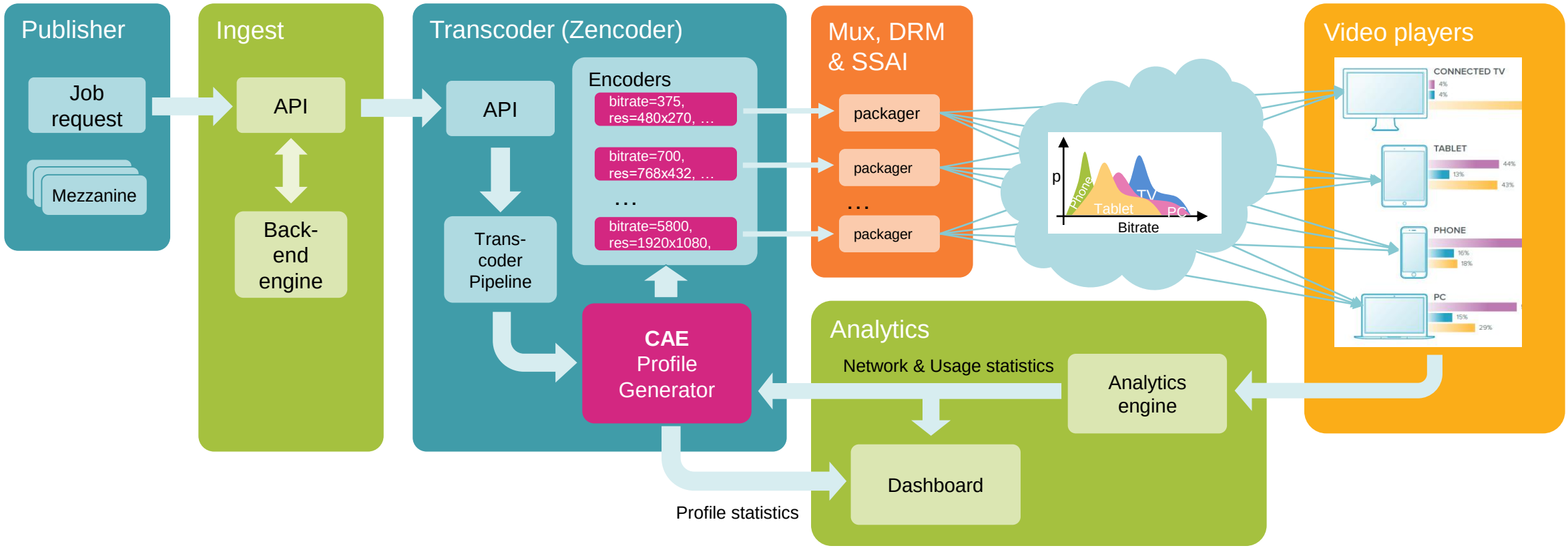
VideoCloud Architecture

Main components / chain of processing steps:



VideoCloud Architecture

Same chain with Context-Aware Encoding (CAE) module:



Key Challenges

Scale

- need to process millions of concurrent streams

Reliability

- things have to work (even if input is totally broken, cloud region is down, etc.)
- everything needs to be redundant and fault-tolerant

Reach

- streams must be decodable on most devices in existence
- including some legacy TVs, game consoles, Blackberries, etc.

Fragmentation

- new codecs, flavors of HDR, DRMs, delivery formats, etc.

Operating costs

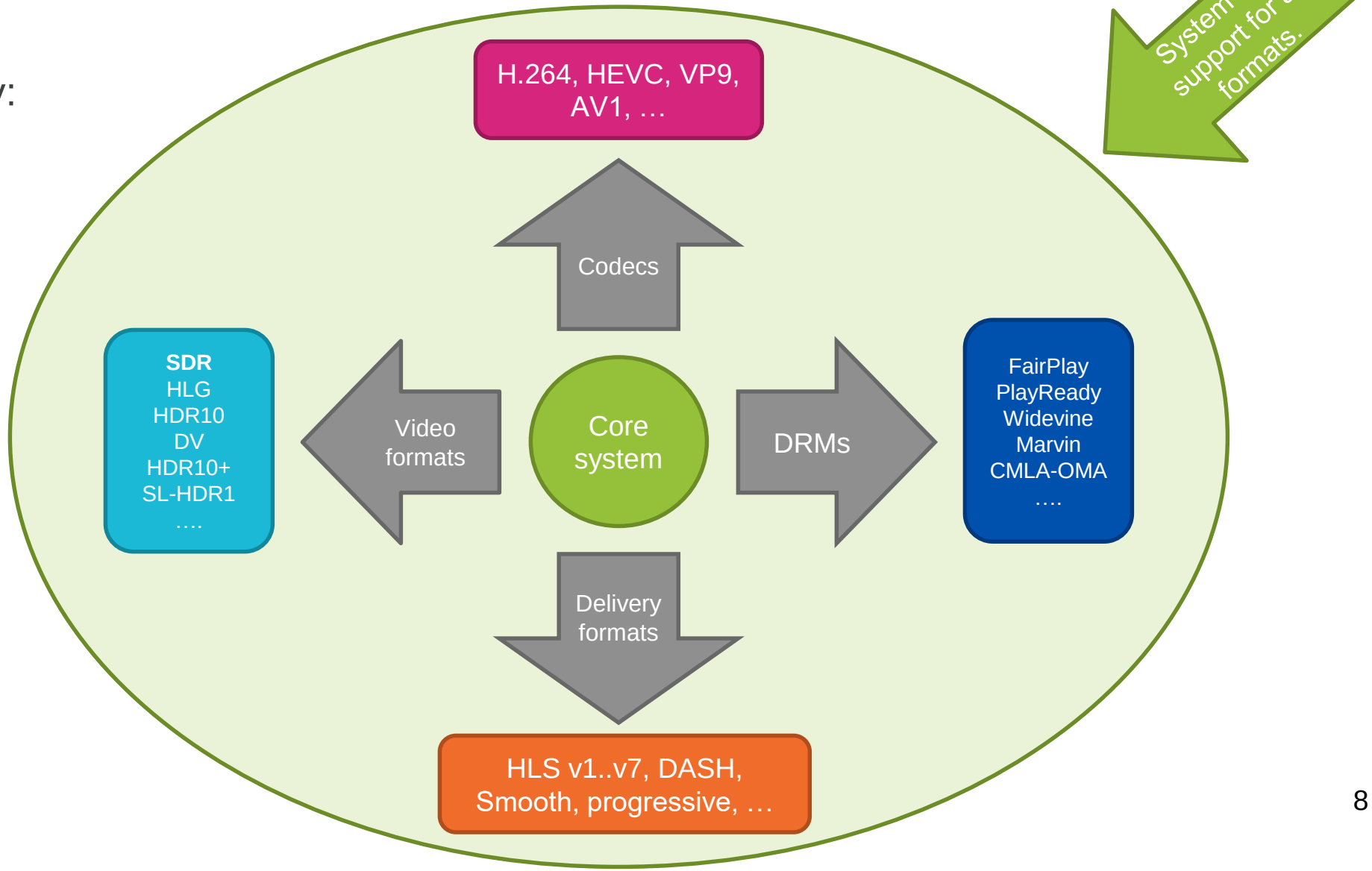
- cloud, bandwidth, CDN bills…

Quality

- delivered streams must be of high quality

Supporting multiple devices & formats

System complexity:



A thought experiment…

Number of streams one needs to generate to reach existing devices:

7 streams per media asset
x 3 codecs (H.264/baseline, H.264, HEVC)
x 3 DRMs (Fairplay (iOS), PlayReady (TVs), Widevine Modular (Android))
x 3 formats (HLS (iOS), DASH (Android, TVs), progressive (legacy))

= 7 * 27 = 189 streams !!!

This is simply not practical!

Platform / DRM support matrix:

	Media players	PlayReady	Widevine Modular	Widevine Classic	FairPlay	Primetime	Marlin	CMLA-OMA
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	✓	✗	✗	✗	✗	✓	✗
Specs	TNT (2.0+)	✓	✗	✗	✗	✗	✓	✗
	HbbTV (1.5+)	✓	✓	✗	✗	✓	✓	✓

Source: https://castlabs.com/resources/drm-comparison/

Minimizing costs of supporting multiple formats / devices

Just-in-time packaging

- packaging and applying DRM dynamically, when clients request streams
- avoids creation of redundant (or unused) streams

Device detection

- detecting and recognizing device capabilities
- targeting manifests and streams accordingly

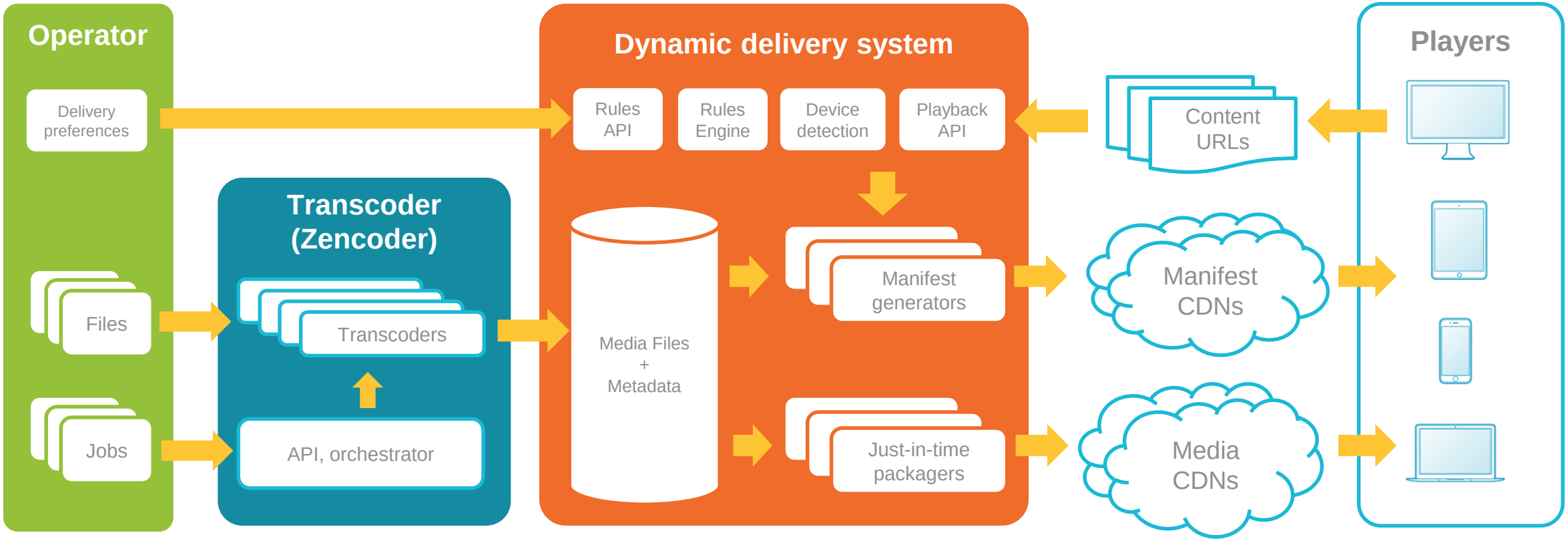
Rules engine

- allows operators to set specific limits on stream profiles delivered to devices of each kind
- e.g. one can specify max resolutions, bitrates, choices of codecs/profiles, etc.

Optimized multi-codec profile generation

- minimizes number of streams needed based on characteristics of content and intended set of delivery devices

Dynamic Delivery system in VideoCloud

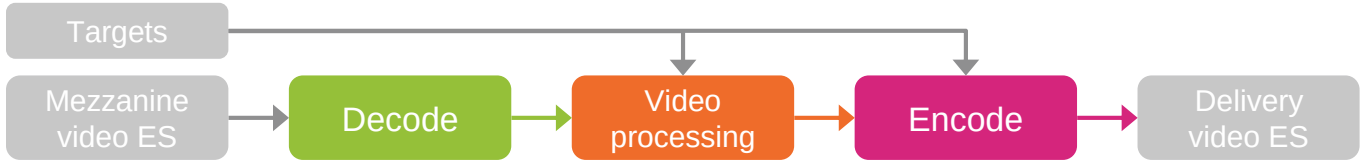


The use of just-in-time packaging reduces the number of format/DRM combinations to one that is necessary to support active players.

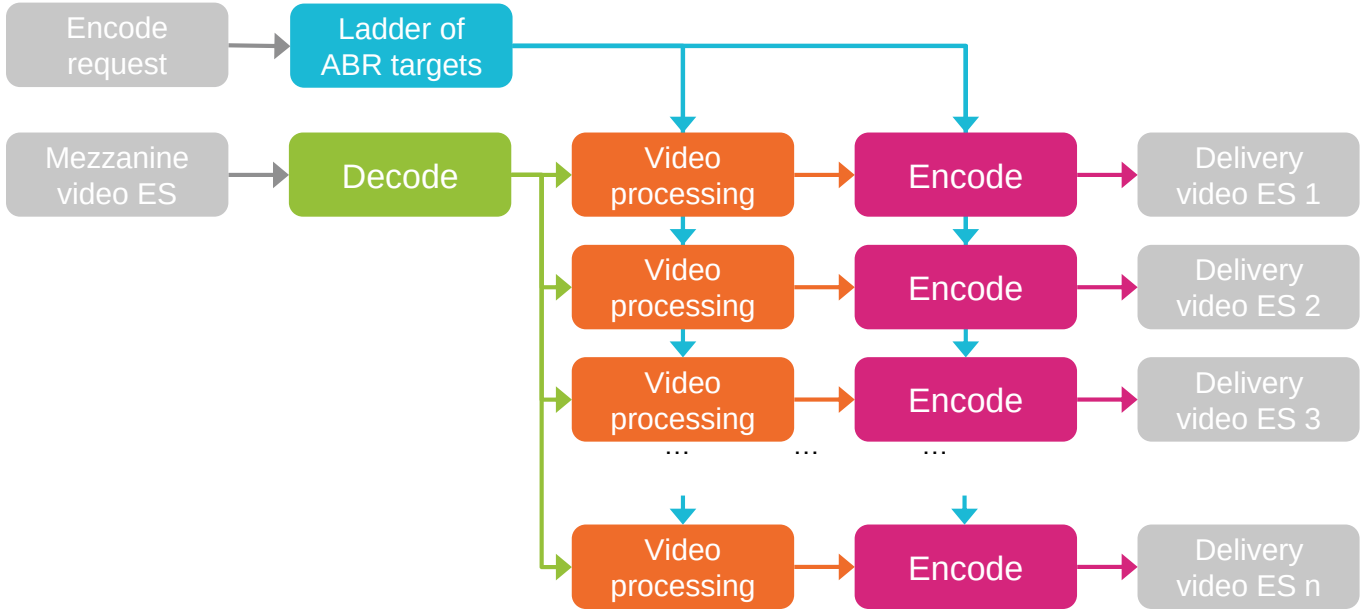
This reduces CDN, storage and transcoding costs!

Video transcoding

Single-rate transcoding:



Transcoding for ABR:



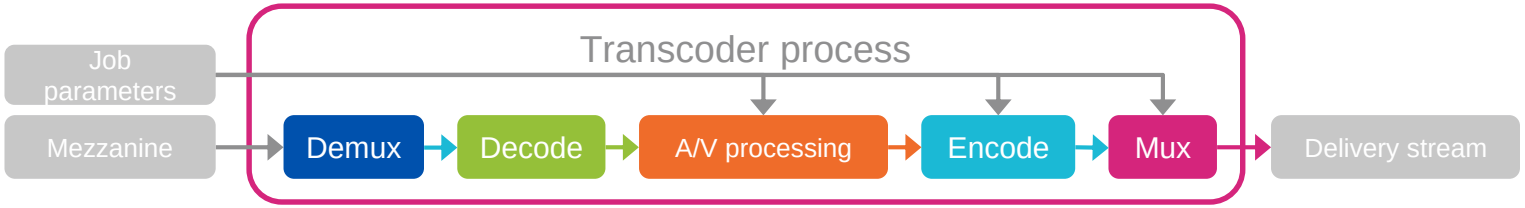
Key operations:

- ABR ladder design
- video processing
- encoding

Transcoding in the cloud

Single transcoding operation:

- executed on a single system

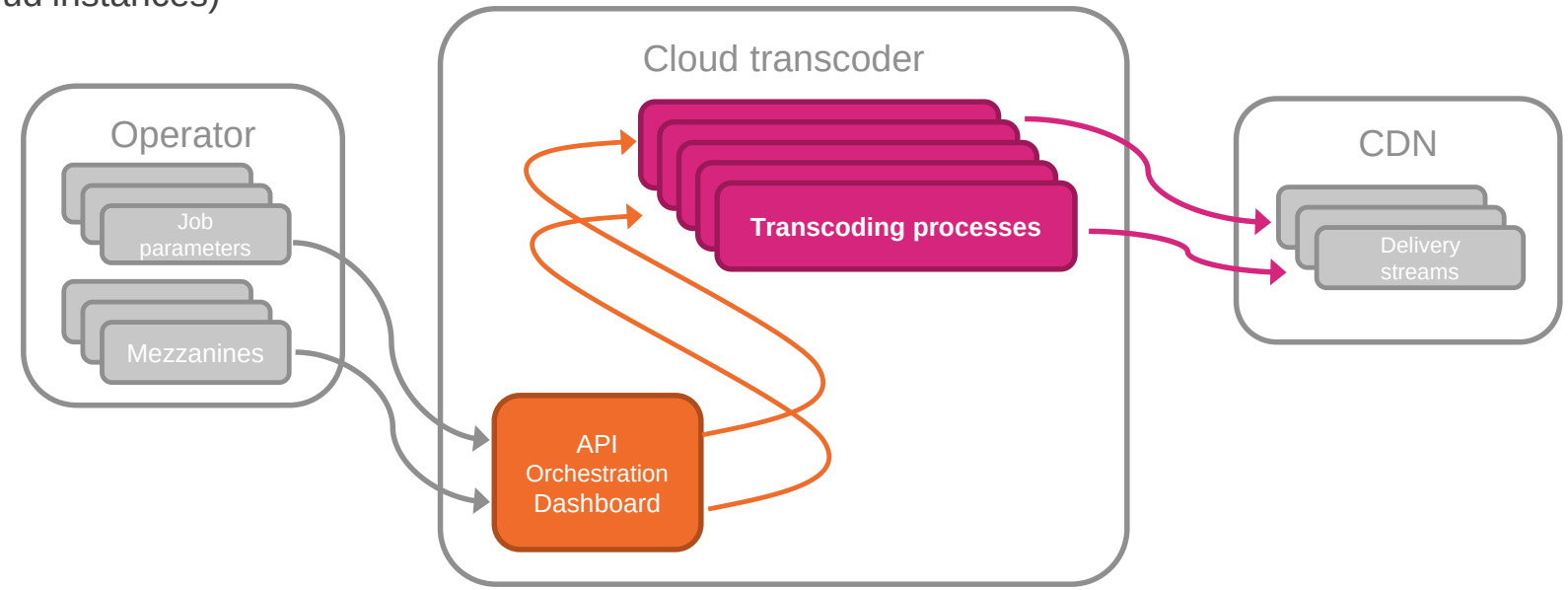


Plurality of transcoder processes executed in the cloud:

- executed on plurality of systems (cloud instances)

Key aspects:

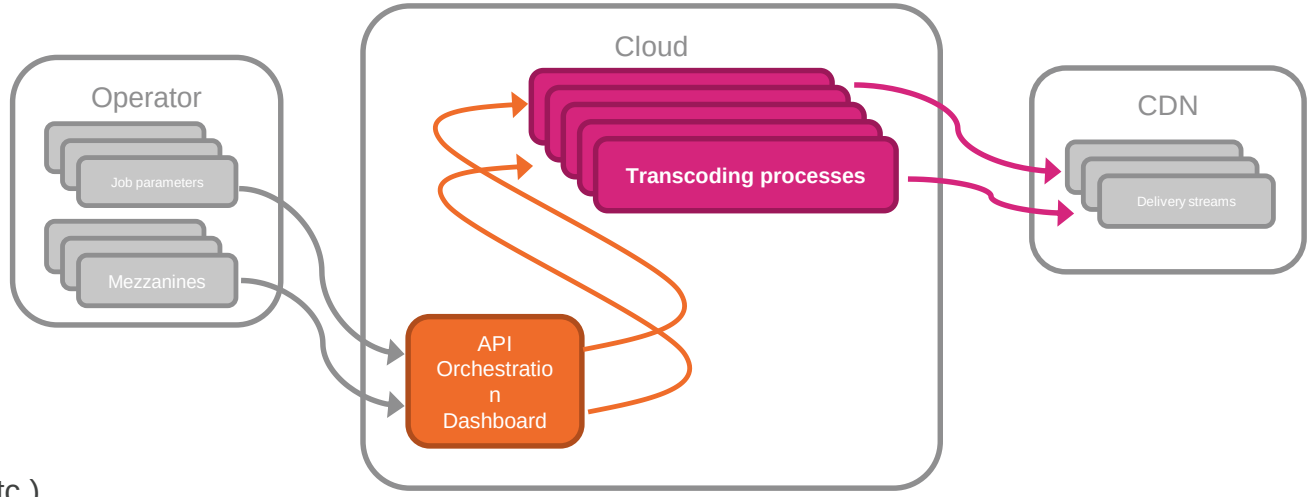
- API
- Distribution of work
- Cloud orchestration
- Monitoring
- Ensuring reliability
- Minimizing costs



Distribution of work

Types of parallelization

- stream-level
  - each stream is encoded at dedicated instance
- segment-level
  - stream is split in segments
  - each segment is encoded at dedicated instance
- instance-level
  - each instance has multiple CPU/GPU cores
  - encoding job is split and allocated to those cores
  - codec-level parallelization (slices, tiles, wavefronts, etc.)



When and what works?

- instance-level: always works
- stream level: works in most cases; packages of ABR streams can be more effectively processed jointly
- segment-level: only for VOD

Transcoding API, notifications, monitoring

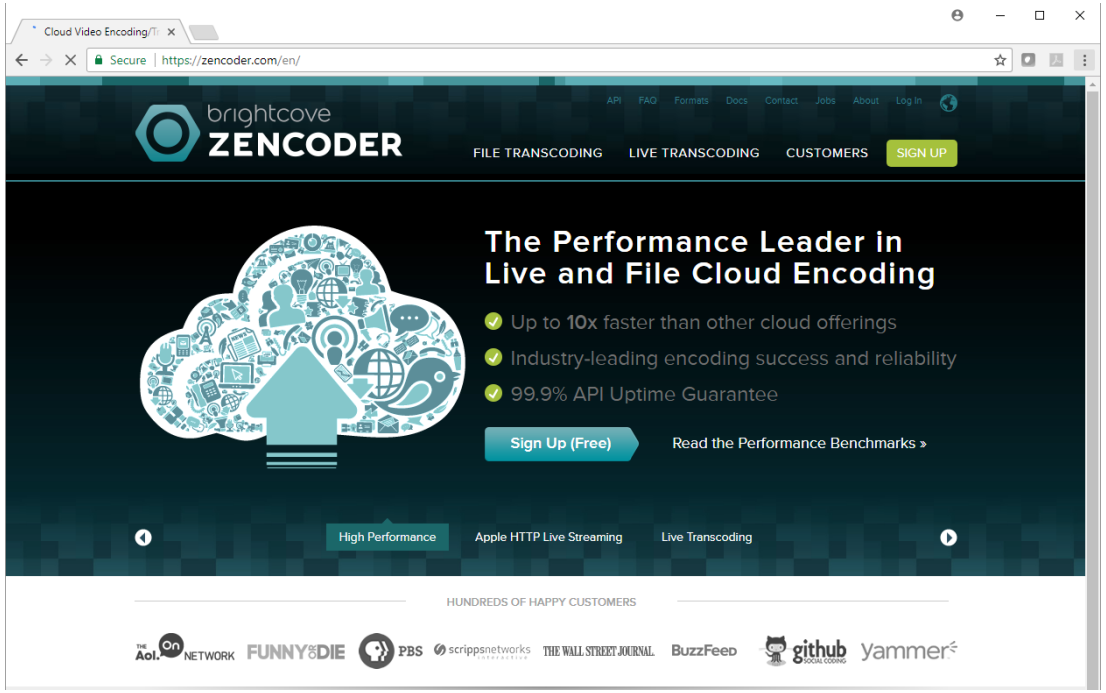
Example: encoding Job API in Zencoder:

An HTTP POST to: https://app.zencoder.com/api/v2/jobs

```
{
  "input": "s3://zencodertesting/test.mov",
  "output": [
     {
        "label": “stream-1-240k",
        "audio_bitrate": 56,
        "audio_sample_rate": 22050,
        "base_url": "s3://my-bucket/",
        "decoder_bitrate_cap": 300,
        "decoder_buffer_size": 800,
        "max_frame_rate": 15,
        "public": 1,
        "type": "segmented",
        "video_bitrate": 200,
        "width": 400,
        "format": "ts"
     },
     …
  ]
}
```

To receive notifications about job status/completion, operator can specify:

```
{
  "notifications": [
   "http://user:password@example.com/zencoder",
   "admin@example.com"
 ],
…
```



Optimizing cloud-based transcoder

Key topics:

- optimizing job allocations
- ensuring reliability
  - redundancy
  - reliable ingest & upload techniques
  - support for legacy & broken formats
- managing costs
  - selecting cloud, region, instances, etc.

In addition, the transcoder by itself must be good!

Attention must be given to:

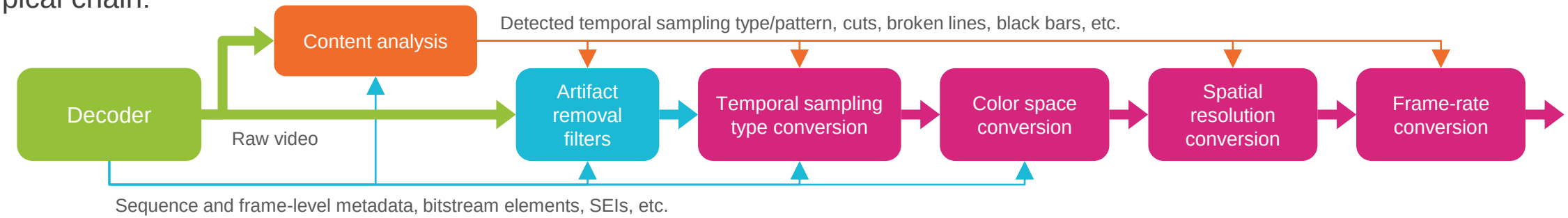
- pre-processing
- setting right codec constraints
- optimizing encoding ladder design

Pre-processing

Key functions:

- understand what type of content it is (which may be different from the way it was previously encoded)
  - progressive / interlace / telecine, cadence type, field order, etc.
- minimize artifacts introduced by prior generation encoder or sampling process
  - blocking, ringing, broken lines, temporal noise, etc.
- perform conversion from source format to format needed for delivery. This includes conversions of:
  - spatial resolution
  - chroma sampling type (4:4:4, 4:2:2, 4:2:0, different kinds of 4:2:0)
  - frame rate
  - temporal sampling type (progressive, telecine, interlace, field order)
  - color (gamma, matrix, primaries, EOTF, mastering display color volume, other display-related metadata, etc.).

Typical chain:



Encoding: quality-rate tradeoffs

Each ABR stream (“representation”) is basically a tradeoff between:

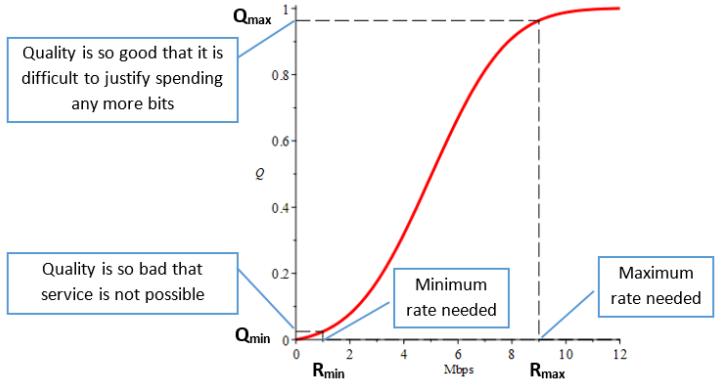
- **bitrate** R used to send video over the network, and
- **quality** Q achieved when video is rendered by receiver

A combination of (R, Q) pairs achievable for a particular sequence and codec is commonly called **quality-rate function** $Q(R)$. It is normally monotonic and saturating with $R \to \infty$.

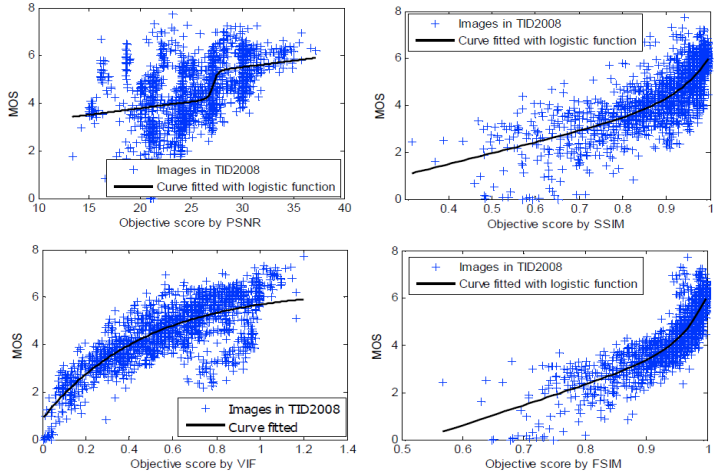
Notes on quality metrics:

- broadly available metrics (MSE, PSNR, SSIM, FSIM, etc.) measure only **codec noise**
  - full-reference, specific to each resolution (spatial, framerate, sampling type, etc.)
  - have no means for discrimination between resolutions, reproduction volume (HDR/SDR), etc.
- tools/models accounting for different resolutions, display & environmental parameters:
  - VDP-derivatives, Tektronix PQA (PQR metric), SSIMWave SSIM+, etc.
- but… none of these metrics/models is perfect
  - human-measured scores typically scatter around
  - the most one can infer from an objective quality score is the **probability** that user assessment of quality is above or below a given threshold.

Hypothetical shape of quality-rate function:



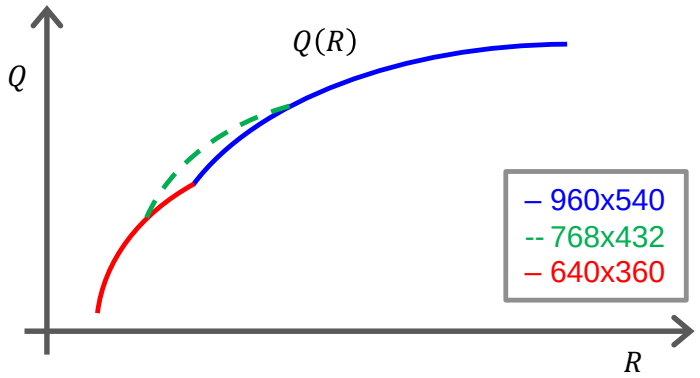
Some existing metrics (Zhang, et al 2011):



Encoding: choices of resolutions

In theory, the more resolutions are available the better:

- allows better quality / rate tradeoffs:



In practice, the choices of resolutions are also constrained by

- content owners
- industry forum guidelines: DVB, HBBTV, DTV, etc.
- closed ecosystem guidelines: Apple TV, etc.
- capabilities of playback devices

DVB recommended resolutions

10.3 Luminance Resolutions and Frame Rates

A Player that supports HD content shall support the decode and display of pictures with the resolutions in Table 17 and Table 18 at all supported frame rates.

NOTE 1: This does not preclude the use of other resolutions within an Adaptation Set, however, a limited number of resolutions are listed here to ease Player testability.

NOTE 2: The resolutions in the table are the resolutions in the Representations within an Adaptation Set. These may not be the same as the final display resolution, and are thus independent of region specific variations that are prevalent in Broadcast TV.

Table 17: Luminance Resolutions for progressive content

Horizontal @maxwidth	Vertical @maxheight
1 920	1 080
1 600	900
1 280	720
1 024	576
960	540
852	480
768	432
720	404
704	396
640	360
512	288
480	270
384	216
320	180
192	108

Table 18: Luminance Resolutions for interlaced content

Horizontal @maxwidth	Vertical @maxheight
1 920	1 080
704	576
544	576
352	288

A Player that supports UHDTV content shall support the decode and display of pictures with the resolutions shown in Table 19 in addition to the resolutions in Table 17 and Table 18.

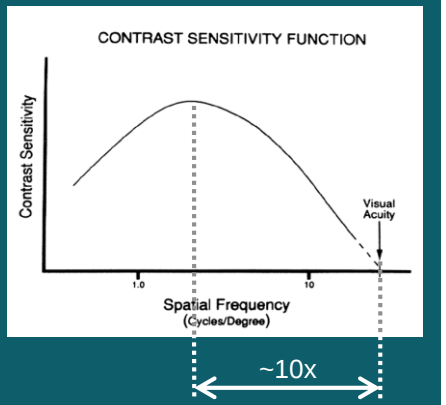
NOTE 3: This does not preclude the use of other resolutions within an Adaptation Set, however, a limited number of resolutions are listed here to ease Player testability.

Table 19: Luminance Resolutions for UHDTV Progressive Content

Horizontal @maxwidth	Vertical @maxheight
3 840	2 160
3 200	1 800
2 560	1 440

For service continuity, reducing the frame rate may be beneficial at lower bitrates, so lower frame rates than are found elsewhere in the present document are needed. A Player shall support frame rates formed by a division by 2 and 4 of those of the frame rate families defined in clause 10.4 that it supports.

Range of resolutions



Note: 10x downscaled videos start losing details that are most prominently visible in normal reproduction setting.

This suggests that the range of resolutions that can be used for encoding should not be larger than 10.

Coincidentally, this is precisely the range supported by the DVB ladder.

Setting right codec constraints

VBR control

- for ABR streaming streams must be capped VBR !!!
  - the use of highly-variable VBR encoding may confuse DASH clients and cause buffering or inefficient use of bandwidth
- typically, maximum bitrate cap is set to about 10-35% above average bitrate
  - **must be lower than next target bitrate** in the ABR encoding ladder
- decoder buffer size must also be limited

GOP length and type

- GOP length must be shorter or equal than GCD of delivery segment lengths
  - E.g. for 4,6, and 10-sec segments, GOP <= gcd(4,6,10) = 2 sec.
- GOP length may also be affected by the need to support ad-insertions/splicing, etc.
- closed GOP is most commonly used

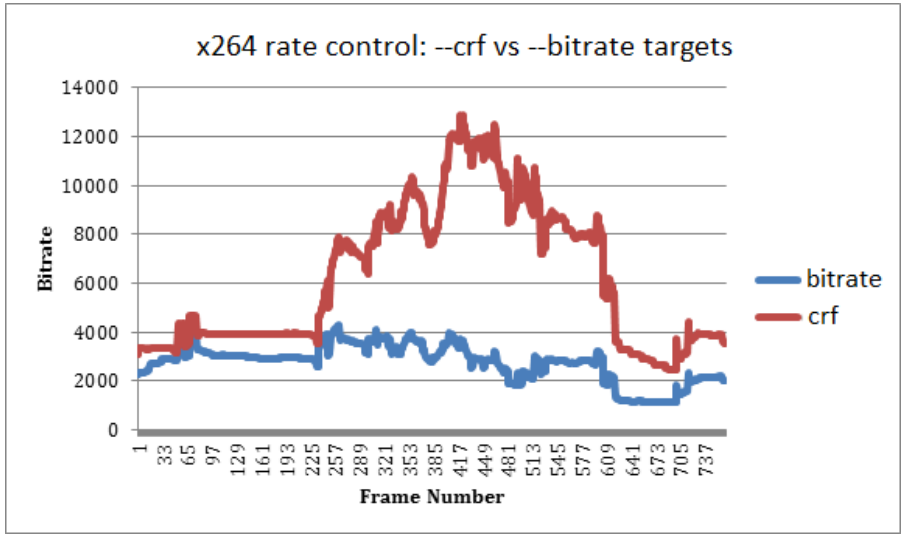
Profiles & levels

- H.264 Baseline profile is needed for legacy devices (e.g. mobile devices prior to 2012)
- Main profile is adequate for streams of up to 720p
- High is better for 1080p and beyond
- Level must be sufficient to carry stream at given resolution, framerate, bitrate, CPB size

Reference frames, B frames:

- for legacy devices (e.g. mobiles of 2012 and earlier) – no B frames, 1 reference
- most STBs can support up to 4 reference frames, 3 B-frames

Example of uncapped VBR behavior:



Example level/profile combinations:

Profile	Level	@codec Parameter (avc1 sample entry)	@codec Parameter (avc3 sample entry)
Constrained Baseline	2.1	avc1.42c015	avc3.42c015
Constrained Baseline	3.0	avc1.42c01e	avc3.42c01e
Main	3.0	avc1.4d401e	avc3.4d401e
Main	3.1	avc1.4d401f	avc3.4d401f
High	3.0	avc1.64001e	avc3.64001e
High	3.1	avc1.64001f	avc3.64001f
High	3.2	avc1.640020	avc3.640020
High	4.0	avc1.640028	avc3.640028

Optimizing ABR encoding profiles

Encoding profile parameters:

- bitrates: $R = (R_1 \dots, R_n)$
- resolutions: $S = (S_1, \dots, S_n),\ S_i = (w_i, h_i, f_i)$ (width, height, framerate)
- codec-related constraints $C = (C_1, \dots, C_n)$

Typical ladder design constraints:

- bitrate monotonicity: $R_{\min} \leq\ R_1 < \cdots < R_n \leq R_{\max}$
- bound on first rate/start-up latency: $R_1 \leq R_{1,\max}$
- bitrate granularity constraints: $\gamma_{\min} \leq \frac{R_{i+1}}{R_i} - 1 \leq \gamma_{\max},\ i = 1, \dots, n-1$
- resolution monotonicity: $\|S_1\| \leq \cdots \leq \|S_n\|,\ \|S_i\| =\ w_i \cdot h_i \cdot f_i$
- codec noise thresholds: $q_{\min} \leq\ q_{S_i}(R_i),\ \mathrm{i} = 1, \dots, \mathrm{n}$
- quality monotonicity: $Q_{\min} \leq\ Q(R_1) < \cdots < Q(R_n)$

Optimal ladder design:

- find: number of points n^*, bitrates $R^* = (R_1^*, \dots R_{n^*}^*)$, resolutions $S^* = (S_1^*, \dots S_{n^*}^*)$, and constraints $C^* = (C_1^*, \dots C_{n^*}^*)$, such that:

$$\Phi(n^*, R^*, S^*, C^*) = \max_{\substack{n \in \mathbb{N},\ R \in \mathbb{R}^n,\ S \in \mathbb{R}^{3 \times n} \\ +\text{all constraints}}} \Phi(n, R, S, C)$$

where: $\Phi(n, R, S, C)$ is a figure of merit function (e.g. best overall quality, lowest cost, etc.)

Brightcove ABR profile generator

Available as “CAE” ingest profiles in VideoCloud and Zencoder:

In VideoCloud there are 3 standard CAE profiles, as shown above.

Advanced users can also define custom CAE profiles using JSON descriptor.

This way users may specify limits for:

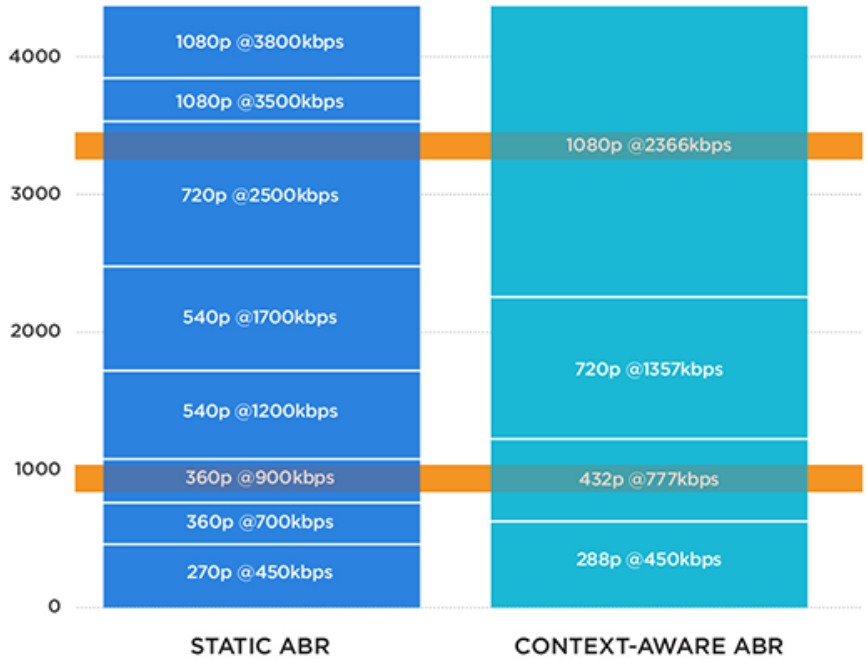
- number of renditions
- range of bitrates to be used
- list of allowed video resolutions, framerates, codec profiles & levels
- allowed granularity of rate steps in the profile
- operator-specific network and usage statistics, etc.

Example custom CAE profile:

```
{
"id": "1234567890",
"version": 1,
"name": "custom-cae-profile",
"dynamic_origin": {
  "dynamic_profile_options": {
    "min_renditions": 2,
    "max_renditions": 6,
    "max_resolution": {
      "width": 1920,
      "height": 1080
    },
    "max_bitrate": 4200,
    "max_first_rendition_bitrate": 400,
    "max_frame_rate": 30,
    "keyframe_rate": 0.5,
    "select_baseline_profile_configuration": true,
    "max_granularity": 100,
    "video_configurations": [
        {"width": 1280, "height": 720},
        {"width": 960,  "height": 540},
        {"width": 640,  "height": 360}
    ]
}}}
```

Brightcove CAE: an example

Example of static vs CAE-generated encoding profiles:



Demo: https://www.brightcove.com/en/blog/2017/05/context-aware-encoding-improves-video-quality-while-cutting-costs

Advantages:

- CEA saves bandwidth: 2366kbps vs 3800kbps at 1080p => 36.77% saving!
- CAE saves number of renditions / transcoding costs: 4 vs 8
- CAE delivers more pixels / better quality at low rates: 432p @ 777kbps vs 360p @ 900kpbs

Summary

Modern-era OTT video delivery poses a number of challenges, and requires some highly specialized tools:

- Dynamic packaging & delivery
  - addresses the need to support multiple devices, codecs, DRMs, and delivery formats
  - minimizes storage, bandwidth and CDN costs
- High-quality mass-scale transcoder
  - scalable at instance, segment, and stream-levels
  - supporting all sorts of input formats
  - achieving high encoding quality
  - incorporating intelligent ABR profile generator
- Means for playback, monitoring and end-to-end optimizations
  - optimized clients
  - end-to-end analytics
  - customizable profiles, delivery rules, etc.

One can have much fun building them, or just pick some of the existing solutions.

To try our products, please click:

https://register.brightcove.com/en/video-cloud – VideoCloud

https://zencoder.com/en/ – Zencoder

Thank you!

Contact:
Yuriy A. Reznik
yreznik@brightcove.com