

ENCODING 2018: CODECS & PACKAGING FOR PCS, MOBILE, & OTT/STB/SMART TVS

Jan Ozer

www.streaminglearningcenter.com

[jozer@mindspring.com/](mailto:jozer@mindspring.com)

276-235-8542

@janozer

Agenda

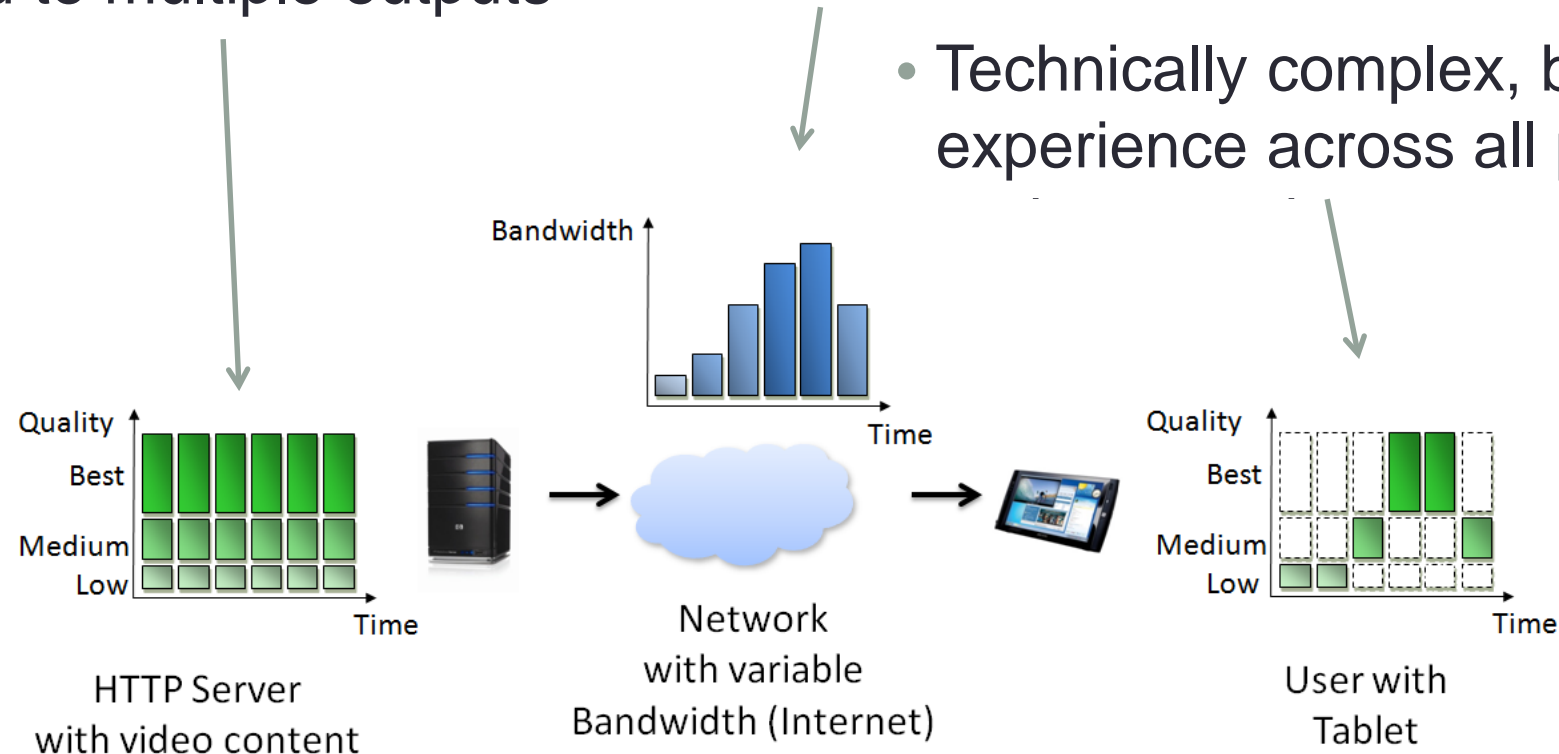
- Introduction
- Lesson 1: Intro to ABR video
- Lesson 2: Choosing an ABR Format
- Lesson 3: Codecs and container Formats
- Lesson 4: Quickie on Manifest Files
- Lesson 5: Intro to Encoding Ladders
- Lesson 6: Intro to Objective Quality Metrics
- Lesson 7: Building Your Encoding Ladder with VMAF/CRF
- Lesson 8: Encoding for ABR
- Lesson 9: Encoding with H.264
- Lesson 10: Encoding with HEVC
- Lesson 11: Dynamic Packaging for VOD and Live

Introduction

- Our goals
 - Happy viewers
 - Happy CFOs
- Happy viewers:
 - High quality video
 - Compatible with device
 - Plays smoothly
- Happy CFOs
 - Efficient to encode
 - Lowest possible bandwidth
 - Lowest possible storage cost
 - Most efficient deliver

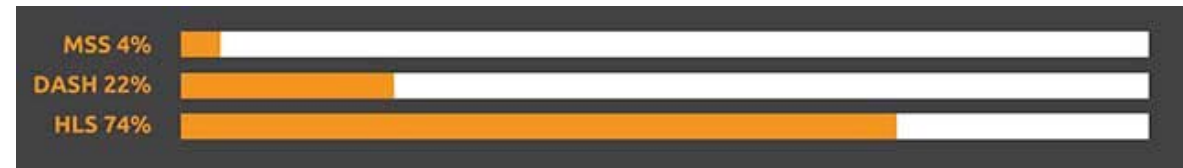
Lesson 1: ABR Formats and How They Work

- Adaptive streaming
 - Single input file (live or VOD)
 - Encoded to multiple outputs
- Delivered adaptively based upon playback CPU and connection bandwidth
 - Technically complex, but optimizes experience across all platforms



ABR Technology Overview

- Two types of systems
 - Server-based (Flash, RTMP)
 - Legacy; on the way out
 - HTTP (most new installations) has various flavors
 - HTTP Live Streaming (HLS)
 - Dynamic Adaptive Streaming over HTTP (DASH)
 - Smooth Streaming (MS game platforms)
 - HTTP-based Dynamic Streaming (HDS)



encoding.com – Global Format
Report
<http://bit.ly/globform18>

Perspective

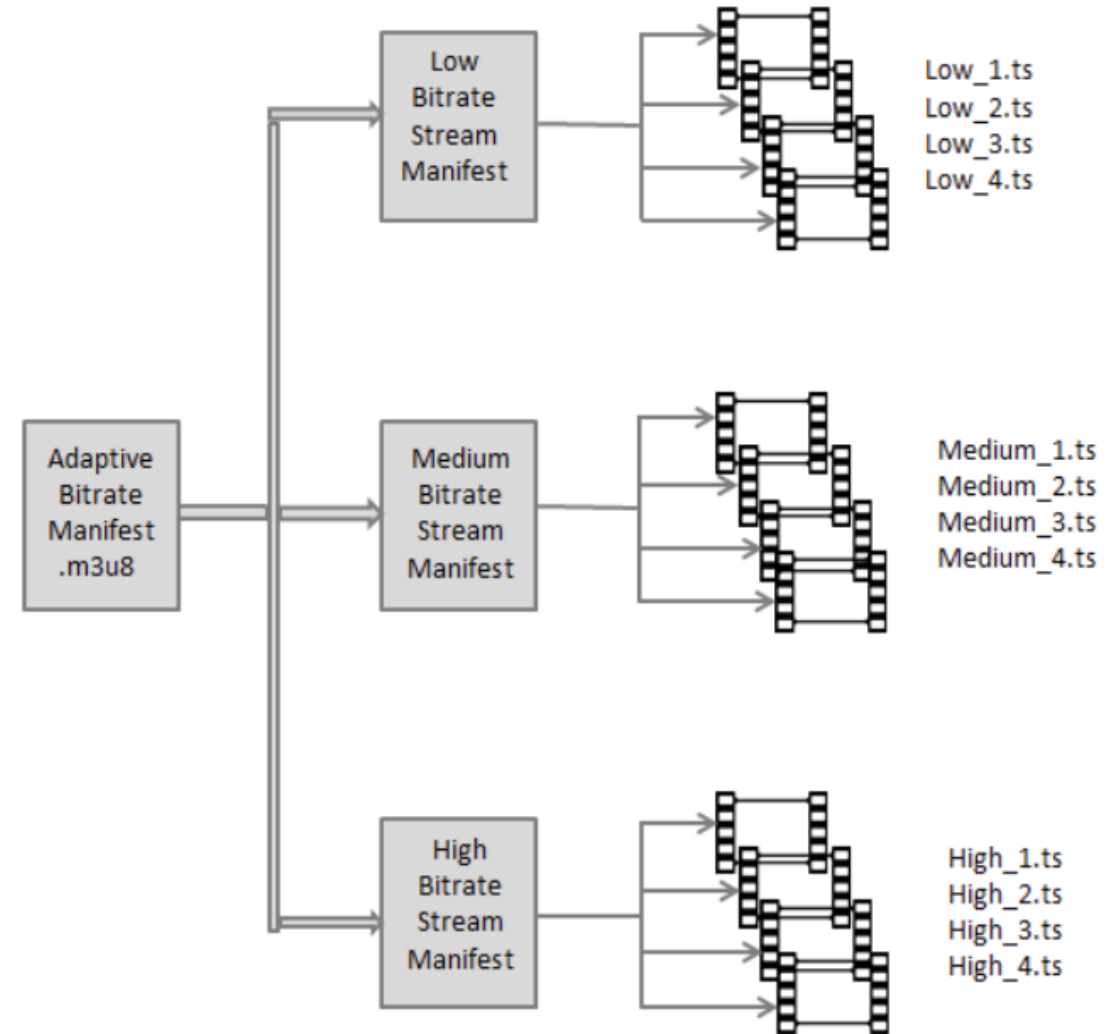
- All HTTP Technologies work the same way
 - Encoding ladder comprised of multiple files
 - Create chunked data files (or discrete segments within longer file)
 - Create index files (also called manifest files) with locations of chunks
 - Uploads all files to HTTP server

16:9 aspect ratio	H.264/AVC	Frame rate
416 x 234	145	≤ 30 fps
640 x 360	365	≤ 30 fps
768 x 432	730	≤ 30 fps
768 x 432	1100	≤ 30 fps
960 x 540	2000	same as source
1280 x 720	3000	same as source
1280 x 720	4500	same as source
1920 x 1080	6000	same as source
1920 x 1080	7800	same as source

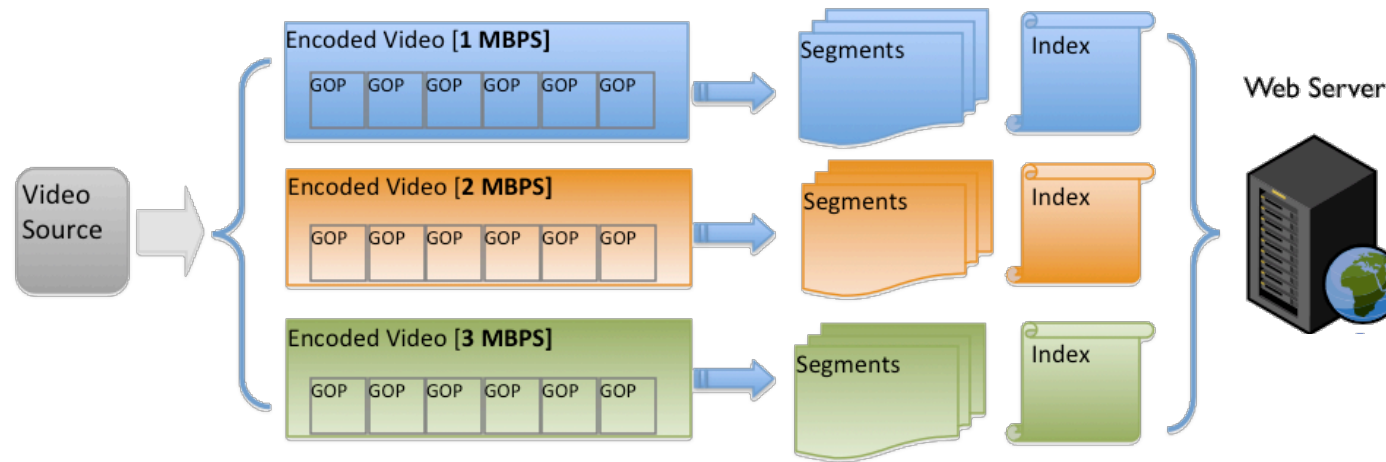
Apple HLS Authoring Specification
http://bit.ly/hls_spec_2017

Perspective

- All HTTP Technologies work the same way
 - Encoding ladder comprised of multiple files
 - Create chunked data files (or discrete segments within longer file)
 - Create index files (also called manifest files) with locations of chunks
 - One master manifest
 - One for each content file
 - Uploads all files to HTTP server

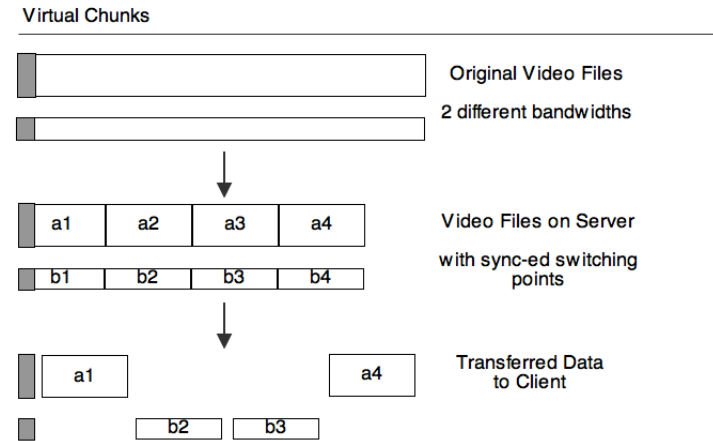
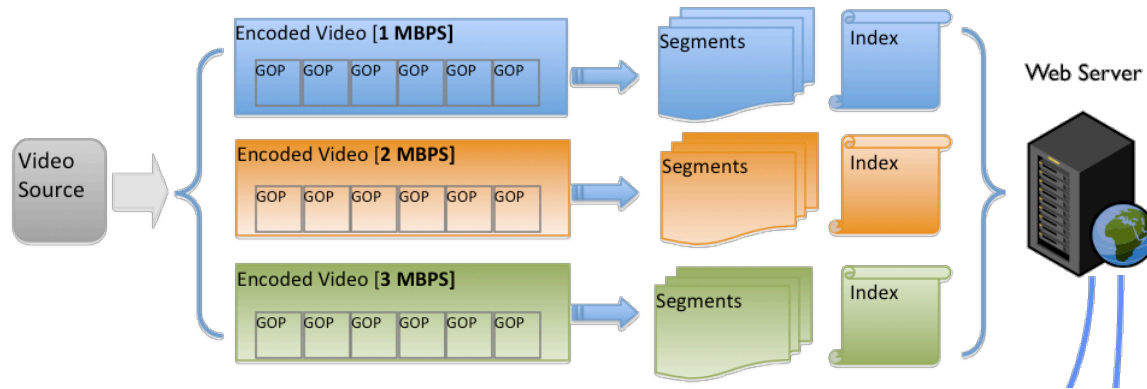


Apple HTTP Live Streaming (HLS)



- Encoder creates:
 - Chunked video files
 - Index files (M3U8) with file descriptions (res/data rate/profile) and chunk URLs
- Uploads to HTTP web server

FILES AND BIT RANGE REQUEST

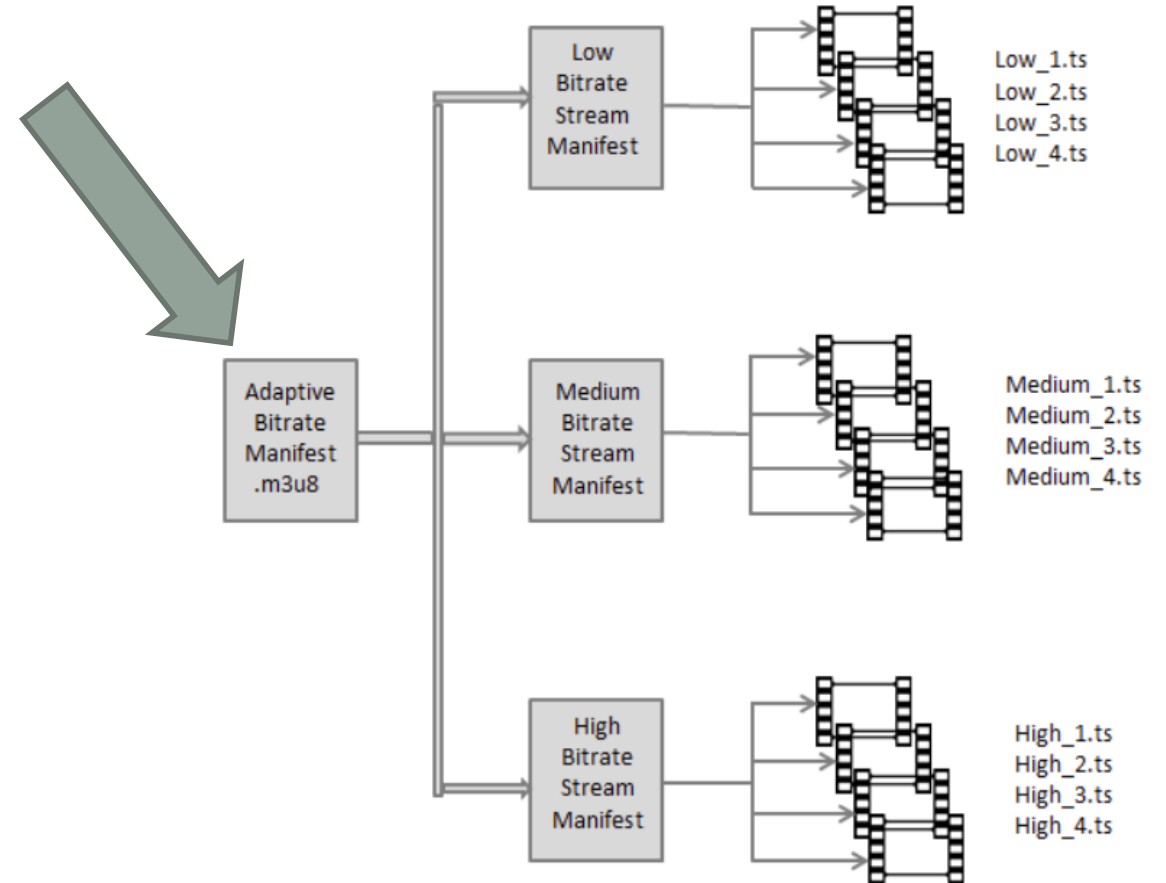


- When HTTP-based ABR started, all content files were split into multiple segments
 - Created administrative nightmare
 - Hundreds of thousands of files for even short videos

- Now all can use “byte range requests” from a single file
 - Upload a long single file per layer with data in the header that identifies the relevant segments
 - MPEG-2 ts for HLS
 - fMP4 for DASH, Smooth Streaming, HDS
- Talk about segments, mean both approaches

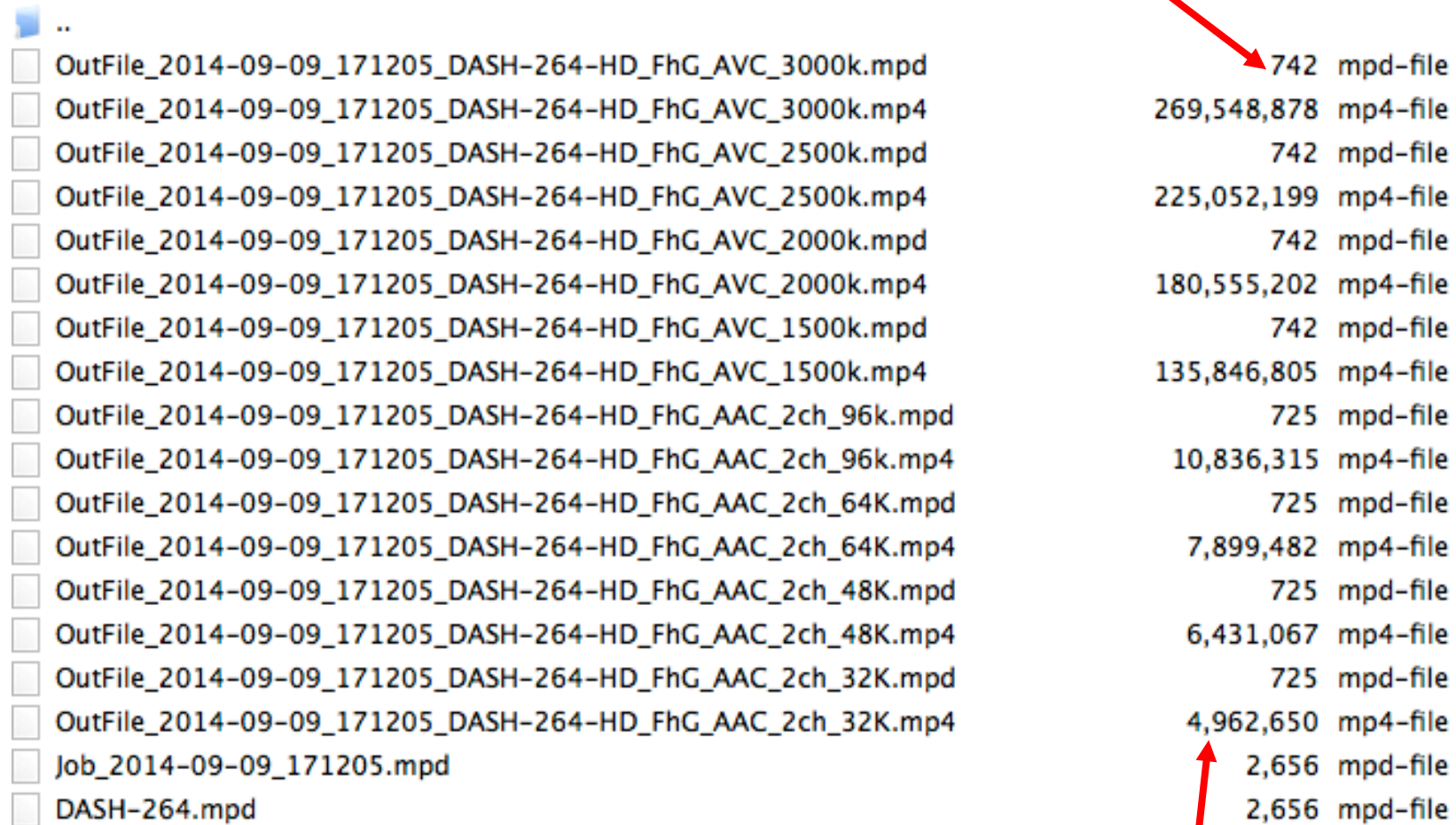
Perspective

- Player side
 - Loads the master manifest file
 - Starts playing first file listed in the master manifest file
 - Monitors playback buffer and (sometimes) CPU use
 - Changes streams as necessary
 - Uses index files to find the right files



DASH

stream (variant) manifest files (.mpd)



..	
OutFile_2014-09-09_171205_DASH-264-HD_FhG_AVC_3000k.mpd	742 mpd-file
OutFile_2014-09-09_171205_DASH-264-HD_FhG_AVC_3000k.mp4	269,548,878 mp4-file
OutFile_2014-09-09_171205_DASH-264-HD_FhG_AVC_2500k.mpd	742 mpd-file
OutFile_2014-09-09_171205_DASH-264-HD_FhG_AVC_2500k.mp4	225,052,199 mp4-file
OutFile_2014-09-09_171205_DASH-264-HD_FhG_AVC_2000k.mpd	742 mpd-file
OutFile_2014-09-09_171205_DASH-264-HD_FhG_AVC_2000k.mp4	180,555,202 mp4-file
OutFile_2014-09-09_171205_DASH-264-HD_FhG_AVC_1500k.mpd	742 mpd-file
OutFile_2014-09-09_171205_DASH-264-HD_FhG_AVC_1500k.mp4	135,846,805 mp4-file
OutFile_2014-09-09_171205_DASH-264-HD_FhG_AAC_2ch_96k.mpd	725 mpd-file
OutFile_2014-09-09_171205_DASH-264-HD_FhG_AAC_2ch_96k.mp4	10,836,315 mp4-file
OutFile_2014-09-09_171205_DASH-264-HD_FhG_AAC_2ch_64K.mpd	725 mpd-file
OutFile_2014-09-09_171205_DASH-264-HD_FhG_AAC_2ch_64K.mp4	7,899,482 mp4-file
OutFile_2014-09-09_171205_DASH-264-HD_FhG_AAC_2ch_48K.mpd	725 mpd-file
OutFile_2014-09-09_171205_DASH-264-HD_FhG_AAC_2ch_48K.mp4	6,431,067 mp4-file
OutFile_2014-09-09_171205_DASH-264-HD_FhG_AAC_2ch_32K.mpd	725 mpd-file
OutFile_2014-09-09_171205_DASH-264-HD_FhG_AAC_2ch_32K.mp4	4,962,650 mp4-file
Job_2014-09-09_171205.mpd	2,656 mpd-file
DASH-264.mpd	2,656 mpd-file

Main manifest file (.mpd)

Content files (.mp4)

Captions and DRM

- Caption formats are specific to each ABR format and are listed in the manifest files
- DRM is handled as part of the final file packaging (more later)

HTTP Adaptive Summary (review)

- All technologies work similarly
 - Chunked or segmented video files
 - Manifest data files
 - HTTP server
 - Player driven operation
- The big differentiating issues are:
 - Where they play
 - Whether they are a standard or proprietary
 - How much they cost (DASH=CASH)

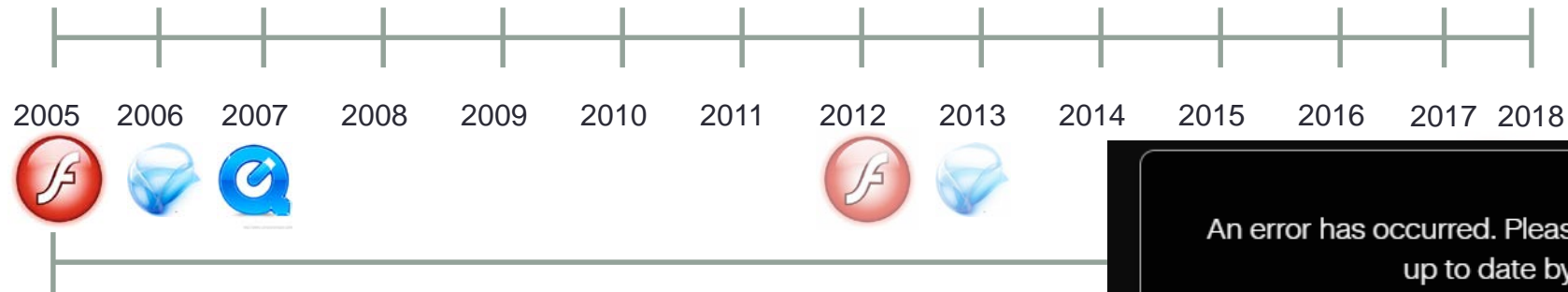
From Plug-ins to HTML: A Retrospective

- HTML5's key benefit
- Where we are today?

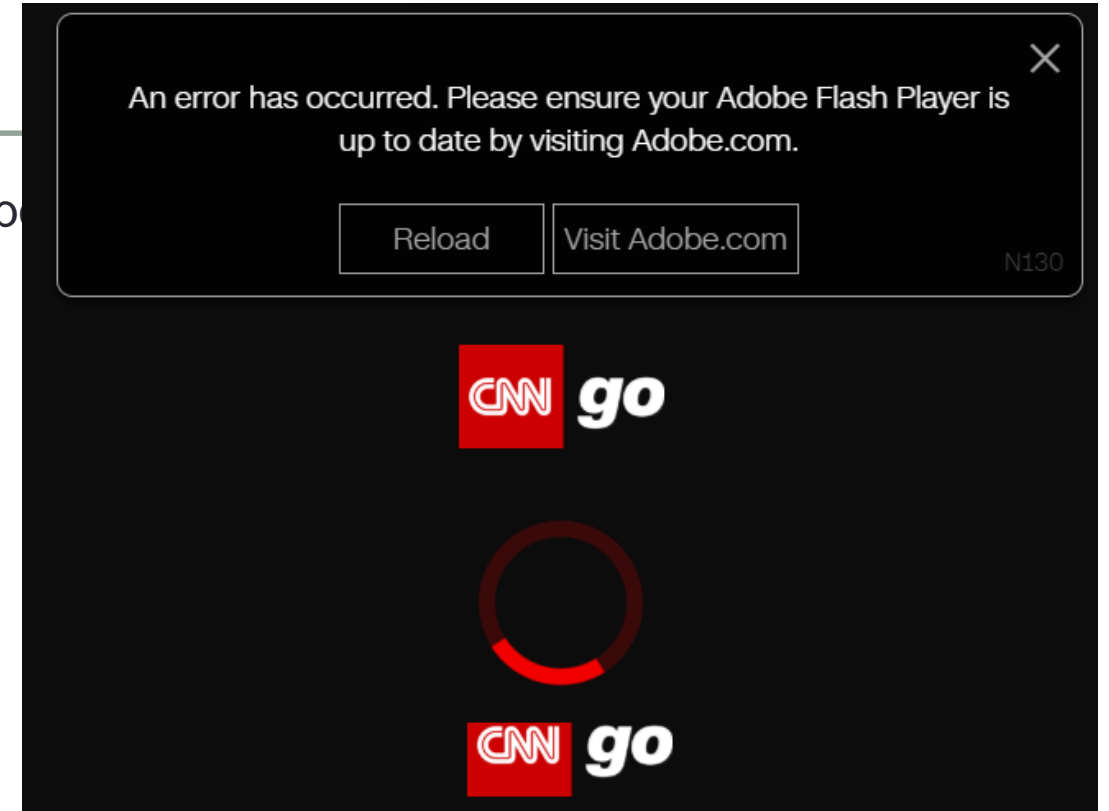
Working in the HTML5 Environment

- HTML5's key benefit
 - Video playback without plug-ins
- How it works
 - Instead of obtaining decoders for H.264 and other codecs from plug-ins like Flash/Silverlight
 - Browsers supply players and decoders
 - Decoders can be in the browser (Chrome, Safari, IE)
 - Decoders can be in the OS (Firefox, Opera)
- You're only as good as the deployed browser
 - Can be a problem for services targeting corporate, government or older viewers (check log files)

HTML5 – Where We Are Today



Plug-in era – primarily used now for advertising support

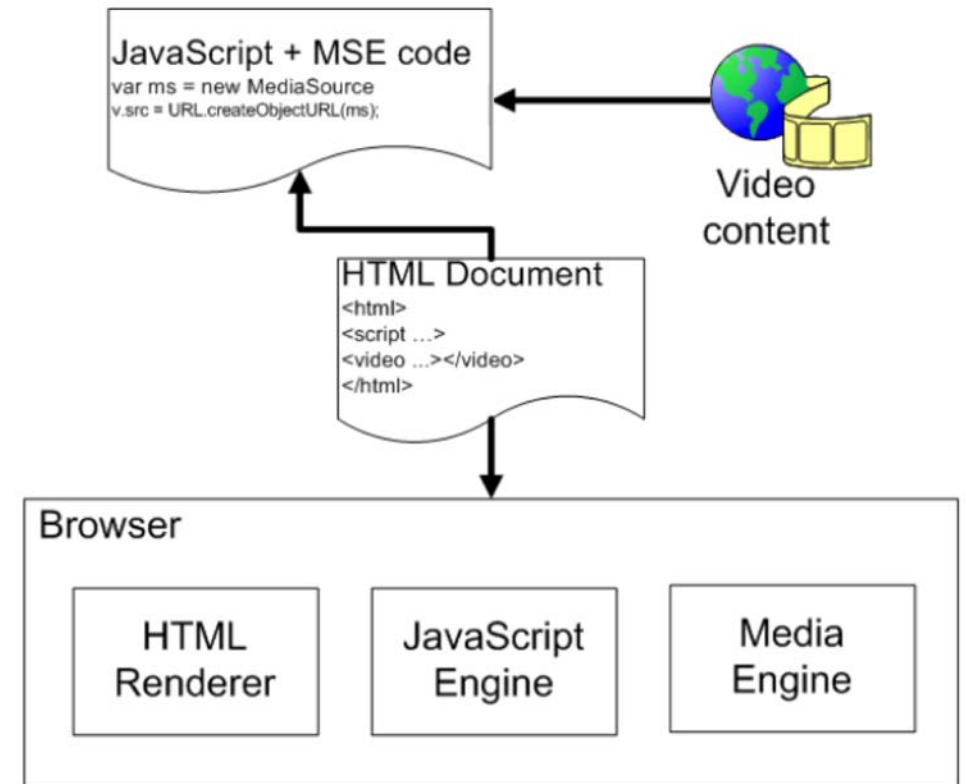


Pieces of the Puzzle

- Media Source Extensions - MSE
- Dynamic Adaptive Streaming over HTTP - DASH
- Encrypted Media Extensions - EME
- ISO-Base Media File Format - BMFF

Media Source Extensions (MSE)

- JavaScript interface to play back media data provided by the JavaScript layer
- A W3C HTML Working Group spec
- More flexible than video tag
 - Media chunks (adaptive) and (closer to) true streaming than progressive
 - Live
 - Better support for captions and DRM (via Encrypted Media Extensions)



What is Dynamic Adaptive Streaming over HTTP (DASH)

- Standardized file format
 - HLS, Smooth, HDS all proprietary
- Like all HTTP-based technologies, it has
 - Fragmented video chunks (or single file with segments)
 - Manifest files
- Now may be subject to a royalty (MPEG-LA)

What is DASH? CASH!



IP History

- MPEG DASH finalized in 2011-2012
- July 2015, MPEG LA announces pool (http://bit.ly/DASH_pool_formed)
- In November 2016, MPEG LA announces license (http://bit.ly/DASH_license)

Analysis and Implications

- This is the first royalty on free internet video
- CNN distributes free video in H264 or HEVC using HLS
 - No royalty
- CNN distributes free video with DASH
 - Royalty on apps and ultimately perhaps browser-based playback
- No exclusions for churches, charities, governments or otherwise
- Really is remarkable in scope

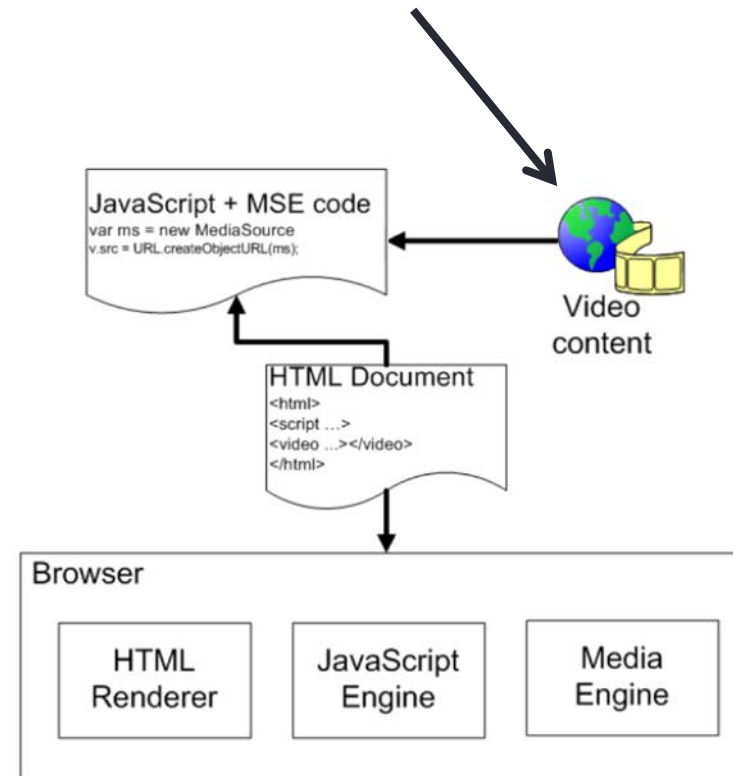
Licensees and Affiliates in Good Standing*

1. AbemaTV, Inc.
2. Abox42 GmbH
3. Agama Technologies AB
4. AVerMedia Technologies, Inc.
5. CBS Interactive Inc.
6. Fluendo S.A.
7. Hama GmbH & Co KG
8. Seiki Corporation
9. Telekom Deutschland GmbH

DASH and MSE

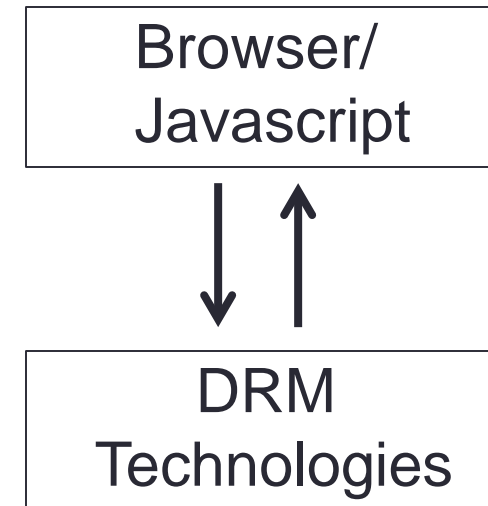
- DASH is one of the file formats MSE expects
- Can write JavaScript code enabling MSE to play HLS and other ABR formats
 - Very common among off the shelf players

DASH, HLS or other ABR technologies



Encrypted Media Extensions (EME)

- JavaScript API
 - Enables HTML5-based digital rights management (DRM)
 - Extends MSE by providing APIs to control playback of protected content.
- License/key exchange is controlled by the browser
 - Not a plug-in



The Problem Is – No Universal DRM

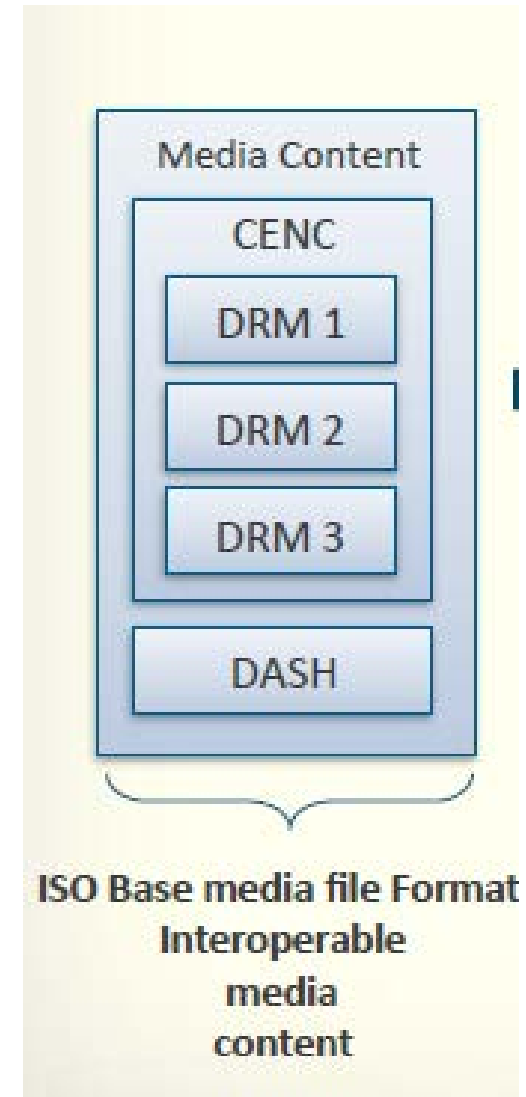
HTML5 Browsers	PlayReady	Widevine MODULAR	Widevine CLASSIC	FairPlay	Primetime (ACCESS)
Chrome (35+)	✗	✓	✗	✗	✗
Firefox (38+ on Windows)	✗	✗	✗	✗	✓
Firefox (47+ on Windows & Mac) ¹	✗	✓	✗	✗	✓
Internet Explorer (11+ on Windows 8.1+)	✓	✗	✗	✗	✗
Microsoft Edge (Windows 10+)	✓	✗	✗	✗	✗
Opera (31+)	✗	✓	✗	✗	✗
Safari (8+ on OS X)	✗	✗	✗	✓	✗

<https://drmtoday.com/platforms/>

- MS browser and mobile – PlayReady
 - Google browser, Android and devices – Widevine
 - Apple browser/devices – FairPlay
 - Firefox – Primetime/Widevine
- So, you need multiple DRMs to distribute to multiple platforms

It's OK from a File Creation Standpoint

- Using MPEG DASH (a media format) plus CENC (Common Encryption Scheme),
- Single adaptive group of files *can contain multiple DRM key technologies*



But You'll Need a Multi-DRM Service Provider

- Adobe Primetime DRM
- Azure
- BuyDRM
- Cisco VideoGuard Everywhere
- DRM Today
- EZDRM
- ExpressPlay
- Verimatrix
- Vualto DRM
- One or more DRMs added during encoding/packaging
- More on this throughout the presentation

Questions?

- Questions

Should be 9:30

Lesson 2: Choosing an ABR Format

- Computers
- Mobile
- OTT
- Smart TVs

Choosing an ABR Format for Computers

- Can be DASH or HLS
- Factors
 - Off-the-shelf player vendor (JW Player, Bitmovin, THEOPlayer, etc.)
 - Encoding/transcoding vendor

Choosing an ABR Format for iOS

- Native support (playback in the browser)
 - HTTP Live Streaming
- Playback via an app
 - Any, including DASH, Smooth, HDS or RTMP Dynamic Streaming

Choosing an ABR Format for Android

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.7%
4.2.x		17	2.2%
4.3		18	0.6%
4.4	KitKat	19	10.5%
5.0	Lollipop	21	4.9%
5.1		22	18.0%
6.0	Marshmallow	23	26.0%
7.0	Nougat	24	23.0%
7.1		25	7.8%
8.0	Oreo	26	4.1%
8.1		27	0.5%

Data collected during a 7-day period ending on April 16, 2018.

Any versions with less than 0.1% distribution are not shown.

Codecs

VP8 (2.3+)↓

H.264 (3+)↓

VP9 (4.4+)↓

HEVC (5+)↓

ABR

HLS (3+) ↓

DASH 4.4+
Via MSE ↓
in Chrome

- Multiple codecs and ABR technologies
 - Serious cautions about HLS
 - **DASH now close to 95%**

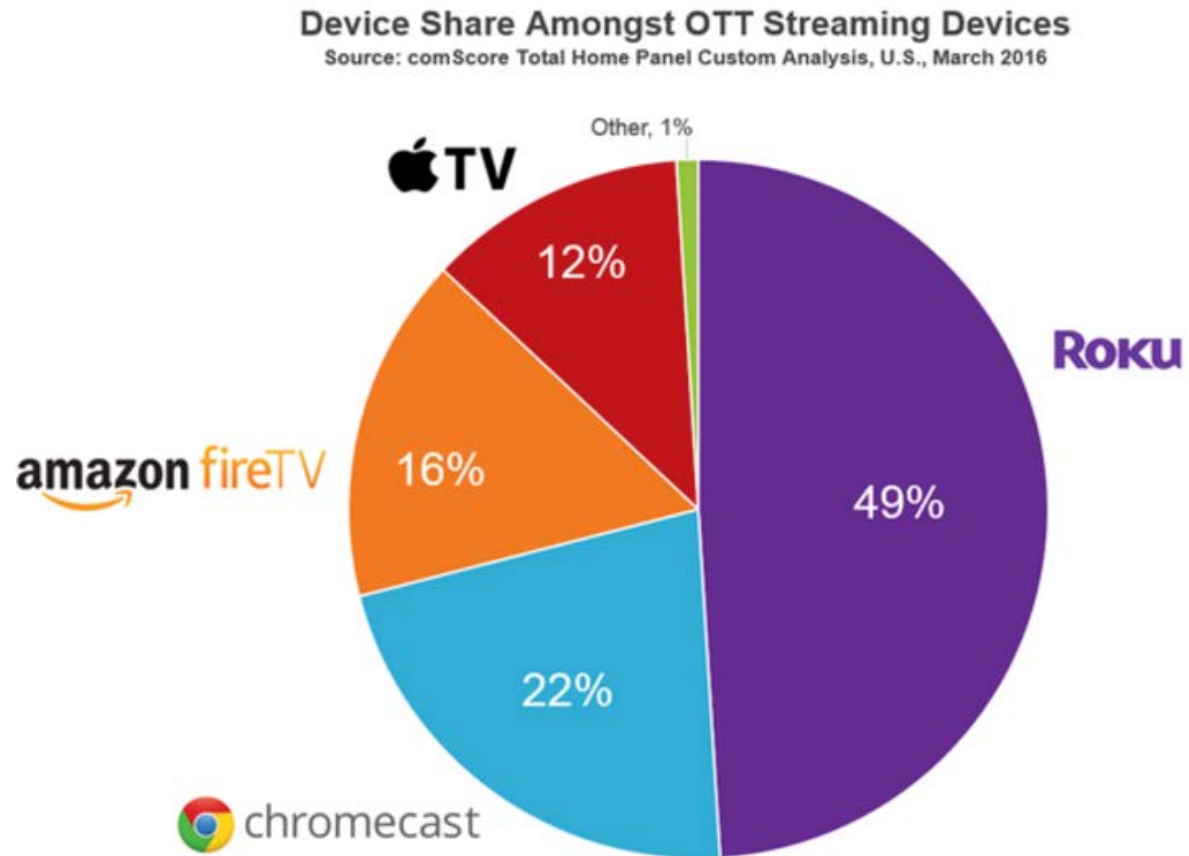
<http://bit.ly/androidvideospecs>

http://bit.ly/And_ver

Adaptive Streaming to OTT

- Format support – general
- Roku
- Apple TV
- Chromecast
- Amazon Fire TV
- PS3/PS4
- Xbox 360/Xbox One

Who Matters?



http://bit.ly/mar_16_ott

OTT Platform-Format Support

Platform	Smooth Streaming	HLS	DASH
OTT Platforms			
Roku (bit.ly/encode_roku)	Yes	Yes	Yes
Apple TV (bit.ly/AppleTV_recs)	No	Yes	No
ChromeCast (bit.ly/Chromecast_media)	Yes	Yes	Yes
Amazon Fire TV (bit.ly/Firetv_media)	Yes	Yes	Yes

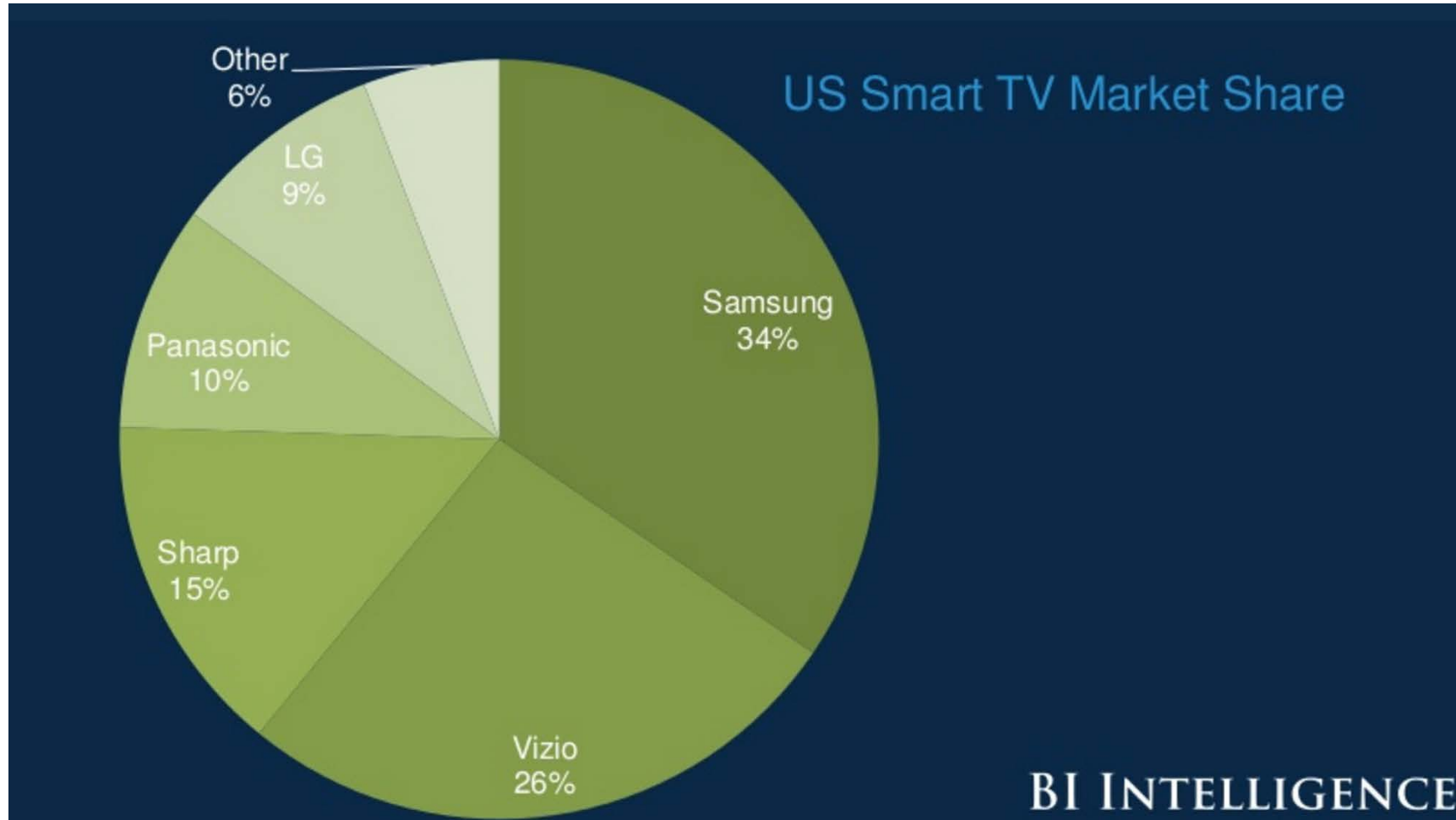
Notes:

- Roku 4 and Roku4 TVs supports HEVC and VP9
- Fire TV Gen 2 supports HEVC
- Fire TV Supports VP9
- Most recent Apple TV specs do support CMAF

Adaptive Streaming to Smart TVs

- Format support – general
- Samsung
- Vizio
- Sharp
- Panasonic
- LG
- Smart TV Alliance
- HbbTV

Who Matters?



Samsung Format Support

- Samsung changed platforms in 2015 to Tizen
 - Old specs - bit.ly/Samsung_oldspec
 - Tizen – spec - bit.ly/Tizen_spec
 - On Tizen, features differ for native or SDK

	Tizen TV	SDK 2.0
codecs	HEVC, H.264, VP8, VP9	
Streaming formats	MPEG-DASH, HLS, Smooth	MPEG-DASH, HLS, Smooth
DRM	PlayReady, Widevine, AES128, Verimatrix, SecureMedia	
Captions	SMPTE-TT, DFXP	SMPTE-TT, DFXP

Vizio Format Support - ?

- Data not publicly available

Sharp Format Support -?

- Data not publicly available

Smart TV Alliance

- Members
 - Panasonic, LG, Toshiba
- Spec – 5.0 (9/2015)
- Codecs
 - H.264, HEVC
- ABR formats (**M**=mandatory)
 - MPEG DASH, Smooth Streaming, HLS
- DRM
 - PlayReady, Widevine
- Captions
 - W3C TTML

Function	Detail	A/V content
General	HTTP 1.1 with Range request	M
	HTTPS streaming over SSL	M
Adaptive	HTTP Live Streaming	M
	Microsoft Smooth Streaming	M
	MPEG-DASH (ISOBMFF & CENC) according to HbbTV version 1.2.1 profile [26]	M

HbbTV 2.01 – 4/16/2016

- Codecs
 - H.264, HEVC
- ABR formats
 - DASH
- DRM
 - CENC
- Captions
 - W3C TTML

HTTP adaptive streaming shall be supported using MPEG DASH as defined in annex E.

Questions?

- Questions

-

Should be 9:40

Lesson 3: Codecs and Container Formats

- Choosing a codec
 - Heritage/Cost
 - Playback
 - Quality
 - Encoding time
 - Playback performance
- Choosing a container format
 - Transmuxing (converting from one container to another)

Heritage/Cost

	H.264	HEVC	VP9	AV1	PERSEUS	RealMedia HD
Heritage	Standards-based	Standards-based	Google	Alliance for Open Media	V-Nova	Real Networks
Cost – free streaming	None	None	None	None	?	?
Cost – PPV/Subscription	Royalty	Uncertain	None	None	?	?
Cost - hardware	Up to \$9.75 million cap	\$60 million+ annual cap*	None	None	?	?
Cost – software player	Up to \$9.75 million cap (total/year)	Same	None	None	?	?

*Includes only two of three known royalty groups

Choosing a Codec – First it Must Play

- Codec – stands for enCOde/DEcode
 - Need the decode side to play the video
- Which platforms have decoders?

	Computer/ Notebook	iOS	Android	Retail OTT (Roku, Apple TV)	Smart TV
H.264	Yes	Yes	Yes	Yes	Yes
HEVC	MacOS/Windows 10 with h/w and Edge	Yes	Version 5+	Most	All 4K
VP9	Chrome, Firefox, Opera, Edge	Not Yet	Version 4	Most (not Apple TV)	Most Newer
AV1	Will have soon	2020	2020	2020	2020



Codec Quality

- HEVC and VP9 are roughly the equivalent
 - Close enough so that it's not a relevant decision factor
- AV1 is up to 30% more efficient than HEVC/VP9

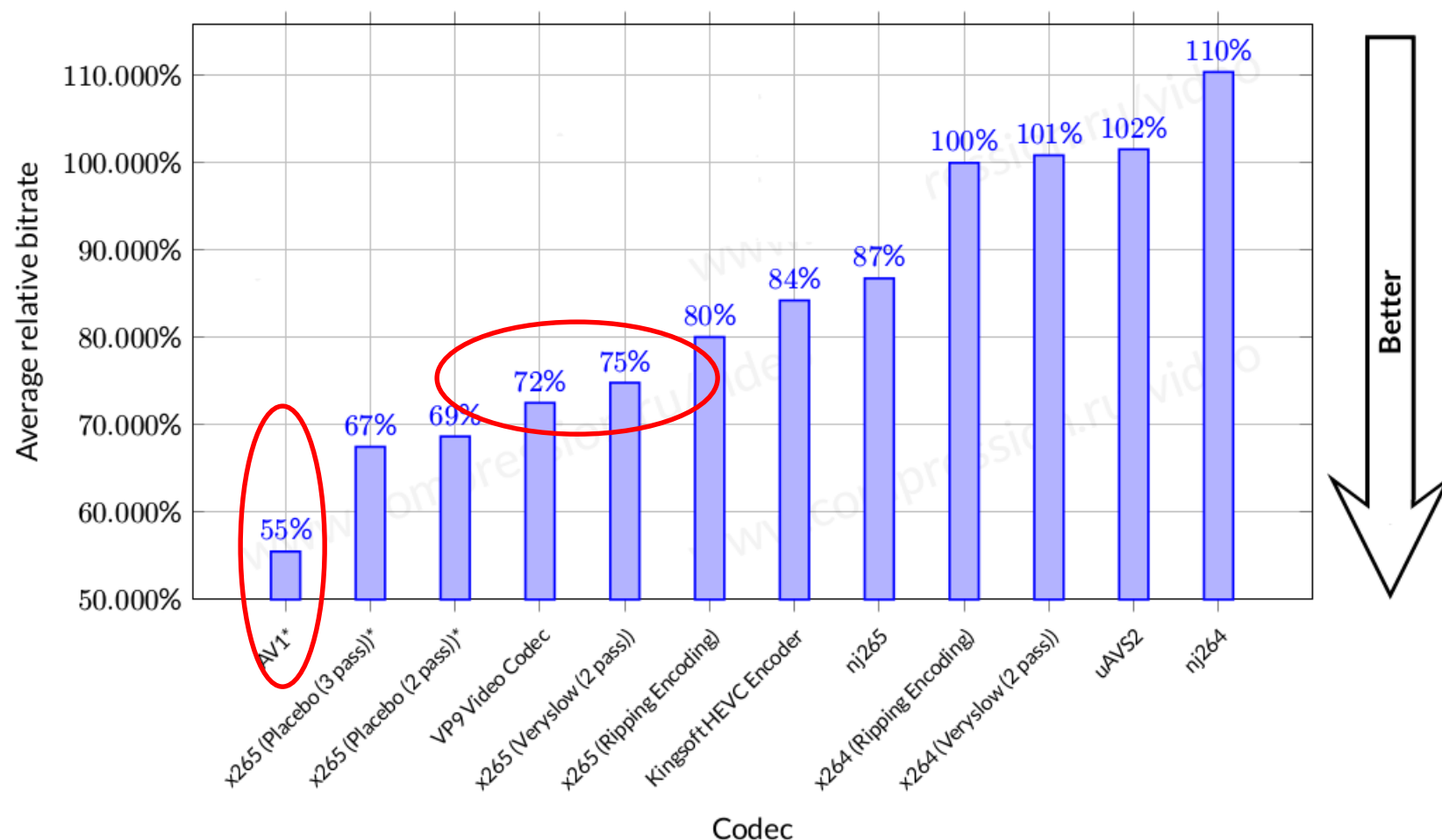


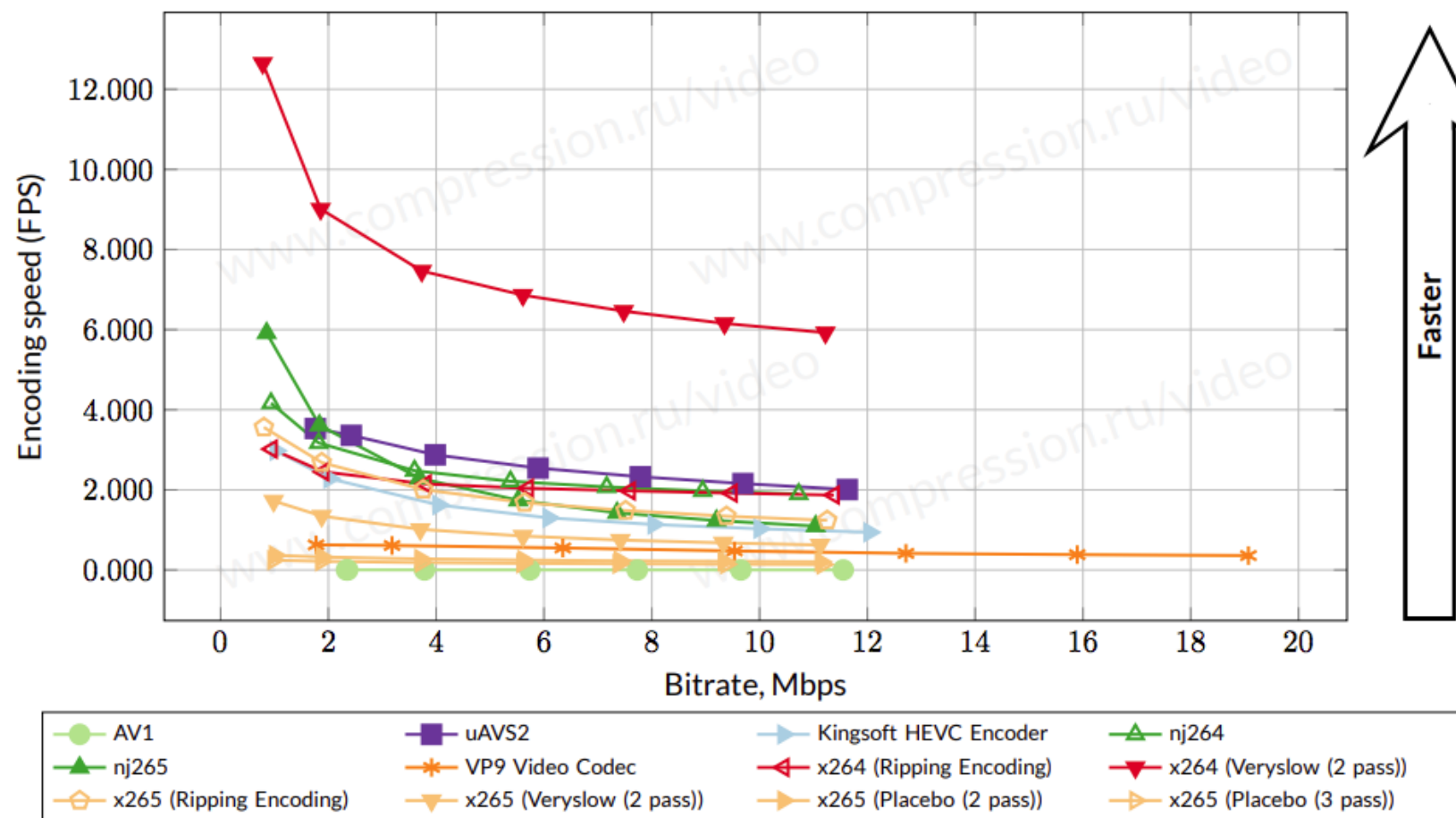
Figure 13: Average bitrate ratio for a fixed quality—use case “Ripping Encoding,” all sequences, YUV-SSIM metric.

Encoding Speed

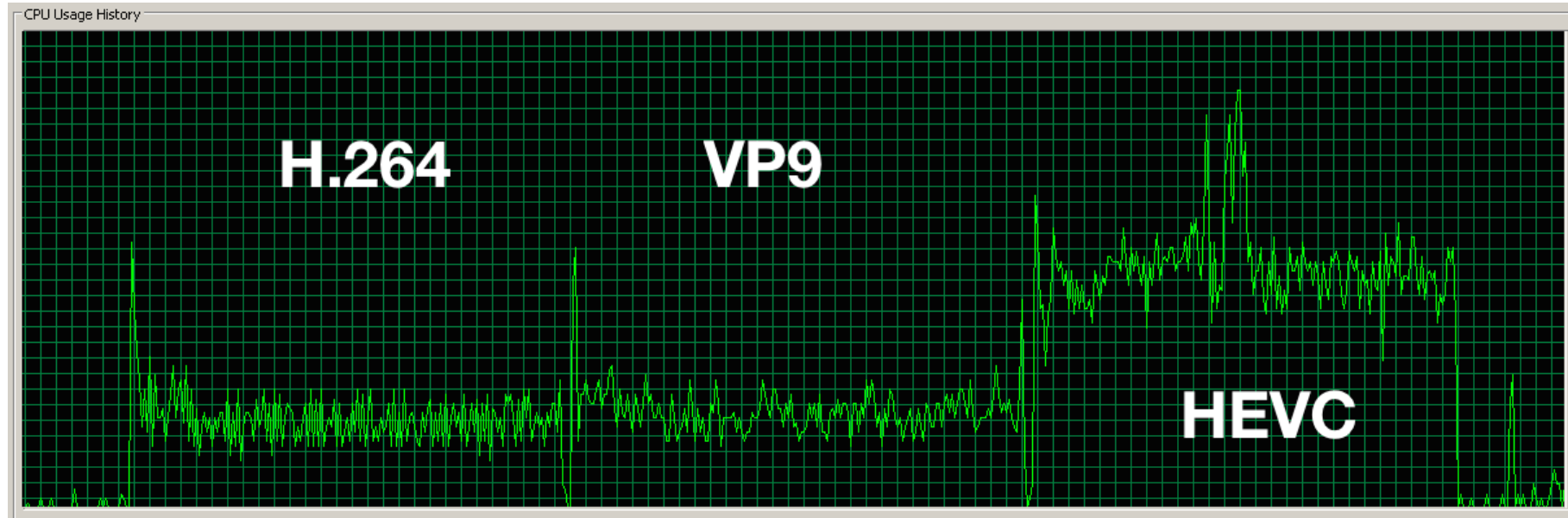
- HEVC is slower than VP9, but it's system dependent
- Both are much slower than H.264
- AV1 is glacial “2500 – 3000 times slower than competitors”

5. ENCODING SPEED

Figures below show difference in encoding speed among participating codecs. AVS2 encoder shows better encoding speed comparing to other encoders. AV1 encoder has extremely low speed – 2500-3000 times lower than the competitors. X265 Placebo presets (2 and 3 passes) have 10-15 times lower speed than the competitors.



Decode CPU



- Software-only playback on 2006 era Dell workstation
- AV1 is estimated to require 2.5 – 3x more CPU than VP9
- Most battery-powered devices (where higher CPU load decreases battery life) have hardware HEVC/VP9/H.264 decode
 - So, all three have a very significant advantage over AV1 until devices with hardware decode arrive (2020)

AV1 Summary

- Quality is alluring, but
 - Encoding cost will be 5-9x VP9 for the foreseeable future
 - Still makes sense if your videos are watched by millions (Netflix, YouTube, Hulu, etc)
 - Not for dozens or even hundreds of thousands of views
 - Lack of hardware decode also makes it a non-starter on mobile devices (narrowing market for potential viewers even further)
 - Check back in a year

PERSEUS and RealMedia HD

- PERSEUS

- Can field upgrade H.264 STBs and OTT devices to HEVC-like performance
 - Sky Italia
- Very good low bit rate performance
 - Fast Filmz (India)
- Has made great strides towards usability (encoding, playback compatibility)
- Big company play; not for the average user

- RealMedia HD

- Is aiming to out AV1 AV1
 - Same or better performance than HEVC, much faster encode, much lower CPU decode
- Haven't tested
- Also big company play

2017 Numbers from encoding.com



encoding.com 2017 Global
Media Format Report
http://bit.ly/ecmf_2017

- Files produced by their customers
 - Big media companies, but not Netflix, YouTube, Hulu, etc.
- H.264 still king (*increased* by 2%)
- HEVC up but still in trial phase
- VP9 down from 11%

Changing Codecs is a Big Deal

- While bandwidth savings are alluring:
 - Still need to encode to H.264 for legacy targets, so encoding and storage costs are additive
 - New codecs reduce caching benefits in distribution infrastructure
- The most attractive option is adding HEVC to HLS, but that's been slow to develop
 - 2018 could be the year
- Per-title encoding (covered later) delivers many of the same benefits without need to change infrastructure

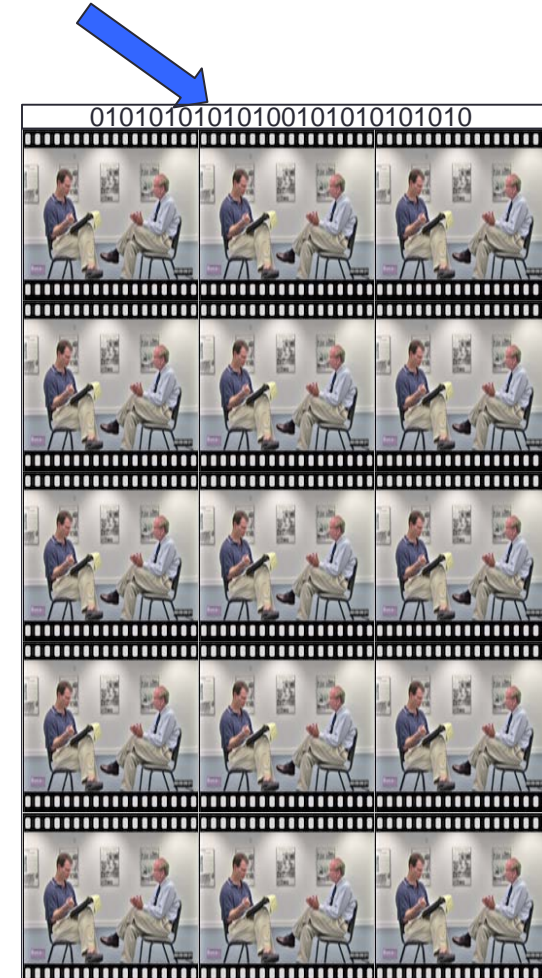
Codecs and Container Formats

- **Codecs:** Compression technologies
 - H.264, VP9, HEVC, AV1
- **Container formats**
 - Specs detailing how data/metadata are stored in a file
 - MP4 (DASH), .ts (HLS), .ISMV (Smooth), .F4F (HDS), FLV (Flash)
 - Also called “wrappers”
 - As in, “encoded the file using the H.264 codec in a QuickTime wrapper”
- **Why important?**
 - File must be in proper container format to play on target platforms

Where is Container Format?

- Text in the file header
 - Very small percentage of overall content
- Can quickly change the container format without affecting A/V content
 - Called transmuxing
 - Critical to operation of tools like Wowza Streaming Engine

File Header



Solving the Multiple Format Problem

- HLS (traditionally) needed MPEG-2 transport streams
 - .ts files
 - Now can use fMP4 as well
- DASH uses fMP4
- So, needed two file groups of files, one for HLS (desktop, mobile), one for DASH (OTT, Smart TVs)



.ts



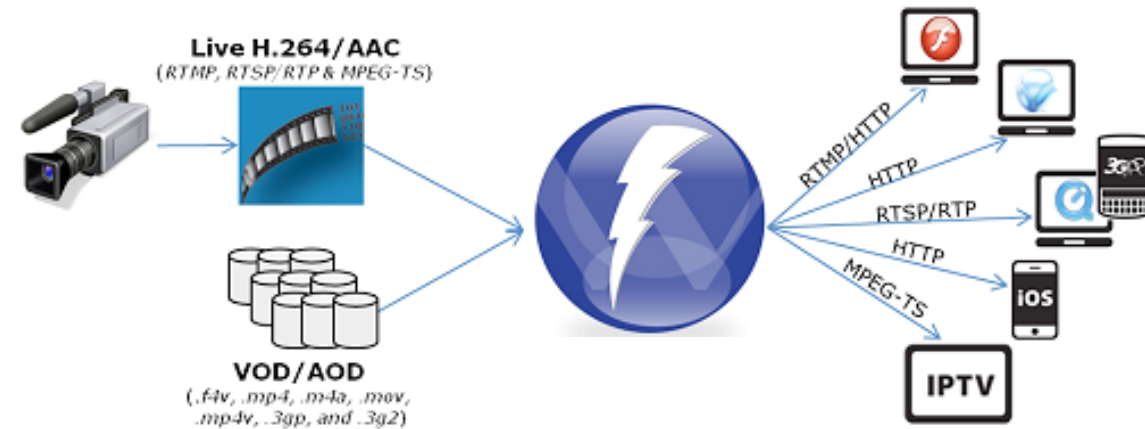
.mp4

- Double encoding cost and storage cost



Solution 1: Transmuxing

- Single format streams in
 - Live or VOD
- Multiple format output streams customized for target
 - Why so fast and efficient?
 - Just adjusting file header
 - Not changing compressed video data at all
- Issues
 - Need server component (Wowza/Nimble Streamer)
 - Cloud computers 24/7 which gets pricey (much more later)



Solution 2: Common Media Application Format (CMAF)

- CMAF

- Apple announcement June 2016
- HLS can use fMP4 files and .ts
- But, two incompatible encryption schemes
 - CBC (Cipher Block Chaining-Apple) and CTR (Counter Mode-everyone else)
 - Still need two copies of content

Before CMAF



DASH



HLS

After CMAF

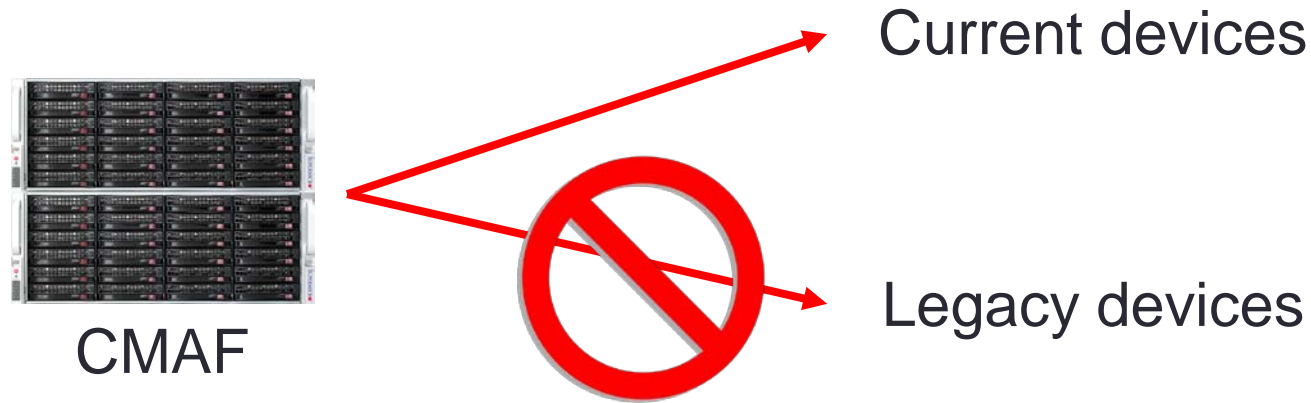


CMAF/CBC



CMAF/CTR

Solution 2: Common Media Application Format (CMAF)



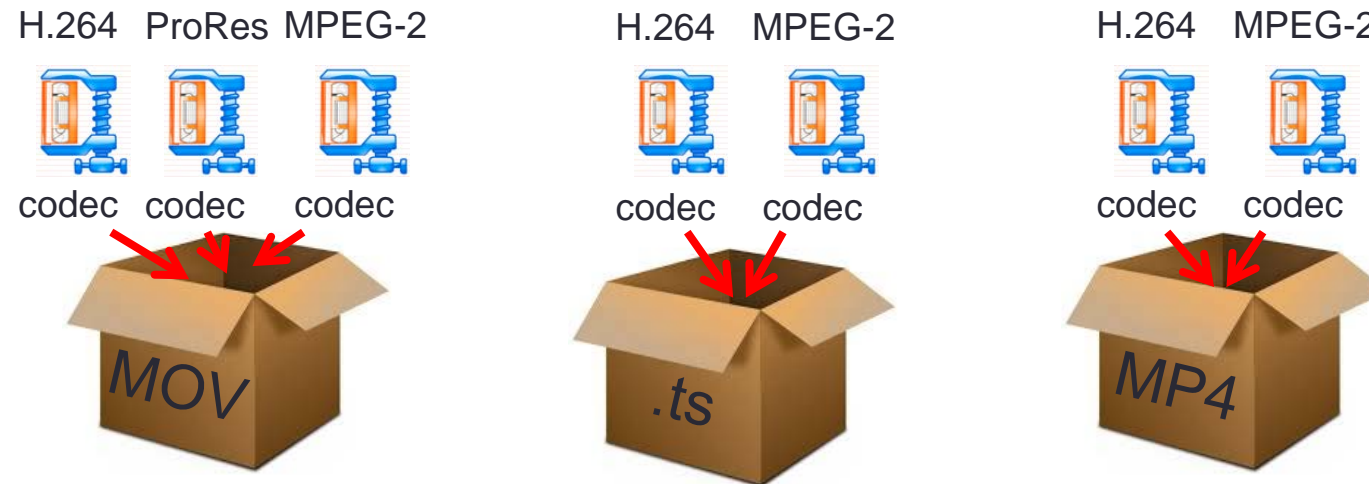
- Since then
 - Google supports CBC in Widevine
 - Playready to support CBC in PlayReady 4
 - <https://www.microsoft.com/playready/newsroom/>
 - So, one set of files deliverable to HLS and DASH clients in 2018
- But: Many legacy HLS devices are incompatible with fMP4
 - Unless you're only serving only the newest clients will either need separate files (.ts for HLS/fMP4 for DASH) or a transmux solution (more later)

Bottom Line on CMAF

- CMAF is very useful, but not a current panacea except for services that exclude older devices

Key Point on Container Formats

- Separate and distinct from choice of codec
 - Can store MPEG-2 compressed video in MP4 file
 - Can store H.264 video in MPEG-2 transport stream



- Whenever you configure encoder for streaming, be aware of selected codec **and** container format

Questions?

- Questions

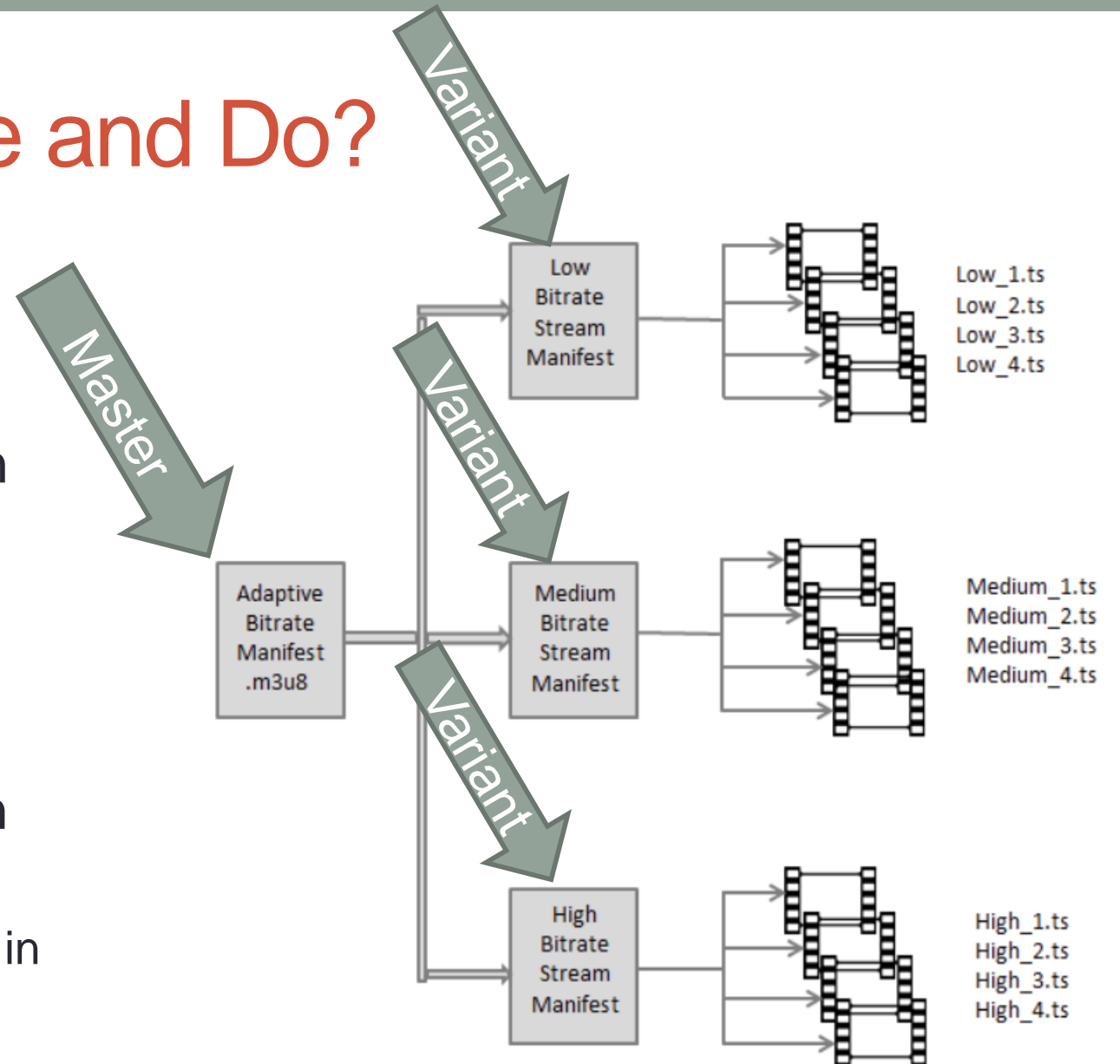
Should be 9:50

Lesson 4: A Quickie on Manifest Files

- What manifest files are and do
- A look inside the master (variant) manifest
- A look inside a media manifest
- Creating the manifest files
- Transmuxing the manifest files

What Manifest Files Are and Do?

- Manifest files are text files:
 - Also called playlists
 - The **master** identifies the location of all content associated with the presentation
 - Video, audio, captions, etc.
 - Is also the file linked to in the Player
 - Each piece of content has its own manifest file (**media playlist**)
 - Contains the addresses of all content in that stream
 - Separate files or byte range requests



A Look Inside the Master Playlist

Bandwidth, resolution, and
codec data so player can
choose the right stream

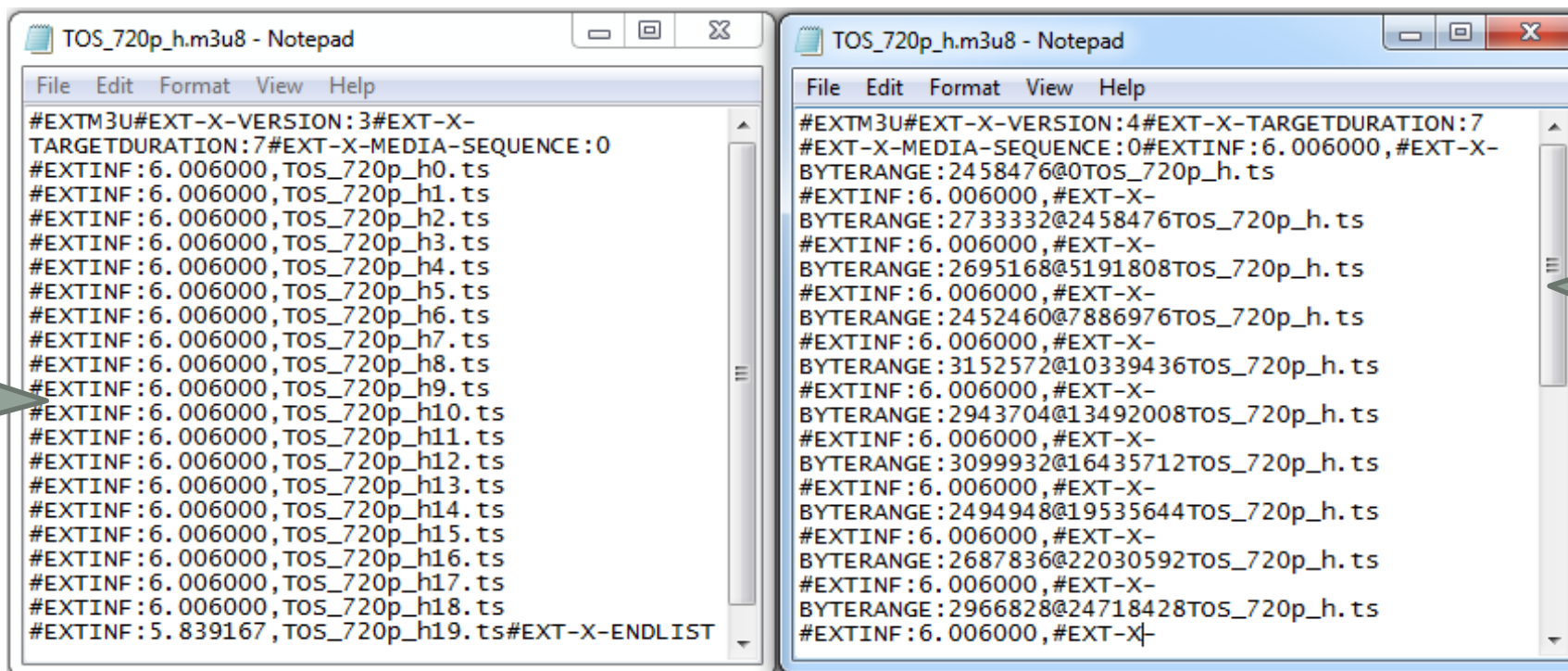
Video variants

Audio variant

```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=174000, RESOLUTION=512x288, CODECS="avc1.42001f, mp4a.40.2"
stream-1-110000/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=294000, RESOLUTION=512x288, CODECS="avc1.42001f, mp4a.40.2"
stream-2-230000/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=544000, RESOLUTION=512x288, CODECS="avc1.42001f, mp4a.40.2"
stream-3-480000/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=1063900, RESOLUTION=640x360, CODECS="avc1.42001f, mp4a.40.2"
stream-4-990000/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=2764000, RESOLUTION=852x480, CODECS="avc1.4d001f, mp4a.40.2"
stream-5-1800000/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=4564000, RESOLUTION=1280x720, CODECS="avc1.4d001f, mp4a.40.2"
stream-6-3000000/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=64000, CODECS="mp4a.40.2"
stream-0-64000/index.m3u8
```

Relative URL for file location

A Look Inside the Media Playlist



```
TOS_720p_h.m3u8 - Notepad
File Edit Format View Help
#EXTM3U#EXT-X-VERSION:3#EXT-X-TARGETDURATION:7#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:6.006000,TOS_720p_h0.ts
#EXTINF:6.006000,TOS_720p_h1.ts
#EXTINF:6.006000,TOS_720p_h2.ts
#EXTINF:6.006000,TOS_720p_h3.ts
#EXTINF:6.006000,TOS_720p_h4.ts
#EXTINF:6.006000,TOS_720p_h5.ts
#EXTINF:6.006000,TOS_720p_h6.ts
#EXTINF:6.006000,TOS_720p_h7.ts
#EXTINF:6.006000,TOS_720p_h8.ts
#EXTINF:6.006000,TOS_720p_h9.ts
#EXTINF:6.006000,TOS_720p_h10.ts
#EXTINF:6.006000,TOS_720p_h11.ts
#EXTINF:6.006000,TOS_720p_h12.ts
#EXTINF:6.006000,TOS_720p_h13.ts
#EXTINF:6.006000,TOS_720p_h14.ts
#EXTINF:6.006000,TOS_720p_h15.ts
#EXTINF:6.006000,TOS_720p_h16.ts
#EXTINF:6.006000,TOS_720p_h17.ts
#EXTINF:6.006000,TOS_720p_h18.ts
#EXTINF:5.839167,TOS_720p_h19.ts#EXT-X-ENDLIST

TOS_720p_h.m3u8 - Notepad
File Edit Format View Help
#EXTM3U#EXT-X-VERSION:4#EXT-X-TARGETDURATION:7
#EXT-X-MEDIA-SEQUENCE:0#EXTINF:6.006000,#EXT-X-BYTERANGE:2458476@0TOS_720p_h.ts
#EXTINF:6.006000,#EXT-X-BYTERANGE:2733332@2458476TOS_720p_h.ts
#EXTINF:6.006000,#EXT-X-BYTERANGE:2695168@5191808TOS_720p_h.ts
#EXTINF:6.006000,#EXT-X-BYTERANGE:2452460@7886976TOS_720p_h.ts
#EXTINF:6.006000,#EXT-X-BYTERANGE:3152572@10339436TOS_720p_h.ts
#EXTINF:6.006000,#EXT-X-BYTERANGE:2943704@13492008TOS_720p_h.ts
#EXTINF:6.006000,#EXT-X-BYTERANGE:3099932@16435712TOS_720p_h.ts
#EXTINF:6.006000,#EXT-X-BYTERANGE:2494948@19535644TOS_720p_h.ts
#EXTINF:6.006000,#EXT-X-BYTERANGE:2687836@22030592TOS_720p_h.ts
#EXTINF:6.006000,#EXT-X-BYTERANGE:2966828@24718428TOS_720p_h.ts
#EXTINF:6.006000,#EXT-X-
```

Separate Files

Byte-range requests in the same file

Uploading and Positioning on Server

- Upload all files to a folder on an HTTP webserver
 - Master goes in the root folder
 - Each subfolder has media playlist
 - All links relative so you can place anywhere

The screenshot displays a file manager window with a table of files and folders. A red arrow points from the text 'Master .m3u8' to the file 'ZOOLANDER_1080p.m3u8' in the file list. Another red arrow points from the same text to the 'ZOOLANDER_1080p.m3u8 - Notepad' window, which shows the content of the master playlist file.

Filename	Filesize	Filetype	Last modified	Permissions	Owner/Gro...
..					
ZOOLANDER_1080p.m3u8	897	M3U8 File	11/8/2016 11:41:39...	adfrw (0644)	14 50
stream-6-3000000		File folder	11/8/2016 11:42:31...	flcdmpe (0...	14 50
stream-5-1800000		File folder	11/8/2016 11:42:26...	flcdmpe (0...	14 50
stream-4-990000		File folder	11/8/2016 11:42:15...	flcdmpe (0...	14 50
stream-3-480000		File folder	11/8/2016 11:42:07...	flcdmpe (0...	14 50
stream-2-230000		File folder	11/8/2016 11:41:59...	flcdmpe (0...	14 50
stream-1-110000		File folder	11/8/2016 11:41:52...	flcdmpe (0...	14 50
stream-0-64000		File folder	11/8/2016 11:41:46...	flcdmpe (0...	14 50

Master .m3u8

ZOOLANDER_1080p.m3u8 - Notepad

```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=174000, RESOLUTION=512x288, CODECS="avc1.42001f, mp4a.40.2"
stream-1-110000/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=294000, RESOLUTION=512x288, CODECS="avc1.42001f, mp4a.40.2"
stream-2-230000/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=544000, RESOLUTION=512x288, CODECS="avc1.42001f, mp4a.40.2"
stream-3-480000/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=1063900, RESOLUTION=640x360, CODECS="avc1.42001f, mp4a.40.2"
stream-4-990000/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=2764000, RESOLUTION=852x480, CODECS="avc1.4d001f, mp4a.40.2"
stream-5-1800000/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=4564000, RESOLUTION=1280x720, CODECS="avc1.4d001f, mp4a.40.2"
stream-6-3000000/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=64000, CODECS="mp4a.40.2"
stream-0-64000/index.m3u8
```

Creating the Manifest

- Job of the encoding tool
- If on-premise encoder, encoder will create all manifests
- If cloud service, cloud service will create all manifests
- If open-source, typically use multiple tools
 - Encode in FFmpeg
 - FFmpeg can create variant, but not master
 - Create master in Bento4 or MP4 Box
- Apple Tools
 - Media File Segmenter – segments files, creates media playlist
 - Variant Playlist Creator – creates the master (also called variant playlist, because it lists all the variants)

Transmuxing the Manifest

- Most transmuxers can convert from one format (HLS) to another (DASH)
- Since text files only, very fast/lightweight operation

Questions?

- Questions

Should be 10:00

Lesson 5: Introduction to Encoding Ladders

- What they are and do
- A brief history of encoding ladder
- Creating a simple ladder – HD/H.264
- Creating a simple ladder 4K/HEVC

What Encoding Ladders Are and What They Do

- What they are
 - Collection of files encoded at different resolutions and data rates
 - Ensures that all viewers on all devices and connection speeds have a stream to view
 - Allows ABR technologies to adapt to changing bandwidth conditions
 - When bandwidth drops, player retrieves lower quality stream
 - When bandwidth increases, player retrieves higher quality stream

Table 2-1 Video average bit rate (kb/s) table 1

16:9 aspect ratio	H.264/AVC	Frame rate
416 x 234	145	≤ 30 fps
640 x 360	365	≤ 30 fps
768 x 432	730	≤ 30 fps
768 x 432	1100	≤ 30 fps
960 x 540	2000	same as source
1280 x 720	3000	same as source
1280 x 720	4500	same as source
1920 x 1080	6000	same as source
1920 x 1080	7800	same as source

A Brief History of Encoding Ladders

- Apple and TN2224
 - First really well developed specification
 - Very specific as to configurations
 - Some aspects tied to App store approval
 - Ensured playback on a range of old and new Apple devices
 - Given great credence by producers; some followed exactly
 - Later superceded by HLS Authoring Specification

Clients			Dimensions for 16:9 aspect ratio	Dimensions for 4:3 aspect ratio	Frame rate	Video bit rate (average)	Video bit rate (peak)	Audio bit rate	Total bit rate
	CELL		416 x 234	400 x 300	12	145	200	64	264
	CELL	ATV	480 x 270	480 x 360	15	365	400	64	464
WiFi	CELL	ATV	640 x 360	640 x 480	29.97	730	800	64	864
WiFi	CELL	ATV	768 x 432	640 x 480	29.97	1100	1200	96	1296
WiFi		ATV	960 x 540	960 x 720	29.97 or source	2000	2200	96	2296
WiFi		ATV	1280 x 720	960 x 720	29.97 or source	3000	3300	96	3396
WiFi		ATV	1280 x 720 or source	1280 x 960 or source	29.97 or source	4500	5000	128	5128
WiFi		ATV	1280 x 720 or source	1280 x 960 or source	29.97 or source	6000	6500	128	6628
WiFi		ATV	1920 x 1080	1920 x 1440	29.97 or source	7800	8600	128	8728

<http://bit.ly/appletn2224>

Ladder from Authoring Specification

- Superseded by Authoring spec
 - Codec specific ladders (this for H.264)
 - Many producers simply start with this ladder and adapt

Table 2-1 Video average bit rate (kb/s) table 1

16:9 aspect ratio	H.264/AVC	Frame rate
416 x 234	145	≤ 30 fps
640 x 360	365	≤ 30 fps
768 x 432	730	≤ 30 fps
768 x 432	1100	≤ 30 fps
960 x 540	2000	same as source
1280 x 720	3000	same as source
1280 x 720	4500	same as source
1920 x 1080	6000	same as source
1920 x 1080	7800	same as source

Apple Authoring Specification
http://bit.ly/hls_spec_2017

Adopting the Apple Spec: High End First

- Full screen viewing on all devices
- Highest quality streams that you can afford

Table 2-1 Video average bit rate (kb/s) table 1

16:9 aspect ratio	H.264/AVC	Frame rate
416 x 234	145	≤ 30 fps
640 x 360	365	≤ 30 fps
768 x 432	730	≤ 30 fps
768 x 432	1100	≤ 30 fps
960 x 540	2000	same as source
1280 x 720	3000	same as source
1280 x 720	4500	same as source
1920 x 1080	6000	same as source
1920 x 1080	7800	same as source

Desktop (browser-based) Next

- At least one stream for each window size in web site (MTV)
- Try to use same configuration as mobile

Scenario	Format	Frame Size	Total Bitrate	Audio Bitrate	bits/pixel *frame @ 30 fps	bits/pixel *frame @ 24 fps
Mobile & constrained (low)	baseline, mono, 10 fps	448x252	150	48	0.09	0.09
Mobile & constrained (high)	baseline, mono	448x252	450	48	0.12	0.15
Sidebar placements	main profile, stereo	384x216	400	96	0.12	0.15
Small in-page	main profile, stereo	512x288	750	96	0.15	0.18
Medium in-page	main profile, stereo	640x360	1200	96	0.16	0.20
Large in-page	main profile, stereo	768x432	1700	96	0.16	0.20
Full size in-page	main profile, stereo	960x540	2200	96	0.14	0.17
HD 720p (full screen)	high profile, stereo	1280x720	3500	96	0.12	0.15

Configuring Your Streams: Mobile Last

- How low will you go?
 - Slowest connection, lowest quality
 - Many drop data rate to preserve frame quality
 - Many producers don't deploy 145 kbps stream
 - Some deploy audio-only stream
 - Try to configure at same resolutions as low end computer targets

Table 2-1 Video average bit rate (kb/s) table 1

16:9 aspect ratio	H.264/AVC	Frame rate
416 x 234	145	≤ 30 fps
640 x 360	365	≤ 30 fps
768 x 432	730	≤ 30 fps
768 x 432	1100	≤ 30 fps
960 x 540	2000	same as source
1280 x 720	3000	same as source
1280 x 720	4500	same as source
1920 x 1080	6000	same as source
1920 x 1080	7800	same as source

Stream Count – Bottom Line

- At least one stream for each playback window in website
- More streams required for HD than SD
 - SD – usually 3, 4 maximum
 - HD – up to 11
 - 4K - up to 13 or higher
- More for entertainment than education/business
 - Entertainment – about the experience
 - Business – it's about making sure the viewer can watch the stream
- More for subscription than general entertainment
 - Provide more options when viewer is paying

What Data Rates?

- Apple TN2224: Keep adjacent bit rates a factor of 1.5 to 2 apart
 - If too close together, you waste band-width because quality difference is minimal (150 kbps and 180 kbps streams)
 - If too far apart, could strand some clients to lower quality stream unnecessarily

Minding the Jump

- Google sheet
 - Compute percentage jump from rung to rung
 - Red is outside 100% - 200%
 - Orange is close

	Width	Height	Data Rate	% Jump	FPS
234p	416	234	145		15
270p	480	270	365	2.52	15
360p	640	360	730	2.00	30
432p	768	432	1100	1.51	30
540p	960	540	2000	1.82	30
720p	1280	720	3000	1.50	30
1080p_l	1920	1080	4500	1.50	30
1080p_m	1920	1080	6000	1.33	30
1080p_h	1920	1080	7800	1.30	30
1440p	2560	1440	8100	1.04	30
2160p_low	3840	2160	11600	1.43	30
2160p_high	3840	2160	16800	1.45	30

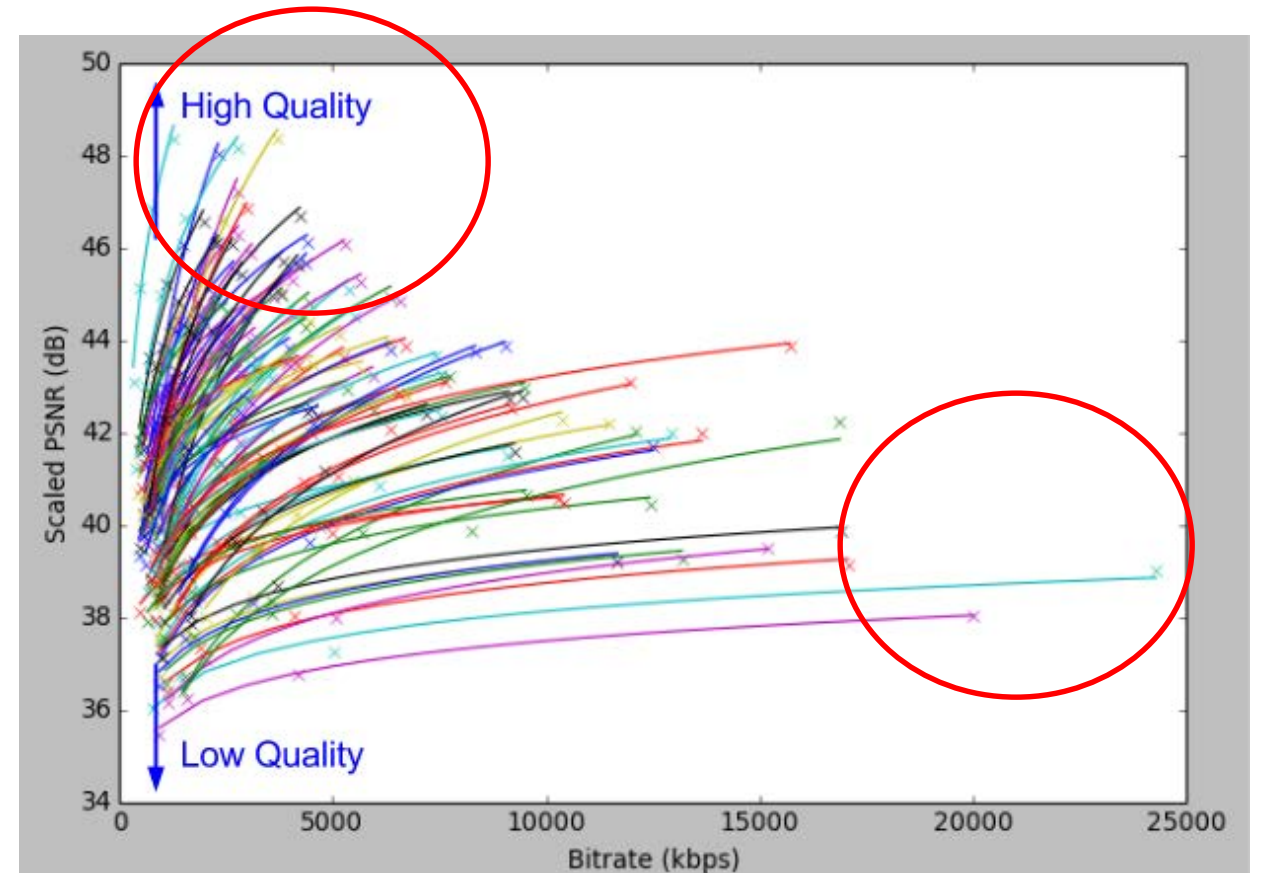
HEVC/VP9/AV1

- Apple has a separate ladder for HEVC
 - Drops lowest data rates, reflecting HEVC's greater efficiency
- Use same technique to derive ladder

16:9 aspect ratio	HEVC/H.265 30 fps	HDR (HEVC) 30 fps	Frame rate
640 x 360	145	160	≤ 30 fps
768 x 432	300	360	≤ 30 fps
960 x 540	600	730	≤ 30 fps
960 x 540	900	1090	≤ 30 fps
960 x 540	1600	1930	same as source
1280 x 720	2400	2900	same as source
1280 x 720	3400	3850	same as source
1920 x 1080	4500	5400	same as source
1920 x 1080	5800	7000	same as source
2560 x 1440	8100	9700	same as source
3840 x 2160	11600	13900	same as source
3840 x 2160	16800	20000	same as source

What's the Problem With a Single Encoding Ladder?

- The Apple specs were the Rosetta Stone for most early producers
- Then Netflix recognized that all videos encode differently
 - Scale on chart (quality/data rate)
 - These high quality at a low bitrate
 - These don't achieve same quality even at a much higher bitrate



Netflix Invented Per-Title Encoding

- All videos encode differently
- Fixed bitrate ladder (animated file)
 - Either data rate too high (wasted bandwidth), or
 - Data rate too low (quality not optimized)
- Per-title – analyzed file
 - Created ladder with unique:
 - Number of rungs
 - Resolutions
 - Data rates

	Before	After
Resolutions	Default bitrate ladder	Per-title bitrate ladder
320x240	235	150
384x288	375	200
512x384	560	290
512x384	750	
640x480	1050	
720x480	1750	440
720x480		590
1280x720	2350	830
1280x720	3000	1150
1920x1080	4300	1470
1920x1080	5800	2150
1920x1080		3840

Pros and Cons of Per-Title

Pros

- Reduced bandwidth and storage for easy to encode clips
- Improved QoE
 - Instead of 720p stream, get 1080p stream
- Improved quality (for hard to encode clips)

Cons

- Cost
- Encoding time
- Complexity
- But
 - Easier and cheaper than deploying a new codec (uses same player)
 - Delivers many of the same benefits

Bottom Line

- Per-title is key technology for all producers distributing mission critical video
- Either
 - Higher QoE
 - Lower bandwidth/storage
 - or, both
- Session on per-title later in the week

Questions?

Should be 10:15

Lesson 6: Introduction to Objective Quality Metrics

- What they are
- Why we need them
- Meet VMAF
- Finding the floor
- Finding the resolutions (the Netflix technique)

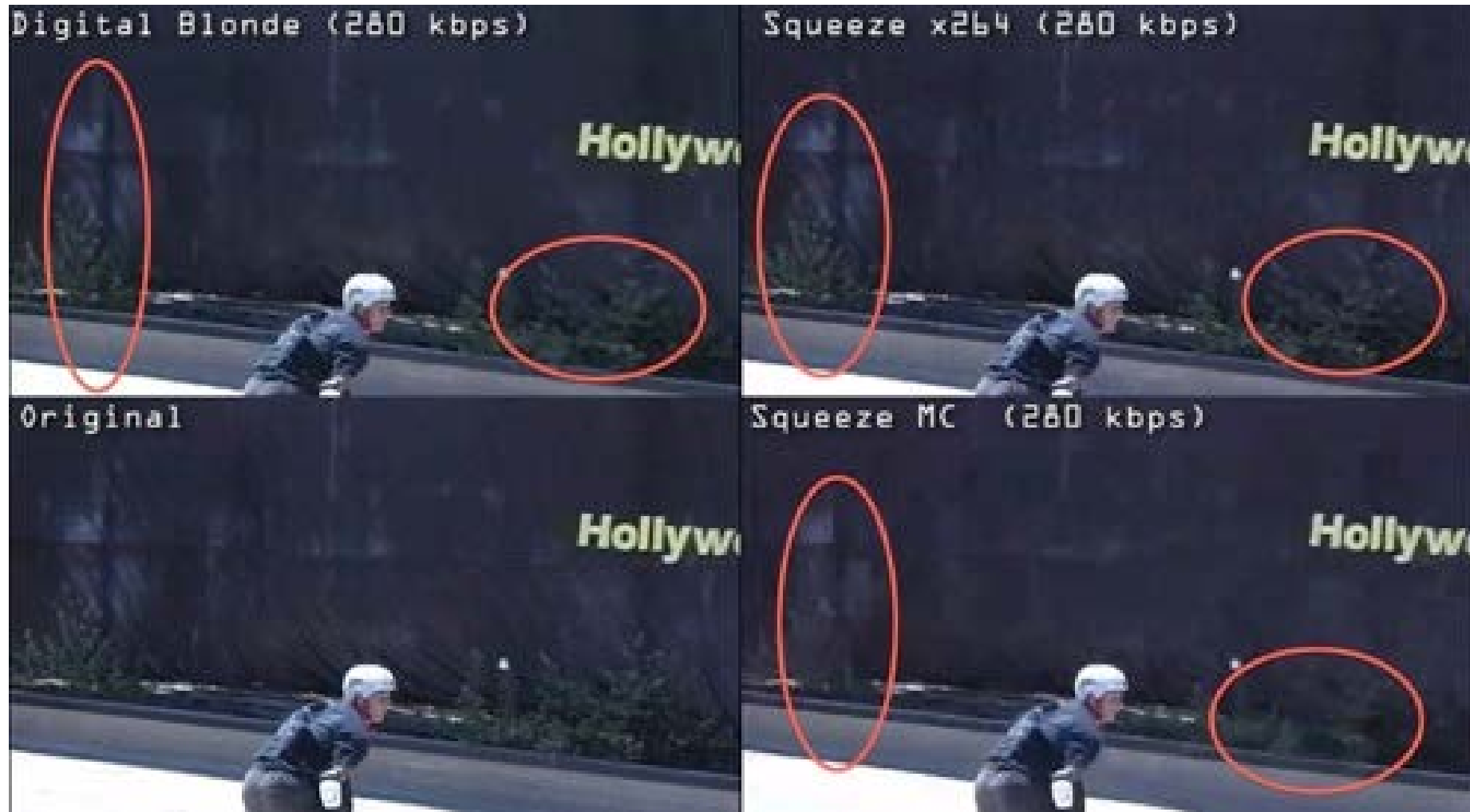
What Are Objective Quality Metrics

- Mathematical formulas that (attempt to) predict how human eyes would rate the videos
 - Faster and less expensive
 - Automatable
- Examples
 - Peak Signal to Noise Ratio (PSNR)
 - Structural Similarity Index (SSIM)
 - SSIMPlus
 - VMAF (Video Multimethod Assessment Fusion)

Why Do We Need Them?

- So many encoding decisions
 - Data rate
 - Keyframe interval
 - B-frame interval
 - Bitrate control technique (VBR vs. CBR)
 - Choice of codec
 - Profile
 - Preset
- All have tradeoffs (quality vs. encoding time)
- Objective quality metrics allow us to mathematically measure quality
- Uses
 - Drive many per-title encoding technologies (Netflix)
 - Useful for many critical encoding decisions

Took Me From Here

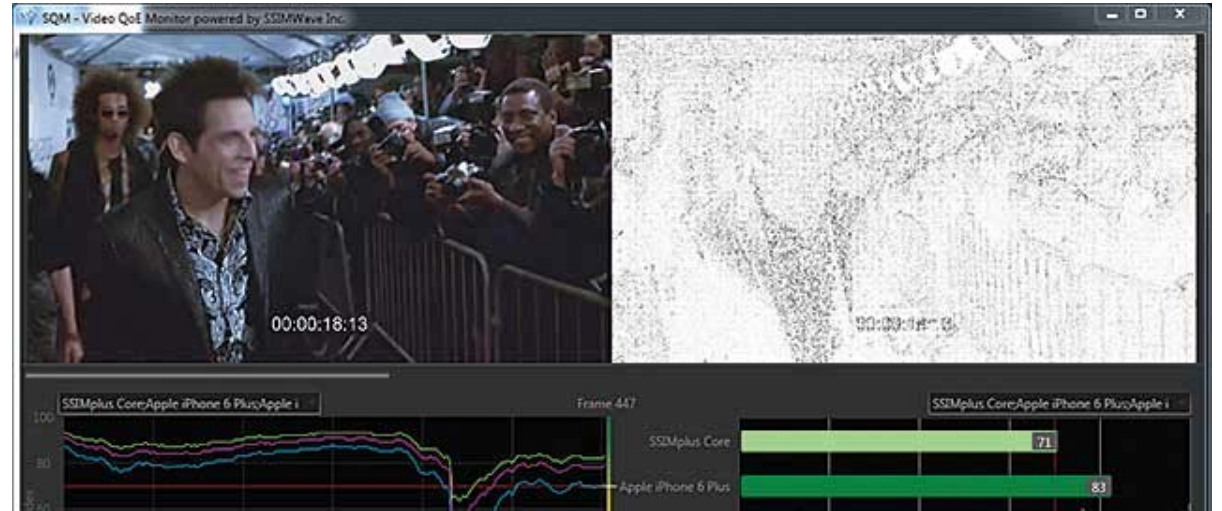


Time consuming and error prone
Subjective comparisons

To Here

VQM (lower is better)					
	Codec A	Codec B	Codec C	High > Low	Codec A > Codec B
Office 1	0.36	0.36	0.37	-3.54%	0.61%
Office 2	0.69	0.61	0.70	-13.51%	12.32%
Office 3	0.28	0.28	0.32	-14.74%	1.32%
Office 4	0.87	0.79	0.87	-9.63%	9.63%
Parking 1	0.68	0.61	0.74	-21.23%	10.90%
Parking 2	0.57	0.55	0.64	-15.47%	3.04%
Parking 3	1.86	1.58	1.76	-17.88%	17.88%
Parking 4	0.47	0.49	0.51	-8.86%	-3.81%
Retail 1	0.56	0.54	0.56	-4.27%	4.27%
Retail 2	0.68	0.66	0.69	-4.45%	3.39%
Retail 3	0.78	0.72	0.76	-8.64%	8.64%
Retail 4	0.73	0.67	0.88	-32.16%	8.52%
Traffic 1	0.55	0.50	0.58	-15.89%	9.14%
Traffic 2	0.34	0.32	0.38	-17.79%	6.39%
Traffic 3	0.52	0.49	0.55	-11.42%	5.29%
Traffic 4	0.68	0.61	0.66	-11.56%	11.56%
Total	10.61	9.78	10.96		
7.84%	Difference between Codec A and Codec B				
-3.34%	Difference between Codec A and Codec C				
-12.13%	Difference between Codec B and Codec C				
	0.61				
	Green equals best in category				
	Orange means worst in category				
	Difference greater than 7.5%				

Statistically meaningful comparisons



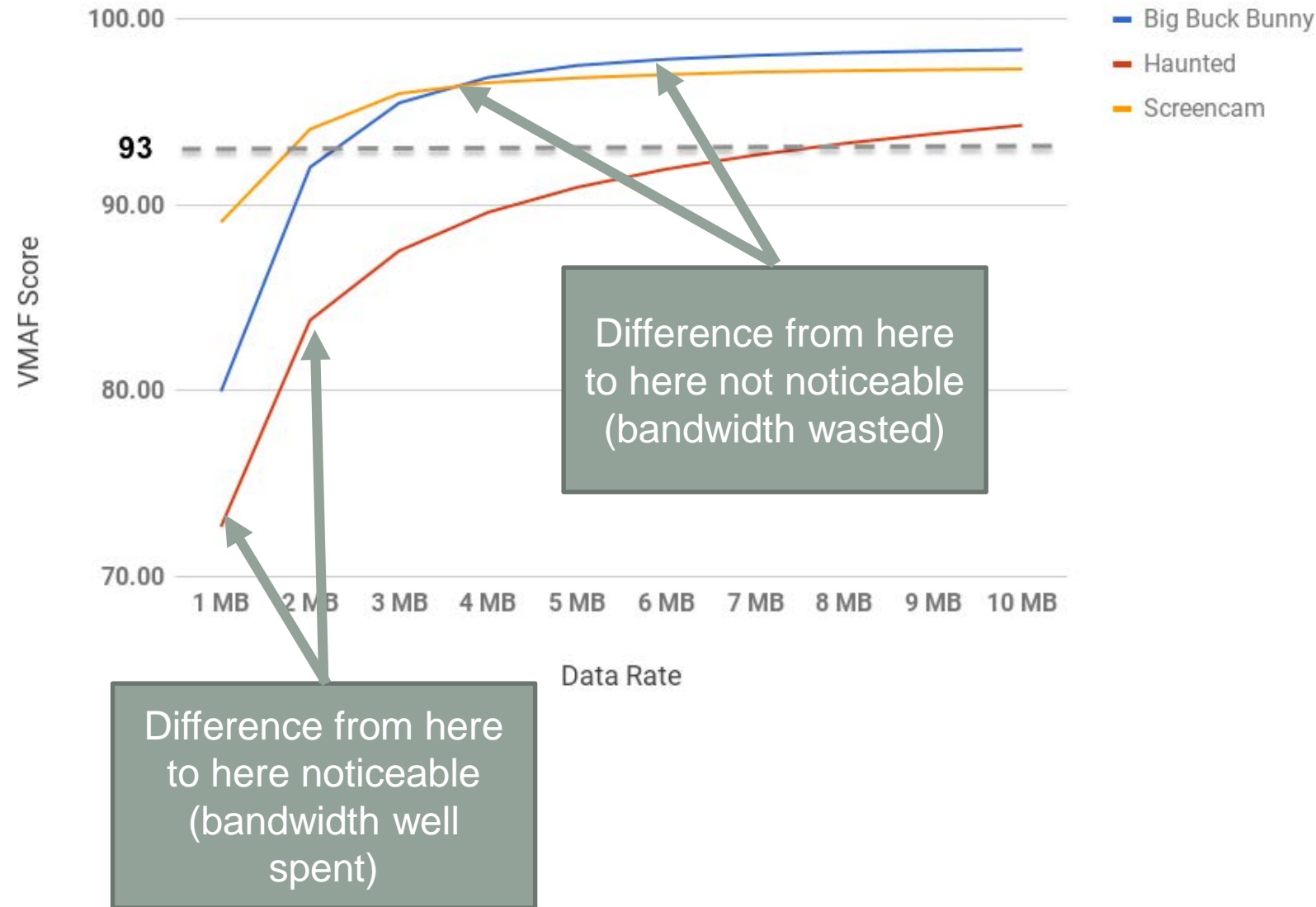
With Objective Quality Metrics You Get

- More data
 - Can run many more tests in much less time
- Better data
 - Mathematical models can detect smaller changes than your eye can easily discern

Meet VMAF

- Video Multimethod Assessment Fusion
- Created by Netflix; blends four objective metrics
- Range – 0 – 100
- At full resolution, value of 93+ means “either indistinguishable from original or with noticeable but not annoying distortion.”
 - RealNetworks http://bit.ly/vrqm_5
- At lower resolutions, no tie to quality, but higher scores are always better
- 6 VMAF points = Just noticeable difference

Impact of Data Rate on VMAF Quality - 1080p



Bottom Line

- There will always be contention for “best” metric
- VMAF seems to be the most effective up and down the encoding ladder
- Good enough for Netflix, good enough for me
- Accessing VMAF
 - Moscow State University Video Quality Measurement Tool (\$995)
 - Hybrik Cloud Encoding Platform
 - Open Source (build a tool yourself)
- More on objective quality metrics?
 - Sessions at Streaming Media West 2017

Questions?

**Should be 10:25
Break**

Lesson 7: Building Your Encoding Ladder with VMAF/CRF

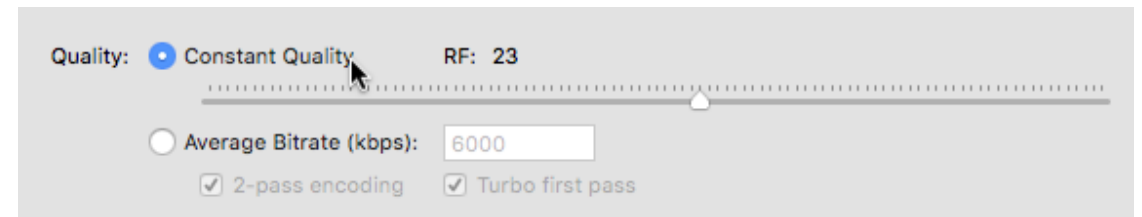
- Simple case
- HEVC and advanced codecs
- Animations and synthetic videos

Using Quality Metrics – Finding the Floor

- What is the floor?
 - The lowest full resolution data rate that delivers acceptable quality
- Finding the floor
 - About CRF
 - VMAF correlation
 - Hollywood proof
 - RealNetwork Verification
- Choosing the resolutions

About Constant Rate Factor Encoding

- Encoding mode available in x264, x265, VP8/9
- Encodes to a specific *quality* level, not a data rate
- Two uses
 - As gauge of encoding complexity
 - With caps, a per-title encoding technique



- Range is 1-51
 - Lower number means higher quality
 - For 2D video, CRF 23 roughly delivers Hollywood (iTunes) quality

Finding the Optimal Data Rate (Per-title)

- Compute data rate with CRF 23
 - Values varied from 1,001 to 6,111 (over 600%)
- Measure VMAF rating
 - Values ranged from 92.74 to 96.88
 - Standard deviation was 1.39 (pretty small)
- Analysis
 - At 2.7 Mbps, a talking head video offers same quality as movie at 6.1 Mbps (even more for synthetic videos)
 - Validating the benefits of per-title encoding

CRF23 - 1080p	FPS	Description	Data Rate	VMAF
Tears of Steel	24	Real world/CG movie	4,747	96.45
Sintel	24	Complex animation	5,168	96.96
Big Buck Bunny	30	Simple animation	3,657	96.88
Screencam	30	Camtasia-based video	1,625	96.59
Tutorial	30	PowerPoint and talking head	1,001	96.68
Talking Head	30	Simple talking head	2,706	95.47
Freedom	30	Concert footage	5,527	95.90
Haunted	30	DSLR movie-like production	6,111	92.74
Average			3,818	95.96
Standard deviation				1.39

- Conclusion:
 - CRF 23 maps accurately to VMAF score of 93

Hollywood Verification

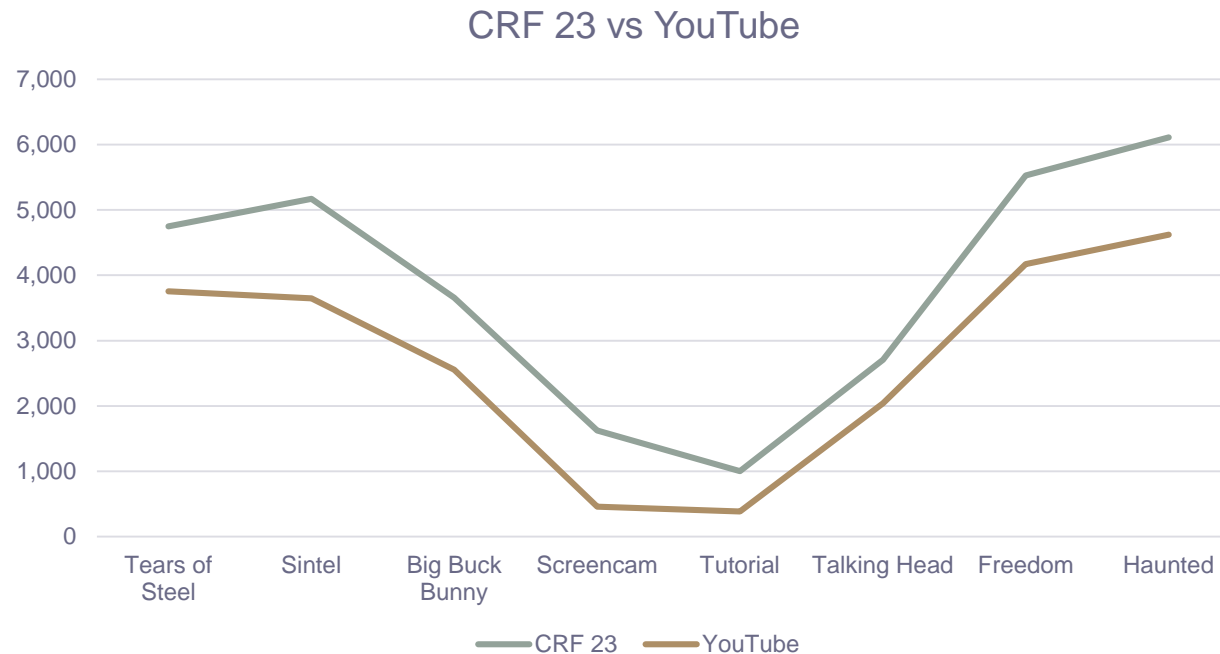
Program	Owner	Width	Height	Frame Rate	Target Video DR	Bits Per Pixel
Angie Tribeca	Tuner	1916	1076	23.976	5,060	0.102
Better Call Saul	Sony	1916	1076	23.976	5,169	0.105
Blackish	ABC/Disney	1920	1080	23.976	4,953	0.1
Brooklyn 999	Universal	1920	1080	23.976	5,094	0.102
Family Guy	FoxTV	1920	1080	23.976	5,173	0.104
Fresh of the Boat	20th Century	1920	1080	23.976	4,946	0.099
Full Frontal	TBS	1920	1080	23.976	5,238	0.105
I am Cait	E!	1440	1080	23.976	5,261	0.141
Sherlock	BBC	1920	1080	25	5,062	0.098
The Affair	Showtime	1912	1080	23.976	4,959	0.1
Last Man on Earth	20th Century	1920	1080	23.976	5,117	0.103
Transformers	Hasbro	1920	1080	23.976	5,128	0.103
Average					5,097	0.105

- Our two 24 fps movie-like titles averaged about 4.95 Mbps
- Hollywood titles downloaded from iTunes averaged 5.1 Mbps
- Data rates are similar
- Verifies that CRF 23 and VMAF 93 deliver “Hollywood” quality

VMAF Verification – 93 is the Number

- Real Networks White Paper - VMAF Reproducibility: Validating a Perceptual Practical Video Quality Metric
 - 4K 2D videos
- The results indicate that if a video service operator were to encode video to achieve a ***VMAF score of about 93*** then they would be confident of optimally serving the vast majority of their audience with content that ***is either indistinguishable from original or with noticeable but not annoying distortion.***
 - http://bit.ly/vrqm_5

Reality Check: YouTube Comparison



- Upload files to YouTube; measure data rate
- YouTube uses AI-based per-title optimization
- Pattern very similar
- YouTube averages 1 Mbps lower
- 3 VMAF points lower (1/2 JND)

So

- Full rez 2D videos, CRF 23 = ~93 VMAF = shippable quality
- Significant data point
 - As you'll see, encoding ladder starts at the top
- What's this mean for you?
 - Fixed bitrate ladder – make sure hardest to encode video equals 93 VMAF score at top rung
 - Per-category – do the same for hardest-to-encode videos in each category
 - Per-title – do the same for each video

Choosing Your Data Rates

- Step 1: Choose highest
- Step 2: Choose lowest
- Step 4: fill in the blanks
(between 150/200% apart)

200 kbps

500 kbps

1000 kbps

1600 kbps

2100 kbps

3100 kbps

4600 kbps

Choosing Resolution:

- Netflix approach
 - Compute VMAF scores at multiple resolutions at each data rate
 - Use resolution with highest VMAF score at each data rate rung

Zap1 - VMAF	4K	2K	1080p	720p	480p	360p	240p
5000	90.19	89.70	84.82				
4500	89.58	88.23	84.38				
4000	88.43	87.50	83.84				
3800	87.88	87.14	83.58				
3600	87.27	86.71	83.25				
3400	86.60	85.72	82.87				
3200	85.80	85.40	82.45				
3000	85.03	85.09	82.01				
2800	83.97	84.34	81.43				
2600	82.86	83.50	80.85				
2400	81.45	82.51	80.09	71.92			
2200	79.79	81.24	79.20	71.35			
2000	77.94	79.82	78.04	70.66			
1800		78.11	76.73	69.70	53.28		
1600		75.91	74.93	68.41	52.82		
1400		73.26	72.64	66.89	52.13	32.07	
1200		69.83	69.69	64.68	51.05	31.75	
1000		65.15	65.75	61.64	49.36	31.17	
900		62.26	63.25	59.64	48.11	30.76	
800		58.69	60.27	57.20	46.54		
700		54.29	56.62	54.06	44.63	29.18	
600		48.79	52.32	50.65	42.02	27.84	
500			46.65	45.96	38.74	25.92	
400			39.06	40.23	34.21	23.11	
300			28.52	32.68	x		
200			13.88	21.73	x		

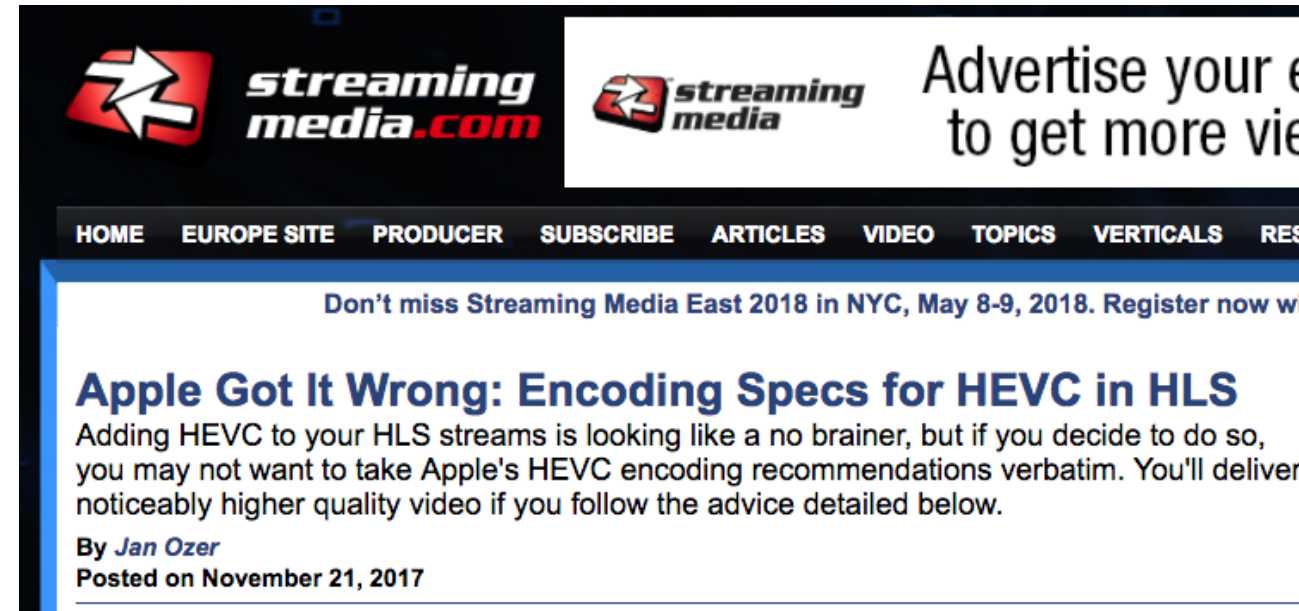
How Ladders Change for Advanced Codecs

- A short pictorial
- June 2017
 - Apple specs show same resolutions for H.264 and H.265



HEVC/H.265 30 fps	H.264/AVC	Resolution 16:9 aspect ratio
145	145	416 x 234
300	365	480 x 270
660	730	640 x 360
990	1100	768 x 432
1700	2000	960 x 540
2400	3000	1280 x 720
3200	4500	1280 x 720
4500	6000	1920 x 1080
5800	7800	1920 x 1080

November 2017

- Streaming Media article
- What did it say?
 - When creating an encoding ladder for HEVC video, don't duplicate the resolutions of the same ladder used for H.264.
 - You'll optimize quality by pushing higher resolutions lower down in the encoding ladder and eliminating the bottom few rungs.



The screenshot shows the top portion of a web page from streamingmedia.com. The header features the site's logo on the left, a navigation menu with links like HOME, EUROPE SITE, PRODUCER, SUBSCRIBE, ARTICLES, VIDEO, TOPICS, VERTICALS, and RESOURCES in the center, and an advertisement on the right. Below the navigation bar is a promotional banner for 'Streaming Media East 2018'. The main article title, 'Apple Got It Wrong: Encoding Specs for HEVC in HLS', is displayed in a large, bold font, followed by a short introductory paragraph and the author's name, Jan Ozer.

 **streaming media.com**  Advertise your e to get more vie

HOME EUROPE SITE PRODUCER SUBSCRIBE ARTICLES VIDEO TOPICS VERTICALS RES

Don't miss Streaming Media East 2018 in NYC, May 8-9, 2018. Register now w

Apple Got It Wrong: Encoding Specs for HEVC in HLS

Adding HEVC to your HLS streams is looking like a no brainer, but if you decide to do so, you may not want to take Apple's HEVC encoding recommendations verbatim. You'll deliver noticeably higher quality video if you follow the advice detailed below.

By *Jan Ozer*
Posted on November 21, 2017

Proof – Tears of Steel

H.264

H.264	1080p	720p	540p	432p	360p	270p	234p
5000	96.22						
4800	96.01						
4600	95.80	95.27					
4400	95.55	95.10					
4200	95.30	94.96					
4000	94.96	94.73					
3800	94.60	94.53					
3600	94.14	94.30					
3400	93.70	93.99					
3200	93.11	93.64					
3000	92.48	93.24					
2800	91.70	92.78					
2600	90.75	92.25					
2400	89.70	91.59	90.39				
2200	88.37	90.80	89.76				
2000	86.72	89.85	88.95	86.93			
1800	84.68	88.66	88.00	86.10			
1600	82.13	87.13	86.77	85.02	81.58		
1400	78.65	85.19	85.16	83.67	80.28		
1200	73.91	82.56	83.01	81.84	78.57		
1000	67.39	78.86	80.02	79.24	76.19		
900	63.18	76.39	77.98	77.47	74.60	66.66	60.58
800	57.93	73.25	75.51	75.34	72.68	65.11	59.23
700	51.47	69.42	72.34	72.59	70.23	63.14	57.49
600	43.12	64.52	68.37	69.11	67.12	60.70	55.33
500	33.31	58.05	63.13	64.66	63.04	57.52	52.46
400	20.82	49.48	56.00	58.46	57.48	53.13	48.59
300	9.74	37.56	45.95	49.62	49.60	46.80	42.96
200	3.73	20.40	30.87	36.12	37.48	36.88	34.03
100		2.75	8.08	14.45	17.50	19.85	18.66

HEVC

HEVC	1080p	720p	540p	432p	360p	270p	234p
5000	97.67						
4800	97.55						
4600	97.44						
4400	97.31						
4200	97.17						
4000	97.01						
3800	96.84						
3600	96.63						
3400	96.41						
3200	96.15	95.41					
3000	95.86	95.16					
2800	95.52	94.87					
2600	95.09	94.52					
2400	94.58	94.12	92.09				
2200	93.97	93.63	91.62				
2000	93.16	92.02	91.05	88.30			
1800	92.18	92.25	90.34	87.63			
1600	90.94	91.27	89.44	86.78	83.18		
1400	89.36	89.97	88.27	85.69	82.12		
1200	87.30	88.26	86.68	84.22	80.73		
1000	84.42	85.84	84.46	82.20	78.79		
900	82.39	84.21	83.02	80.86	77.51	68.45	62.18
800	80.03	82.20	81.23	79.19	75.91	67.09	60.92
700	77.04	79.67	78.90	77.07	73.91	65.38	59.35
600	73.10	76.34	75.88	74.29	71.36	63.21	57.34
500	68.11	71.98	71.82	70.61	67.89	60.30	54.69
400	61.01	65.92	66.31	65.54	63.19	56.29	51.89
300	50.13	57.34	58.21	58.06	56.18	50.49	45.69
200	25.00	44.30	45.88	46.47	45.29	40.96	37.13
100	4.14	13.75	24.62	26.16	25.85	23.86	21.53

1080p best quality at far lower data rates than H.264

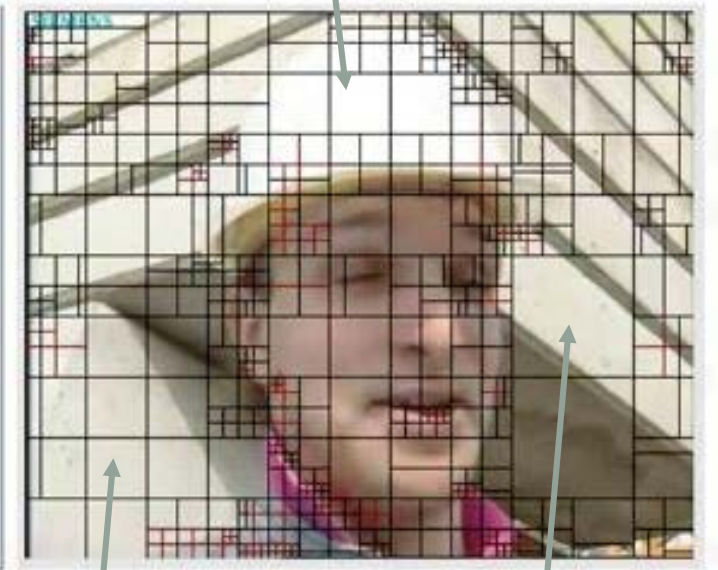
Lower resolutions don't provide the best quality

Why is HEVC More Efficient?

- Simply a better codec
- One prominent advantage – larger block sizes
 - H.264 – 16x16
 - HEVC – 64x64
- Can encode large images more efficiently



H.264



H.265

March 2018 – Apple Creates Separate Tables for HEVC and H.264

Eliminates lower
rez rungs

16:9 aspect ratio	H.264/AVC	Frame rate
416 x 234	145	≤ 30 fps
640 x 360	365	≤ 30 fps
768 x 432	730	≤ 30 fps
768 x 432	1100	≤ 30 fps
960 x 540	2000	same as source
1280 x 720	3000	same as source
1280 x 720	4500	same as source
1920 x 1080	6000	same as source
1920 x 1080	7800	same as source

16:9 aspect ratio	HEVC/H.265 30 fps	Frame rate
640 x 360	145	≤ 30 fps
768 x 432	300	≤ 30 fps
960 x 540	600	≤ 30 fps
960 x 540	900	≤ 30 fps
960 x 540	1600	same as source
1280 x 720	2400	same as source
1280 x 720	3400	same as source
1920 x 1080	4500	same as source
1920 x 1080	5800	same as source
2560 x 1440	8100	same as source
3840 x 2160	11600	same as source
3840 x 2160	16800	same as source

Didn't change switch
points (was 3200)

Conclusion

- Use different resolutions and switch points for H.264 and advanced codecs

What About Different Types of Content?

- In general:
 - Synthetic videos encode at higher quality at lower bitrates (not shown here)
 - Look better at higher resolutions
 - Push 1080p lower down in the encoding ladder
 - Push 720p further down the ladder
- Not huge difference here, but much more profound for screencams and similar videos

Tears of Steel (real world/CG)

HEVC	1080p	720p	540p	432p	360p	270p	234p
5000	97.67						
4800	97.55						
4600	97.44						
4400	97.31						
4200	97.17						
4000	97.01						
3800	96.84						
3600	96.63						
3400	96.41						
3200	96.15	95.41					
3000	95.86	95.16					
2800	95.52	94.87					
2600	95.09	94.52					
2400	94.58	94.12	92.09				
2200	93.97	93.63	91.62				
2000	93.16	93.02	91.05	88.30			
1800	92.18	92.25	90.34	87.63			
1600	90.94	91.27	89.44	86.78	83.18		
1400	89.36	89.97	88.27	85.69	82.12		
1200	87.30	88.26	86.68	84.22	80.73		
1000	84.42	85.84	84.46	82.20	78.79		
900	82.39	84.21	83.02	80.86	77.51	68.45	62.18
800	80.03	82.20	81.23	79.19	75.91	67.09	60.92
700	77.04	79.67	78.90	77.07	73.91	65.38	59.35
600	73.10	76.34	75.88	74.29	71.36	63.21	57.34
500	68.11	71.98	71.02	70.61	67.89	60.30	54.69
400	61.01	65.92	66.31	65.54	63.19	56.29	51.05
300	50.13	57.34	58.21	58.06	56.18	50.40	45.64
200	25.00	44.30	45.88	46.47	45.24	40.96	37.13
100	4.14	13.75	24.62	26.16	25.85	23.86	21.53

Sintel (animation)

HEVC	1080p	720p	540p	432p	360p	270p	234p
5000	97.83						
4800	97.74						
4600	97.63						
4400	97.50						
4200	97.36						
4000	97.19						
3800	97.01						
3600	96.78						
3400	96.52						
3200	96.22	94.39					
3000	95.86	94.11					
2800	95.45	93.78					
2600	94.94	93.40					
2400	94.32	92.93	89.84				
2200	93.62	92.37	89.34				
2000	92.72	91.69	88.71	85.40			
1800	91.63	90.84	87.94	84.72			
1600	90.21	89.76	87.00	83.84	79.64		
1400	88.44	88.36	85.74	82.74	78.62		
1200	86.02	86.39	84.07	81.24	77.24		
1000	82.81	83.73	81.70	79.13	75.35		
900	80.79	82.02	80.16	77.76	74.10	64.67	58.74
800	78.22	79.83	78.25	76.06	72.55	63.43	57.63
700	75.22	77.22	75.91	73.94	70.64	61.88	56.22
600	71.44	73.84	72.94	71.27	68.17	59.87	54.42
500	66.61	69.68	69.13	67.71	64.90	57.24	52.02
400	60.19	62.98	63.94	62.97	60.47	53.61	48.73
300	48.81	56.19	56.62	56.22	54.16	48.26	43.81
200	26.36	44.11	45.66	53.05	44.22	39.79	36.06
100	5.17	15.45	23.86	26.96	26.53	24.50	21.89

Questions

Should be 11:00

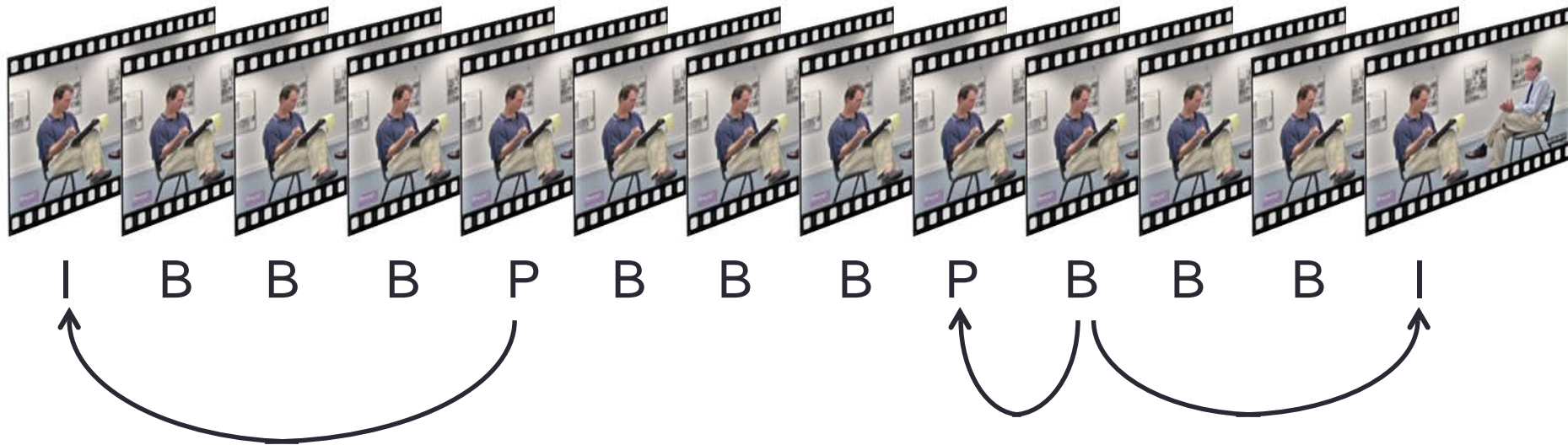
Lesson 8: Encoding for ABR Videos

- In general; all encoding rules apply, but **three** are specific to ABR
- Choosing the keyframe interval
- Choosing segment duration
- Choosing bitrate control technique

I-Frame Interval (Keyframe)

- What are I-frames
- Choosing a keyframe interval
- Configuring keyframe parameters

What are I, B and P Frames?



- I-Frame - encoded without reference to other frames (also called keyframes)
- P - looks backward to I and P frames (predictive)
- B - looks forward and backward to previous I and P frames (Bi-directional interpolated)
 - No frames refer to B-Frame (most of the time)

Configuring keyframes

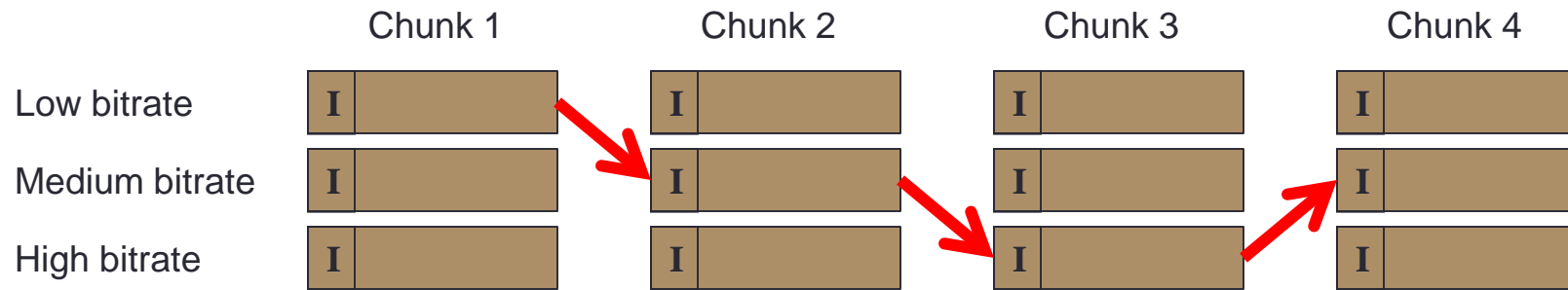
- Though largest frame, keyframes enhance interactivity
 - All playback starts on a keyframe
 - When seeking to random frame (like the third p-frame), must start playback at preceding keyframe
 - To enhance interactivity, maximum key interval should be 5-10 seconds



What About Adaptive?

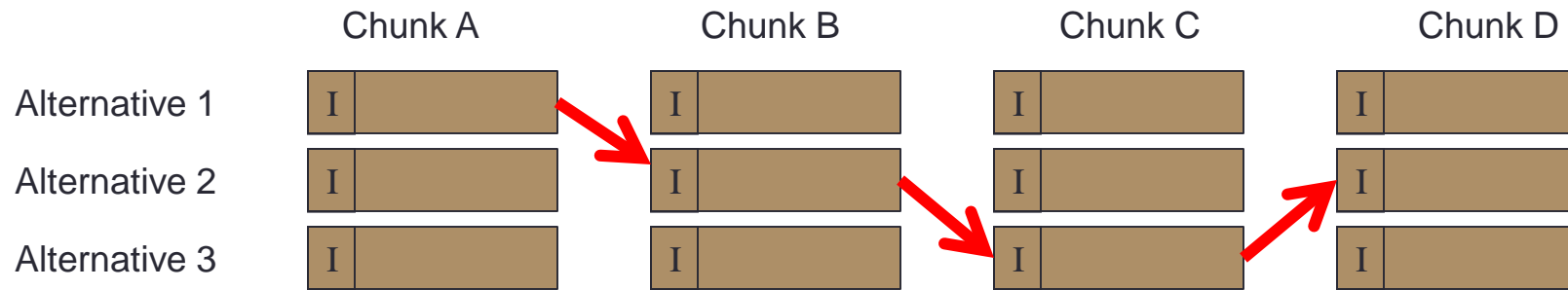
- Rules change when encoding a group of files for adaptive streaming
 - Shorter keyframe interval to enable more nimble stream switching
 - Ten seconds is forever when bandwidth drops
 - Keyframe interval must match in all files
 - Need regular interval (e.g.. every 90 frames)
 - Disable scene change detection when this will change this interval
 - Needs to divide evenly into segment size

What About Adaptive?



- Adaptive involves multiple streams (low, medium and high) using multiple chunks (1,2,3,4)
 - Switch streams by retrieving chunks from different alternative
 - So, need keyframe (I-frame) ***at start of every chunk***
 - ***So, keyframe interval must equal chunk size or be divisible into chunk size***

What About Adaptive?

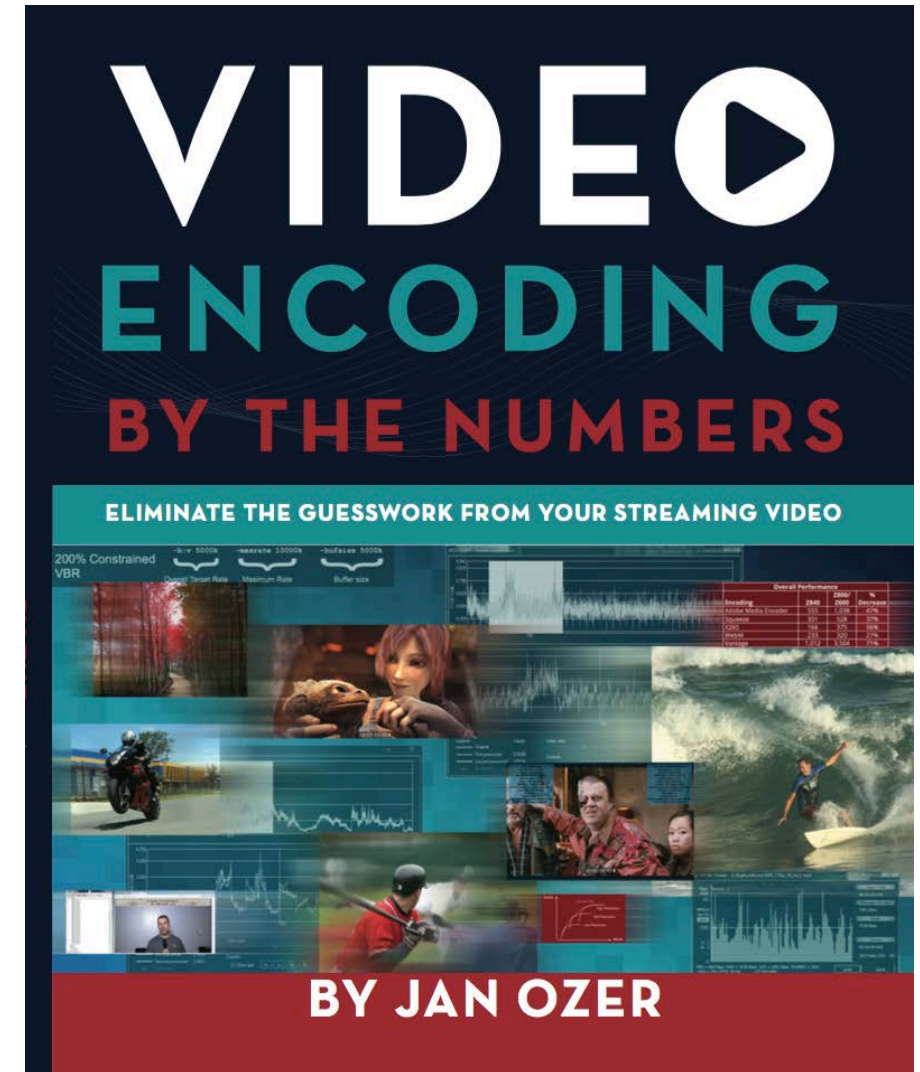


- Need ***regular*** keyframes
 - Some encoders restart keyframe interval when inserting a keyframe at a scene change
 - ***For these, disable scene change detection***, or:
 - Otherwise ensure keyframe at I-frame distance

☒ Fixed I-Frames Distance

Configuring Your Encodes

- Background: Video Encoding by the Numbers; December 2016
- Eight files
 - 1 movie (Tears of Steel)
 - 2 animations (Sintel, BBB)
 - Two general purpose (concert, advertisement)
 - One talking head
 - Screencam
 - Tutorial (PPT/Video)
- Tied all encoding decisions to PSNR
 - Updating to VMAF/Adding VR now

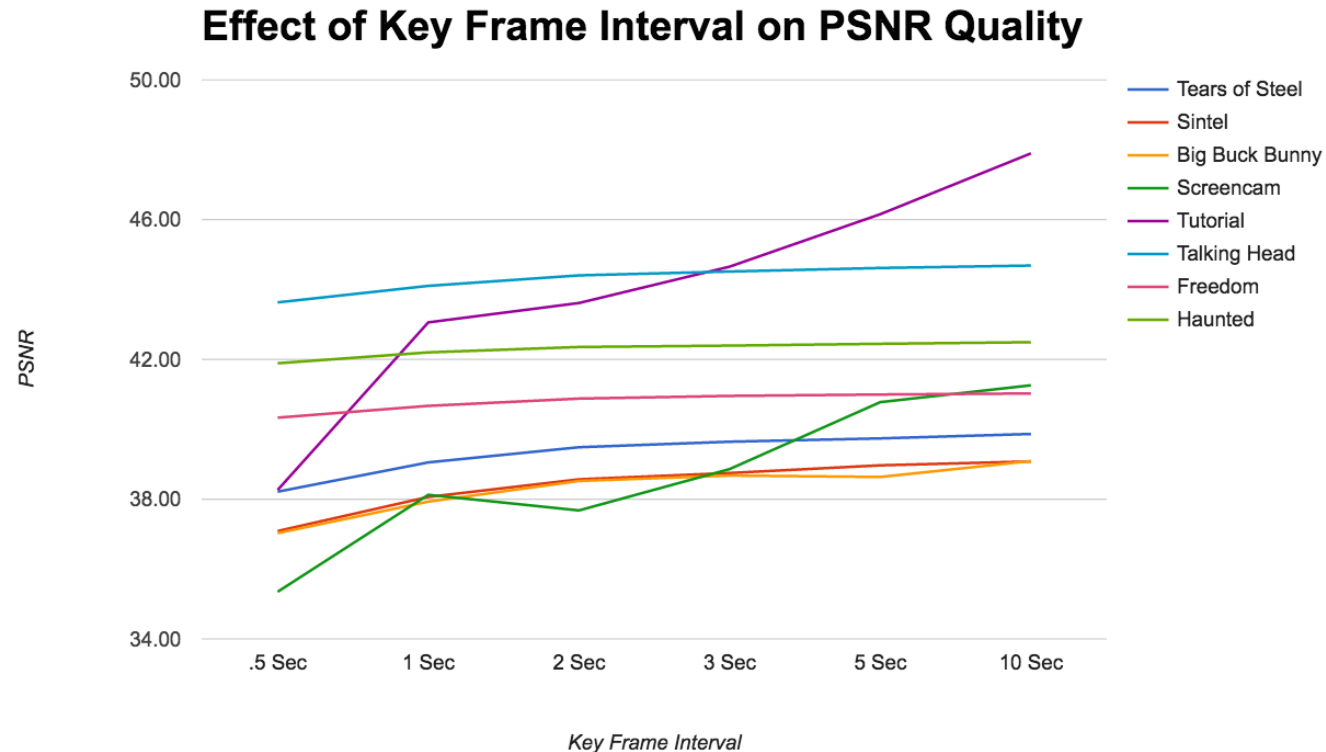


What's the Best Keyframe Interval

	.5 Sec	1 Sec	2 Sec	3 Sec	5 Sec	10 Sec	Max Delta
Tears of Steel	38.22	39.05	39.49	39.64	39.74	39.87	4.32%
Sintel	37.09	38.06	38.57	38.75	38.97	39.08	5.37%
Big Buck Bunny	37.03	37.93	38.52	38.68	38.64	39.09	5.57%
Talking Head	43.63	44.10	44.40	44.51	44.61	44.68	2.42%
Freedom	40.33	40.67	40.88	40.96	40.99	41.03	1.72%
Haunted	41.89	42.20	42.35	42.39	42.45	42.49	1.44%
Average	39.26	39.96	40.37	40.51	40.59	40.75	3.88%
Screencam	35.35	38.13	37.68	38.86	40.78	41.26	16.71%
Tutorial	38.26	43.06	43.61	44.65	46.15	47.89	25.17%

- .5 second never best option
- 2 seconds recommended with good reason

What's the Best Keyframe Interval



- Not as much difference as you might think
 - Screencam and Tutorial (PPT based video) - outliers
- Real world video, stops increasing after 2-3

Keyframe Summary

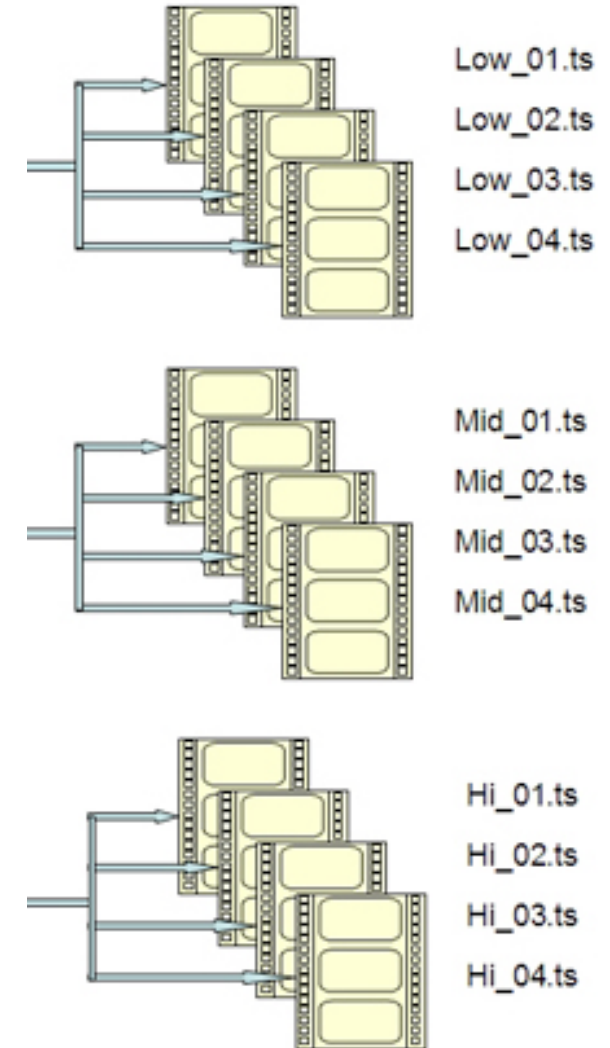
- Single file – 10 – 15 seconds OK
- Adaptive
 - 2-3 seconds (Apple recommends 2 seconds)
 - Divide evenly into segment size
 - Ensure key frames at start of each segment
 - Disable keyframes at scene changes, or
 - Force keyframes at selected interval

Choosing the Segment Size

- What is it?
- The simple answer
- Factors in the informed decision
- What are the recommendations

What is Segment Size

- Not technically an encoding decision; it's a packaging decision
- Duration of individual segments (if separate files) or byte range requests retrieved by the player



Use 6 Seconds – Because Apple Says So

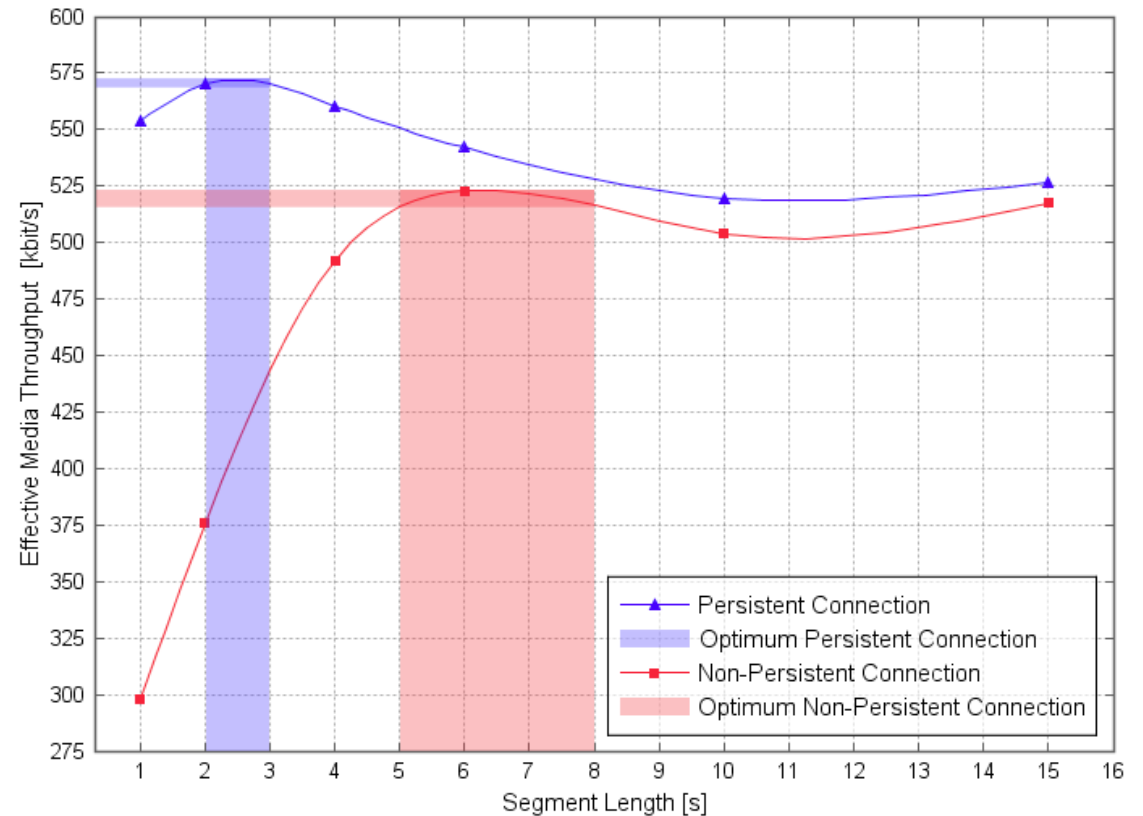
Media Segmentation

7. Media segmentation requirements

- 7.1. Your media **MUST** be continuous across segments, with the exception of transitions for ads and other inserted material.
- 7.2. If using a transport stream, continuity counters and timestamps **MUST** be sequential.
- 7.3. If using fMP4, the track fragment decode time **MUST** be consistent with the decode time and duration of the previous segment.
- 7.4. Video segments **MUST** start with an IDR frame.
- 7.5. Target durations **SHOULD** be 6 seconds.
- 7.6. Segment durations **SHOULD** be nominally 6 seconds (e.g., NTSC 29.97 may be 6.006 seconds).
- 7.7. Media segments **MUST NOT** exceed the target duration by more than 0.5 seconds.

Factors in the Informed Decision: Throughput

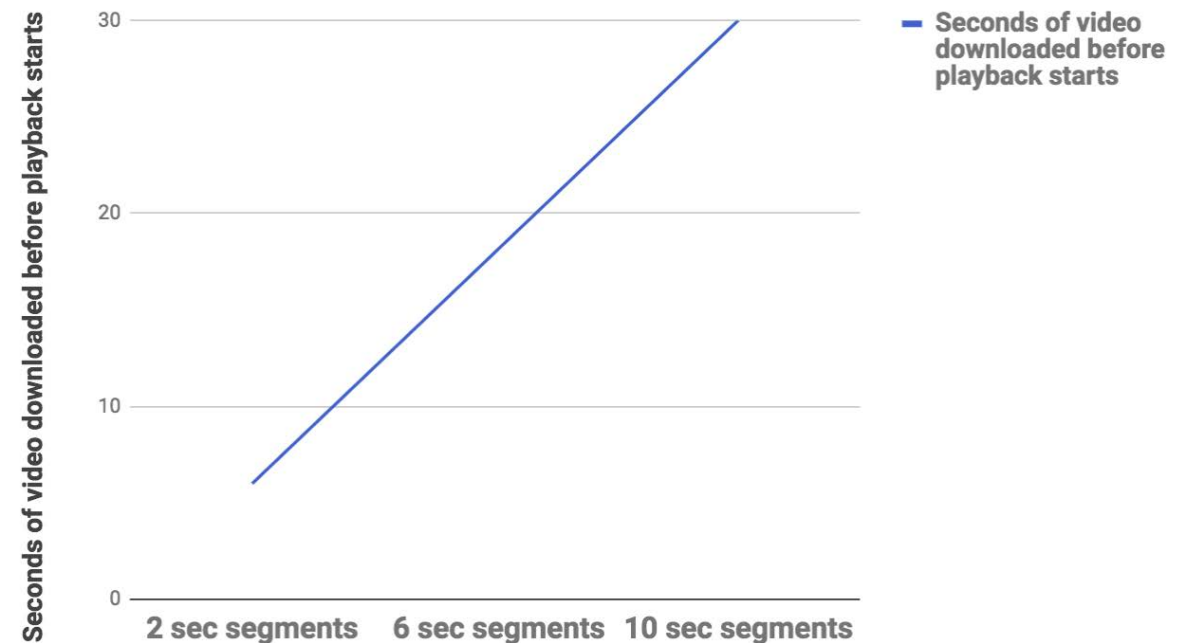
- Network throughput
 - Smaller segments sizes increase the number of requests the web server has to manage
 - If too frequent (and non-persistent connection), waste too much server cycles managing requests
 - Not an issue with persistent connections
 - Problem: Tough to ensure persistent connection in all instances
 - Resolution: use 6 seconds



Factors in the Informed Decision: Latency

- Many players don't start playback until it receives three segments
 - Segment size directly affects latency
- VOD – “time to download”
 - Not 30 seconds (time to download 30 seconds of video)
 - Time to start playback
- Live - best case, 3x segment size behind actual live event
 - True latency (delay vs. live)
 - If latency is critical need to modify

Seconds of Video Downloaded Before Playback Starts



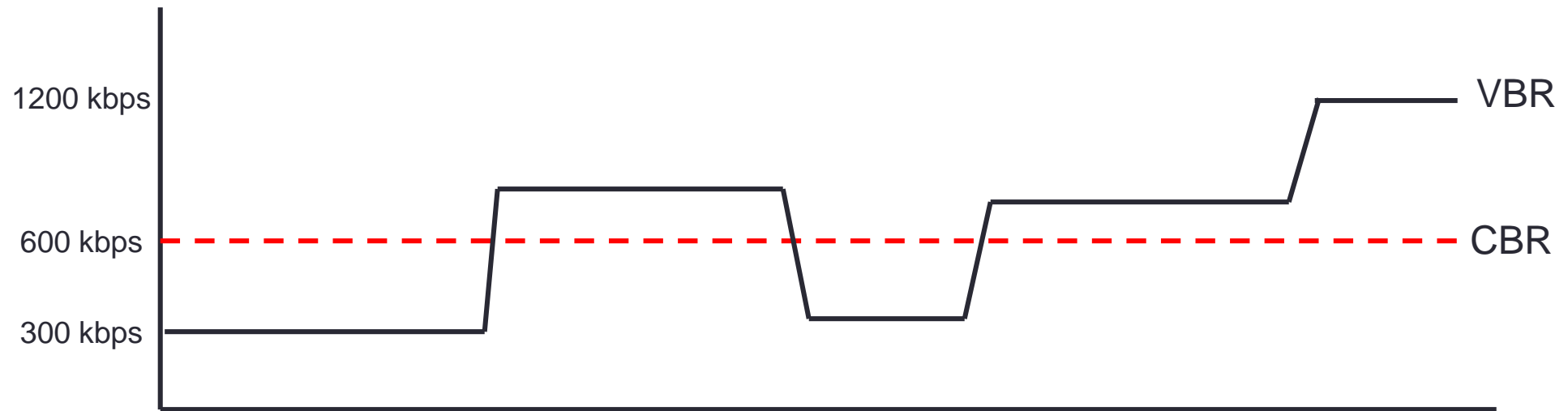
Bottom Line

- For most applications, 6 second segment size is fine

Bitrate Control

- Constant Bitrate (CBR) vs. Variable Bitrate (VBR)
- Producing top quality VBR and CBR
- When to use CBR and VBR

Bitrate Control Alternatives Constant (CBR) vs. Variable Bit Rate (VBR)



CBR File Illustrated



603 kbps
Average

- Faint (sorry) wavy blue line is data rate
- Relatively consistent throughout

VBR File Illustrated

596 kbps
Average



- Faint (sorry) wavy blue line is data rate
- Varies with scene complexity

Constant Bitrate

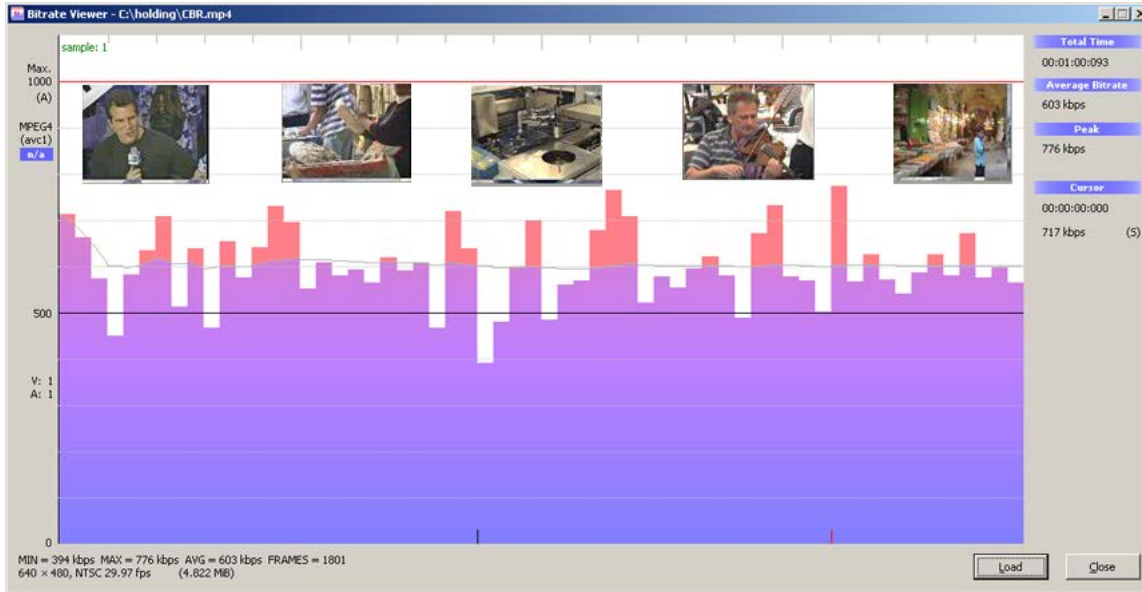
- Defined: One bit rate applied to entire video, irrespective of content
- Pros:
 - Computationally easy
 - Fast - one pass will do it
- Cons: Doesn't optimize quality

Variable Bitrate

- Defined: Dynamic bit rate matches motion in video
- Pros: Best quality
- Cons:
 - Need two or more passes
 - Can produce stream with variability

CBR vs. VBR

CBR



- Which file is easier to deliver over fixed bandwidth connections?
 - CBR

VBR



- Which file streams more reliably over changing conditions?
 - CBR

Adaptive - VBR vs. CBR

- Adaptive—most pundits recommend CBR
 - More consistent stream
 - Fewer encoding-related stream switches
- In practice—many producers use constrained VBR
 - Some as high as 200% (MTV)
 - Obviously, they wouldn't if this caused problems

Choosing Between VBR and CBR

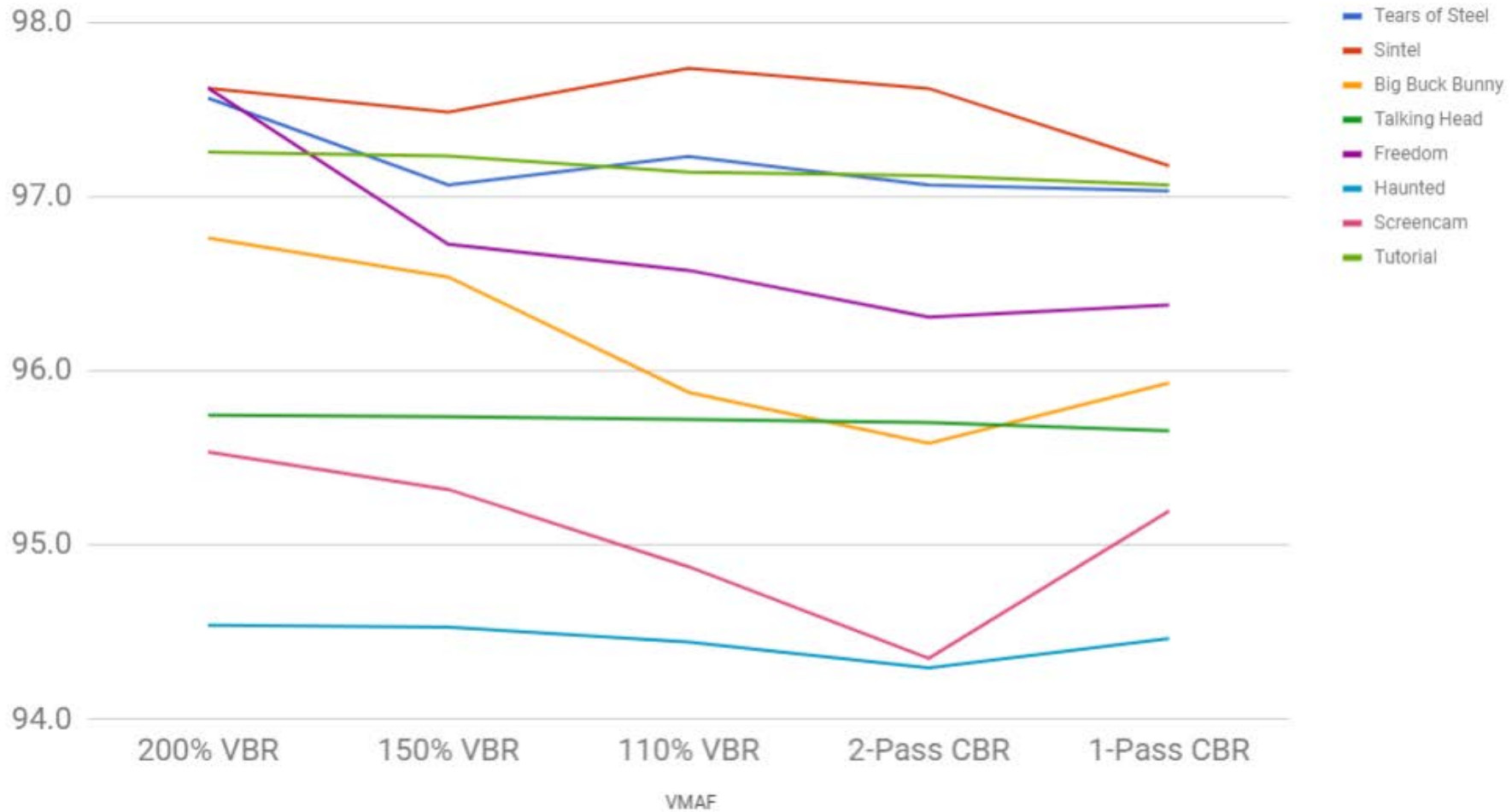
- Getting objective
- Overall quality
- Transient quality
- Deliverability

How Much Better Quality is VBR over CBR?

VMAF	200% VBR	150% VBR	110% VBR	2-Pass CBR	1-Pass CBR	Total Delta	Delta 110%-200%
Tears of Steel	97.6	97.1	97.2	97.1	97.0	0.53	-0.34
Sintel	97.6	97.5	97.7	97.6	97.2	0.56	0.11
Big Buck Bunny	96.8	96.5	95.9	95.6	95.9	1.18	-0.89
Talking Head	95.7	95.7	95.7	95.7	95.7	0.09	-0.03
Freedom	97.6	96.7	96.6	96.3	96.4	1.32	-1.05
Haunted	94.5	94.5	94.4	94.3	94.5	0.24	-0.10
Screencam	95.5	95.3	94.9	94.3	95.2	1.18	-0.66
Tutorial	97.3	97.2	97.1	97.1	97.1	0.19	-0.12
Average	96.6	96.3	96.2	96.0	96.1	0.58	-0.38

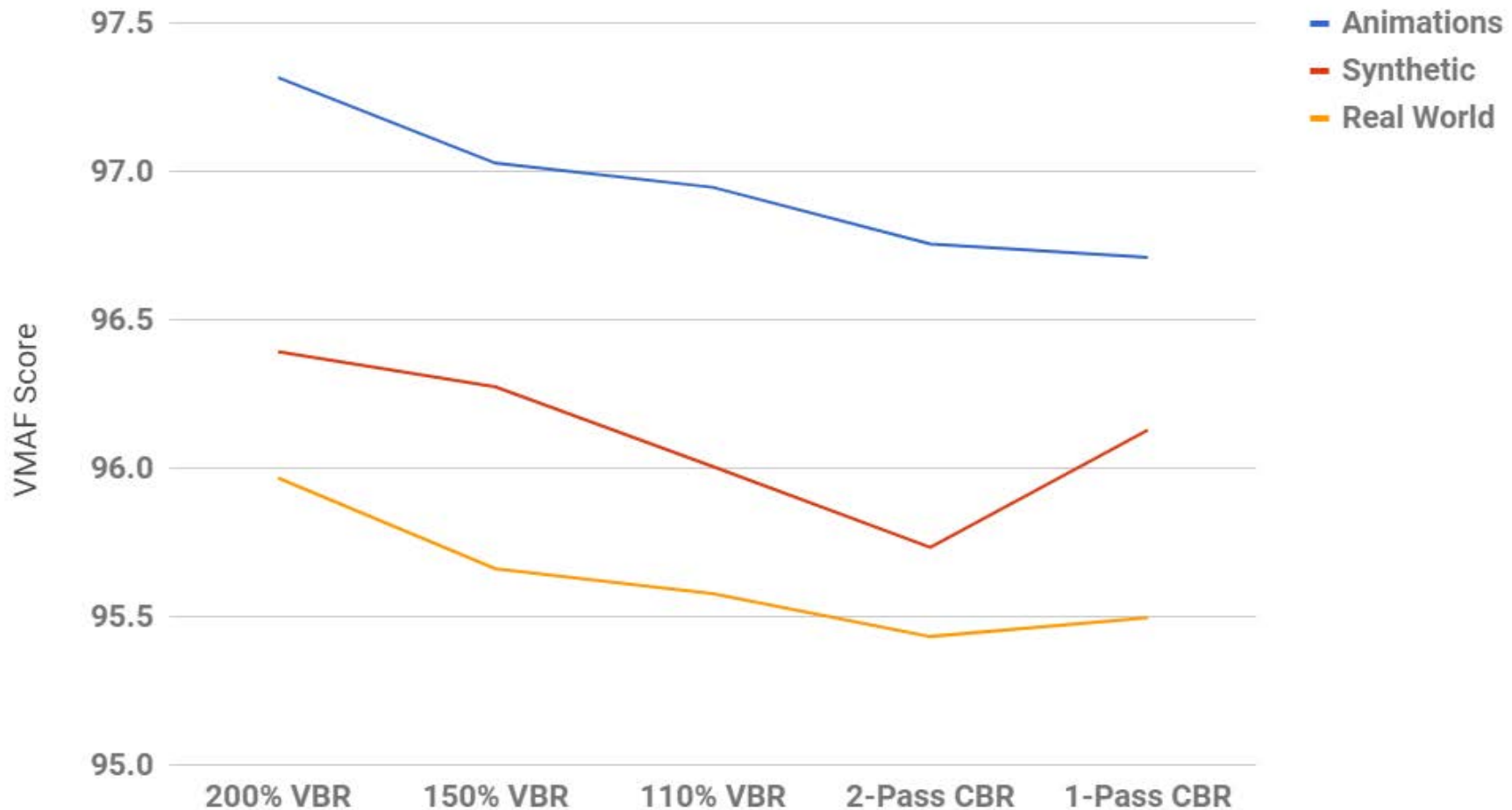
- Across the spectrum of different types of content
 - 200% CVBR always the highest
 - CBR always the lowest
- Total quality differential is minimal (JND is 6 points)

Effects of Bitrate Control on Overall VMAF Quality



- Not as substantial as you would think

Effects of Bitrate Control on Overall VMAF Quality



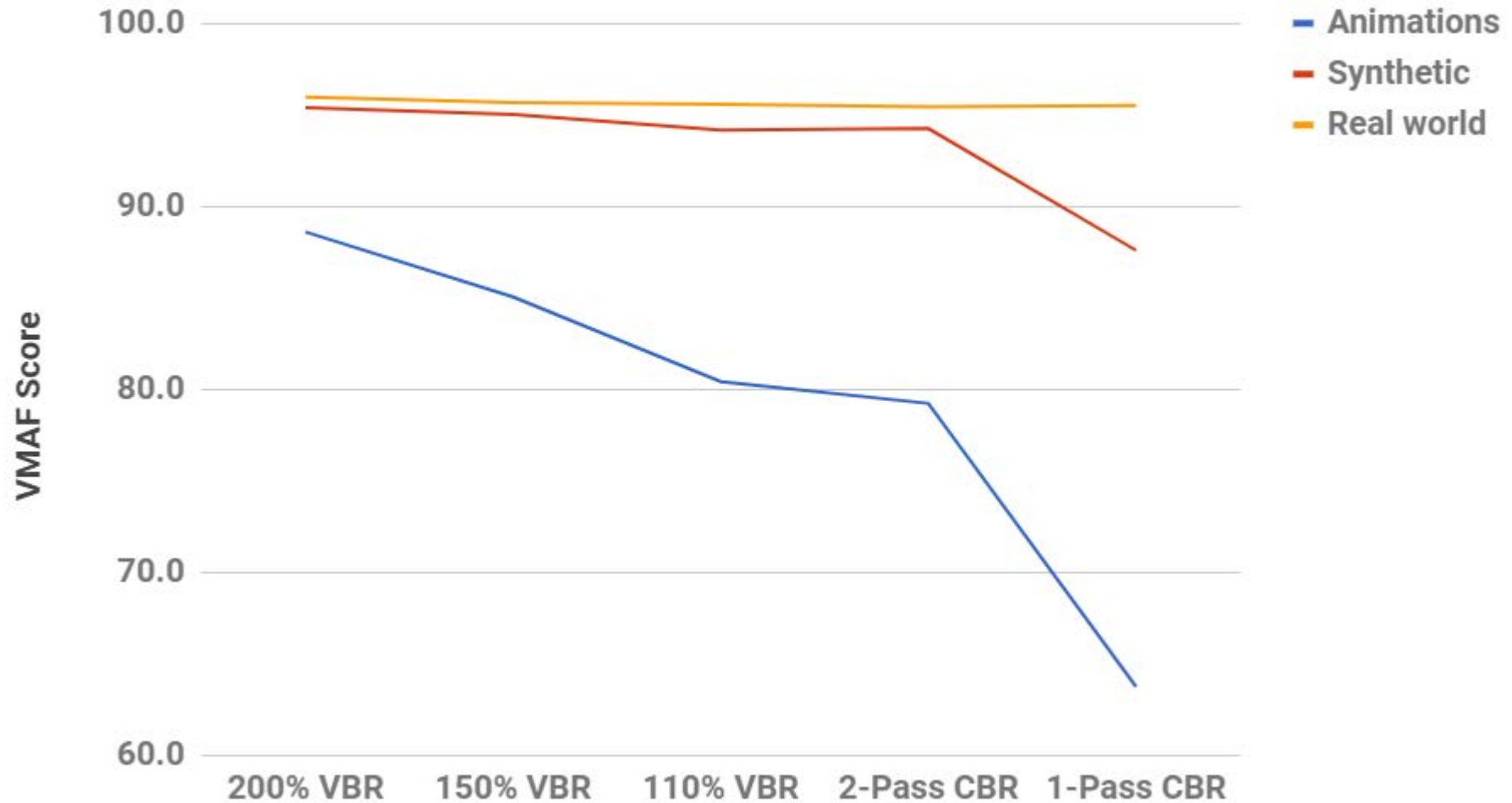
- By class – still not a big deal
 - Over 93 is good enough

Low Frame Quality

Low Frame Quality	200% VBR	150% VBR	110% VBR	2-Pass CBR	1-Pass CBR	Total Delta	Delta 110%-200%
Tears of Steel	89.4	84.5	79.7	78.6	79.0	10.81	-9.70
Sintel	88.4	87.6	86.3	86.3	50.3	38.07	-2.09
Big Buck Bunny	87.9	83.0	75.2	72.8	61.8	26.12	-12.75
Talking Head	93.0	92.7	91.9	91.3	91.3	1.68	-1.06
Freedom	90.9	90.7	87.9	84.5	83.7	7.22	-2.99
Haunted	82.1	81.2	78.0	78.6	79.1	4.15	-4.15
Screencam	94.6	94.4	94.2	94.3	93.6	0.93	-0.32
Tutorial	96.2	95.6	94.1	94.2	81.5	14.69	-2.15
Average	90.3	88.7	85.9	85.1	77.6	12.77	-4.40

- Across the spectrum of different types of content
 - 200% CVBR always the highest
 - CBR always the lowest
- Huge difference in low frame quality (JND is 6 points)

Effects of Bitrate Control on Low Frame VMAF Quality



- Creates significant issues with low frame quality, particularly in animations

Transient Quality Issues (ugly frames)

- Moscow University metric visualization
 - Red is CBR; Green is VBR
 - Circled areas shows very significant quality delta
 - Click “show frame” to see



Here's What Those Valleys Look Like



Analysis

- Transient differences like this are:
 - Much more likely in high motion files with significant scene variability
 - Rare
 - Short (1-2 frames)
- That said, VBR
 - Avoids this problem
 - Produces slightly better quality overall

Deliverability

- Research study
- Compared playback efficiency of CBR and 200% constrained VBR files
 - Mixed talking head and ballet footage
 - Worst case experience
- Restricted playback – used tool called Charles Debugging Suite to limit bandwidth during playback

Video Streams			
Codec	Width	Height	Bitrate (kbps)
H.264	1920	1080	4500 (VBR)
H.264	1280	720	3100 (VBR)
H.264	1280	720	2100 (VBR)
H.264	960	540	1500 (VBR)
H.264	640	360	1000 (VBR)
H.264	480	268	550 (VBR)
H.264	320	180	260 (VBR)

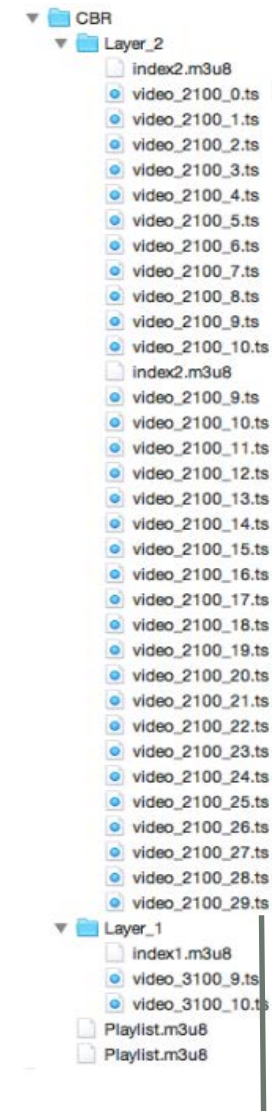
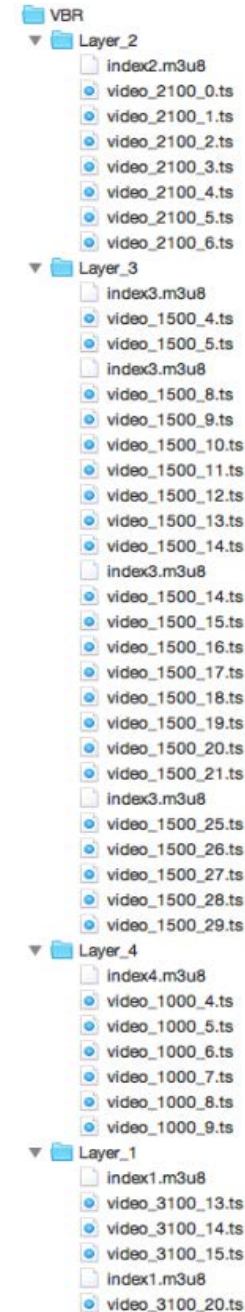
Our Findings

- Throttle to 3200

Layer	Bitrate	VBR - Safari			CBR - Safari		
		Segs	%	Bandwidth	Segs	Percent	Bandwidth
0	4500		0%	0		0%	0
1	3100	11	22%	34,100	2	6%	6,200
2	2100	7	14%	14,700	32	94%	67,200
3	1500	22	45%	33,000		0%	0
4	1000	9	18%	9,000		0%	0
5	550		0%	0		0%	0
6	260		0%	0		0%	0
Total		49	100%	90,800	34	100%	73,400

More bandwidth
(repeat packets) (\$\$\$)

Better QoE



Higher quality, more
consistent experience

Throttle to 4500 – Playback in Safari

- Throttle to 4500

Layer	Bitrate	VBR - Safari			CBR - Safari		
		Segs	%	Bandwidth	Segs	Percent	Bandwidth
0	4500		0%	0		0%	0
1	3100	7	14%	21,700	30	88%	93,000
2	2100	25	51%	52,500		0%	0
3	1500	6	12%	9,000		0%	0
4	1000		0%	0		0%	0
5	550		0%	0		0%	0
6	260		0%	0		0%	0
Total		38	78%	83,200	30	88%	93,000

More switching, many
lower-quality segments
(reduce QoE)

More consistent quality
Better QoE

Playback m3u8 in Safari

Throttle to 4500 – JW Player

		VBR - JWPlayer			CBR - JWPlayer		
Layer	Bitrate	Segs	Percent	Bandwidth	Segs	Percent	Bandwidth
0	4500		0%	0		0%	0
1	3100	7	18%	21,700	30	88%	93,000
2	2100	25	63%	52,500		0%	0
3	1500	5	13%	7,500		0%	0
4	1000		0%	0		0%	0
5	550		0%	0		0%	0
6	250		0%	0		0%	0
Total		37	93%	81,700	30	88%	93,000

More switching, many lower-quality segments (reduce QoE)

More consistent quality
Better QoE

What Apple Says and Does

- Initial version of TN2224 mandated no more than 110% constrained VBR
- Apple Authoring Spec as 200% Constrained VBR is OK (http://bit.ly/hls_spec_2017)

1.30. For VOD content the peak bit rate SHOULD be no more than 200% of the average bit rate.

- Apple's bitrate ladder ~ 110% CBR (Do as I do, not as I say)

HEVC	Width	Height	FPS	(Peak) Bandwidth	Average Bandwidth	VBR Constraint
1080p_h	1920	1080	60	6,472,228	5,913,163	109%
1080p_m	1920	1080	60	5,235,899	4,609,073	114%
1080p_l	1920	1080	60	3,887,770	3,249,312	120%
720p	1280	720	60	2,572,701	2,443,933	105%
540p	960	540	60	1,972,328	1,774,314	111%
432p	768	432	30	1,034,255	946,612	109%
360p	640	360	30	709,770	637,339	111%
270p	480	270	30	356,927	330,229	108%
234p	416	234	30	148,713	122,941	121%
Average						112%

(<https://developer.apple.com/streaming/examples/>)

Conclusions – Generic Recommendations

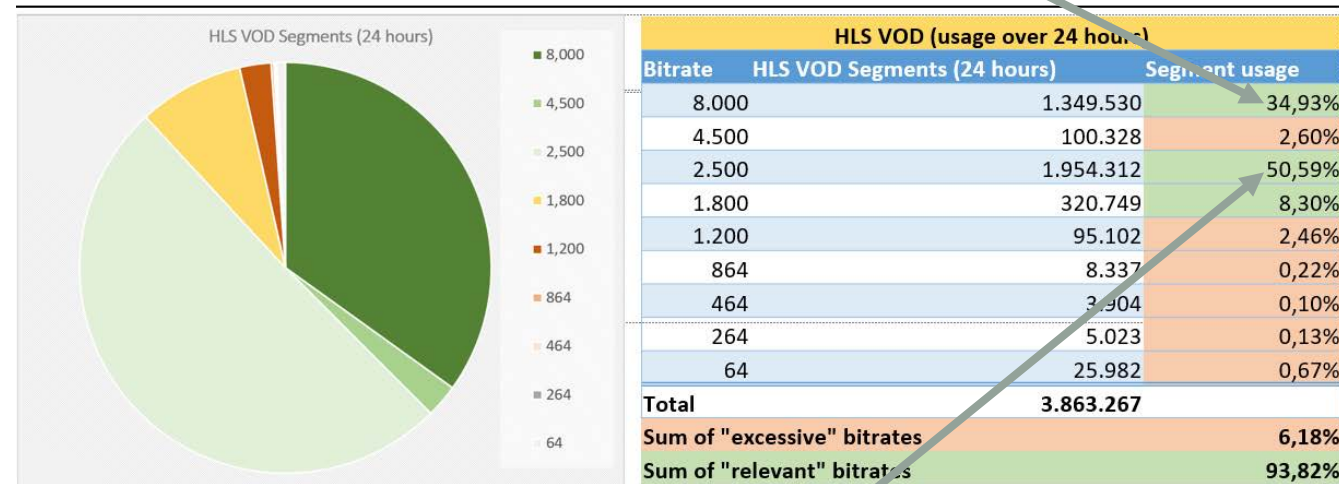
- 200% constrained VBR may reduce QoE when delivered over constrained conditions
- CBR best for overall QoE
 - But, has transient quality issues
 - 110%-150% constrained VBR is the best compromise

Even Better-Check Your Log Files

- In many European countries, bandwidth is so high that highest quality streams are predominantly retrieved
 - 200% constrained VBR only impacts deliverability when bandwidth is constrained
 - Go for quality – and use 200% CVBR
- In contrast, if users access all files equally (third-world) be more conservative and use 110% constrained VBR

35% - highest quality HD file

VOD using HLS (all clients)



51% - highest quality SD file

Which Layer Should You Play First

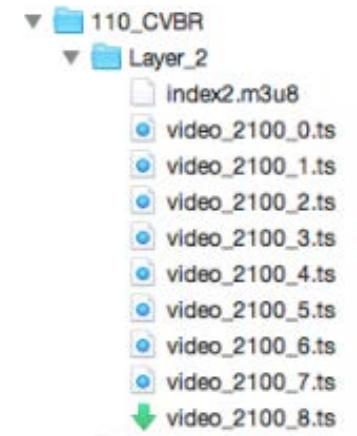
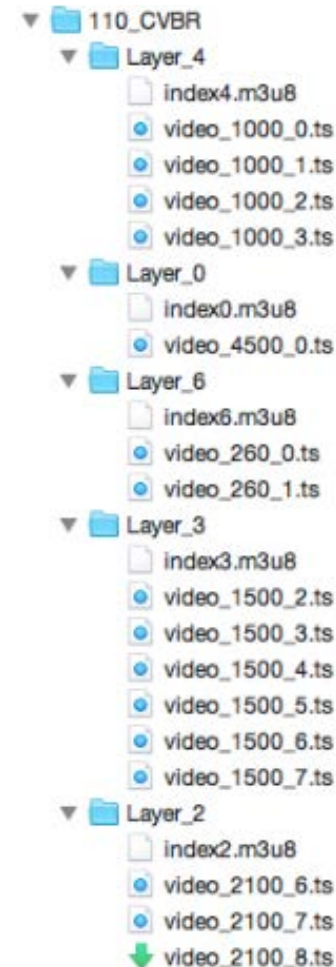


```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=5331166,CODECS="mp4a.40.2, avc1.4d401f"
HD_ProRes_PCM-Broadband%20High.segments/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=2840376,CODECS="mp4a.40.2, avc1.4d401e"
HD_ProRes_PCM-Broadband%20Low.segments/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=513855,CODECS="mp4a.40.2, avc1.42e015"
HD_ProRes_PCM-Cellular%20High.segments/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=255232,CODECS="mp4a.40.2, avc1.42e015"
HD_ProRes_PCM-Cellular%20Low.segments/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1396773,CODECS="mp4a.40.2, avc1.4d401e"
HD_ProRes_PCM-Wi-Fi%20High.segments/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=788787,CODECS="mp4a.40.2, avc1.42e01e"
HD_ProRes_PCM-Wi-Fi%20Low.segments/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=72265,CODECS="mp4a.40.2"
HD_ProRes_PCM-Audio%20for%20HTTP%20Live%20Streaming.segments/prog_index.m3u8
```

- With HLS, the player automatically retrieves the first file listed in the master manifest
 - Many encoders use the encoding order; usually top down, in the master
- See above: the highest quality file (5.3 mbps) is listed first
- What happens if the player doesn't have sufficient bandwidth?

A Disastrous Initial Experience

- Description – Same encoded files in both trials
 - Left – Layer 0 is first (4500)
 - Right – Layer 2 (2100)
 - Constrain at 3200
 - Play till segment 8
- Observation
 - Layer 0 first; player switched 5 times before stabilizing
 - Layer 2 first; no switches; stable playback
 - More packets, lower QoE when choose wrong starting point
- Conclusions
 - First file selected should be sustainable
 - Should change depending upon connection



Apple Recommendations for HLS

- Wi-Fi – 2 mbps stream first
- Cellular – 760 kbps stream first
- Implementation?
 - Create two masters for same set of HLS packaged files
 - Desktop/OTT/Smart TV
 - Mobile
 - Query player and send m3u8 accordingly.

Questions?

Should be 11:20

Lesson 9: Encoding with H.264

- Introduction to H.264
- Profiles and levels

What H.264 Is and Why It's Important

- H.264 is a codec defined in Part 10 of the MPEG-4 specification
- Jointly sponsored by MPEG and ISO standards bodies
 - That's why it's **H.264** and **AVC**

About x264

- Because H.264 is a standard, there are many compliant codecs
 - Apple, MainConcept, Intel, NVIDIA, Ittiam, many others
 - x264 is the open-source encoder included with FFmpeg
 - Widely agreed to be the highest quality H.264 codec

Critical H.264 Encoding Parameters

- Some parameters apply to all H.264 codecs
 - Profiles, levels, Entropy Coding
 - No matter which H.264 codec you work with, you'll have to set these
- Some only apply to x264
 - Presets (x264's way of balancing encoding time and quality)
 - Other codecs may have a similar control, but only x264 (and x265) has presets called slow, very slow, etc.

H.264 Profiles

- What profiles are and why they exist
- Compatibility aspects
- Quality aspects

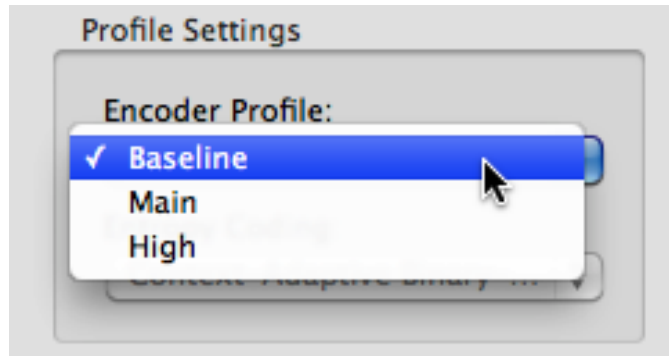
What Profiles are and why they Exist

- Profiles enable different encoding techniques to balance decoding complexity
- Baseline uses the fewest, so is easiest to decode
 - Early video-capable iPods only supported the Baseline codec
- High uses the most, so is the hardest to decode
 - All computers, mobile devices, TVs, STBs manufactured in the last four years can play the High profile

	Baseline	Main	High
I and P Slices	Yes	Yes	Yes
B Slices	No	Yes	Yes
Multiple Reference Frames	Yes	Yes	Yes
In-Loop Deblocking Filter	Yes	Yes	Yes
CAVLC Entropy Coding	Yes	Yes	Yes
CABAC Entropy Coding	No	Yes	Yes
Interlaced Coding (PicAFF, MBAFF)	No	Yes	Yes
8x8 vs. 4x4 Transform Adaptivity	No	No	Yes
Quantization Scaling Matrices	No	No	Yes
Separate Cb and Cr QP control	No	No	Yes
Separate Color Plane Coding	No	No	No
Predictive Lossless Coding	No	No	No
	Baseline	Main	High

Encoding

- Profiles/Levels
 - Most critical ***compatibility-related*** setting
 - Encode using wrong profile, file won't play on target device
 - Profile is available on all encoding tools
- Don't exceed profile of target device
 - Exclusively a concern with older mobile
 - Computers and OTT devices can play High profile (any level)



Profiles and Quality

VMAF-Average	Baseline	Main	High	Delta - Baseline/Main	Delta - Main/High	Total Delta
Tears of Steel	92.83	95.46	96.23	2.83%	0.80%	3.66%
Sintel	93.65	95.78	96.38	2.27%	0.63%	2.91%
Big Buck Bunny	92.14	94.72	95.52	2.80%	0.83%	3.67%
Talking Head	94.35	94.93	95.19	0.61%	0.28%	0.90%
Freedom	92.87	94.65	95.36	1.91%	0.74%	2.67%
Haunted	89.56	91.11	91.99	1.73%	0.95%	2.70%
Screencam	92.80	94.01	94.34	1.30%	0.35%	1.66%
Tutorial	95.85	96.13	96.15	0.29%	0.03%	0.32%
Average	93.01	94.60	95.14	1.72%	0.57%	2.31%

- High is always the best; Baseline always the worst
 - Jump from Baseline > Main more significant than Main > High
- Difference is greater in hard to encode files
 - TOS – 3.66%
 - Talking Head – .9%

iOS History Lesson

Width	Height	Frame Rate	Video Bitrate	Audio Bitrate	I-Frame	Profile	B-frames	Segment Size	iPod Touch 2-4	iPod Touch 5	iPhone 3G, 3GS, 4	iPhone 4S, 5, 5C, 5S	iPad 1,2	iPad 3, 4, 5	Apple TV 2	Apple TV 3
416	234	12	200	64	36	Baseline	NA	9	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
480	270	15	400	64	45	Baseline	NA	9	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
640	360	29.97	600	64	90	Baseline	NA	9	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
640	360	29.97	1200	96	90	Baseline	NA	9		Yes		Yes	Yes	Yes	Yes	Yes
960	540	29.97	3500	96	90	Main	As needed	9		Yes		Yes	Yes	Yes	Yes	Yes
1280	720	29.97	5000	128	90	Main		9		Yes		Yes	Yes	Yes	Yes	Yes
1280	720	29.97	6500	128	90	Main		9		Yes		Yes	Yes	Yes	Yes	Yes
1920	1080	29.97	8500	128	90	High		9		Yes		Yes		Yes		Yes

- Initial version of TN2224 customized profile for different targets

Current HLS Authoring Specs Abandon Legacy Devices

HDR (HEVC) 30 fps	HEVC/H.265 30 fps	H.264/AVC	Resolution 16:9 aspect ratio	Frame rate
160	145	145	416 x 234	≤ 30 fps
360	300	365	480 x 270	≤ 30 fps
800	660	730	640 x 360	≤ 30 fps
1200	990	1100	768 x 432	≤ 30 fps
2050	1700	2000	960 x 540	same as source
2900	2400	3000	1280 x 720	same as source
3850	3200	4500	1280 x 720	same as source
5400	4500	6000	1920 x 1080	same as source
7000	5800	7800	1920 x 1080	same as source
9700	8100	n/a	2560 x 1440	same as source
13900	11600	n/a	3840 x 2160	same as source
20000	16800	n/a	3840 x 2160	same as source

- Significant change:
 - Expect all to play High profile
 - Keyframe – 2 seconds
 - Segment size – 6 seconds
 - Still 200% constrained VBR
 - **Class poll**

http://bit.ly/A_Devices_Spec

Encoding for Android Devices

Table 2. Examples of supported video encoding parameters for the H.264 Baseline Profile codec.

	SD (Low quality)	SD (High quality)	HD 720p (N/A on all devices)
Video resolution	176 x 144 px	480 x 360 px	1280 x 720 px
Video frame rate	12 fps	30 fps	30 fps
Video bitrate	56 Kbps	500 Kbps	2 Mbps
Audio codec	AAC-LC	AAC-LC	AAC-LC
Audio channels	1 (mono)	2 (stereo)	2 (stereo)
Audio bitrate	24 Kbps	128 Kbps	192 Kbps

- Android support is bifurcated
 - In OS software – Baseline profile only
 - In hardware/device supplied software, up to High
- Google recommends using Baseline (bit.ly/androidvideospecs)
 - Ignored by many
- **Class poll?**

How Much Quality Difference?

Talking Head	Data Rate	Baseline	High	Delta		Haunted	Data Rate	Baseline	High	Delta
234p	145,000	33.79	34.20	1.22%		234p	145,000	30.46	31.56	3.61%
270p	350,000	35.72	35.99	0.75%		270p	365,000	33.14	33.73	1.79%
360p	600,000	38.16	38.37	0.54%		360p	900,000	35.99	36.38	1.10%
540p	1,000,000	40.04	40.33	0.71%		540p	1,500,000	38.09	38.62	1.38%
720p	1,500,000	40.78	41.32	1.34%		720p	2,500,000	39.28	39.84	1.42%
1080p	2,500,000	43.53	44.11	1.34%		1080p	6,000,000	41.31	41.86	1.32%
Average		38.67	39.05	0.98%		Average		36.38	37.00	1.77%

- Talking head on left, DSLR movie footage on right
- FFmpeg/x264/New **TN2224/PSNR**
- Very minor difference at all configurations

Encoding for Mobile - Choices

- Ignore older devices – all high profile
- Or, one set of files – mixed baseline, main, high, for all targets
 - Cheapest, easiest
 - May be leaving some quality on the table
- Or, separate ABR groups customized for devices:
 - Baseline – old iOS and Android
 - Main – old iOS and Android
 - High – new iOS, computers and OTT
 - Optimal quality, but more encoding, storage and administrative costs

Conclusions

- More and more, it seems as if publishers DON'T customize streams for different targets; either:
 - Go High profile and abandon legacy (really iPhone 4 and previous)
 - Use one set of streams with mixed profiles
- Justification
 - Quality isn't that different

Questions

Should be 11:30

Lesson 10: Encoding HEVC

- About HEVC
- HEVC profiles

What HEVC Is and Why It's Important

- HEVC is a standards-based compression technology
- Jointly sponsored by MPEG and ISO standards bodies
 - That's why it's **HEVC** and **H.265**
- Available in
 - iOS/Mac/Apple TV (end 2017)
 - Android
 - Windows 10/Edge if the system features hardware acceleration
- Not supported in Chrome, Firefox, Opera, or Internet Explorer

Critical HEVC Encoding Parameters

- Some parameters apply to all HEVC codecs
 - Profiles
 - No matter which HEVC codec you work with, you'll have to set these

What Profiles are and Why They Exist

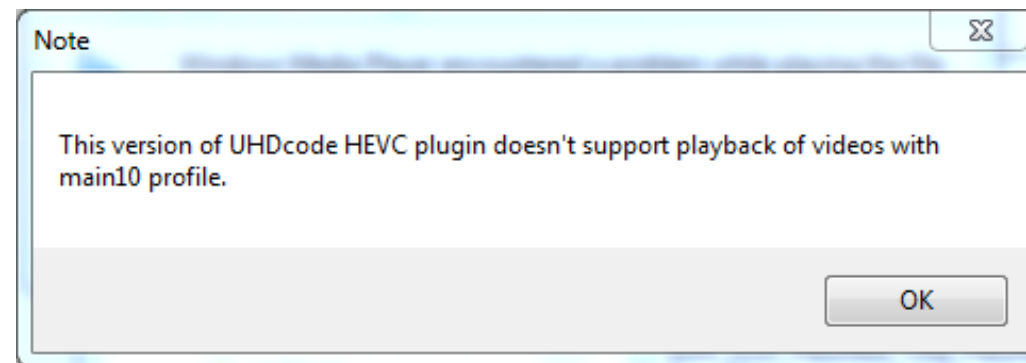
- Profiles enable different encoding techniques to balance decoding complexity
 - Version 2 codecs use more advanced features
- Today, FFmpeg outputs Main and Main 10
 - Need different FFmpeg builds for each
 - Primary difference is bit depth

Feature	Version 1	
	Main	Main 10
Bit depth	8	8 to 10
Chroma sampling formats	4:2:0	4:2:0
4:0:0 (Monochrome)	No	No
High precision weighted prediction	No	No
Chroma QP offset list	No	No
Cross-component prediction	No	No
Intra smoothing disabling	No	No
Persistent Rice adaptation	No	No
RDPCM implicit/explicit	No	No
Transform skip block sizes larger than 4x4	No	No
Transform skip context/rotation	No	No
Extended precision processing	No	No

https://en.wikipedia.org/wiki/High_Efficiency_Video_Coding

Main or Main10?

- Main players can't play Main 10 encoded content
 - Some early HEVC players are Main only
 - If encoding for general-purpose playback, use Main
- Main 10 has a very slight quality advantage
 - If encoding for Main10 player, use Main 10
 - Main 10 players can play Main



720p - x265	Main	Main 10	Delta
Tears of Steel	37.05	37.73	1.84%
SIintel	41.37	41.25	-0.29%
Big Buck Bunny	37.21	37.16	-0.13%
Talking Head	41.15	41.15	0.00%
Freedom	39.70	39.57	-0.31%
Haunted	39.56	41.78	5.61%
Average	39.34	39.77	1.12%

Encoding for iOS Devices – HEVC

HDR (HEVC) 30 fps	HEVC/H.265 30 fps	H.264/AVC	Resolution 16:9 aspect ratio	Frame rate
160	145	145	416 x 234	≤ 30 fps
360	300	365	480 x 270	≤ 30 fps
800	660	730	640 x 360	≤ 30 fps
1200	990	1100	768 x 432	≤ 30 fps
2050	1700	2000	960 x 540	same as source
2900	2400	3000	1280 x 720	same as source
3850	3200	4500	1280 x 720	same as source
5400	4500	6000	1920 x 1080	same as source
7000	5800	7800	1920 x 1080	same as source
9700	8100	n/a	2560 x 1440	same as source
13900	11600	n/a	3840 x 2160	same as source
20000	16800	n/a	3840 x 2160	same as source

- WWDC – June 2017
 - Max is Main 10, Level 5
 - Must be fMP4
 - Should provide H.264 for backwards compatibility

http://bit.ly/A_Devices_Spec

Bottom Line – HEVC Profile

- If encoding solely for iOS - use Main 10
- If encoding for iOS and general purpose – consider Main

Questions

Should be 11:40

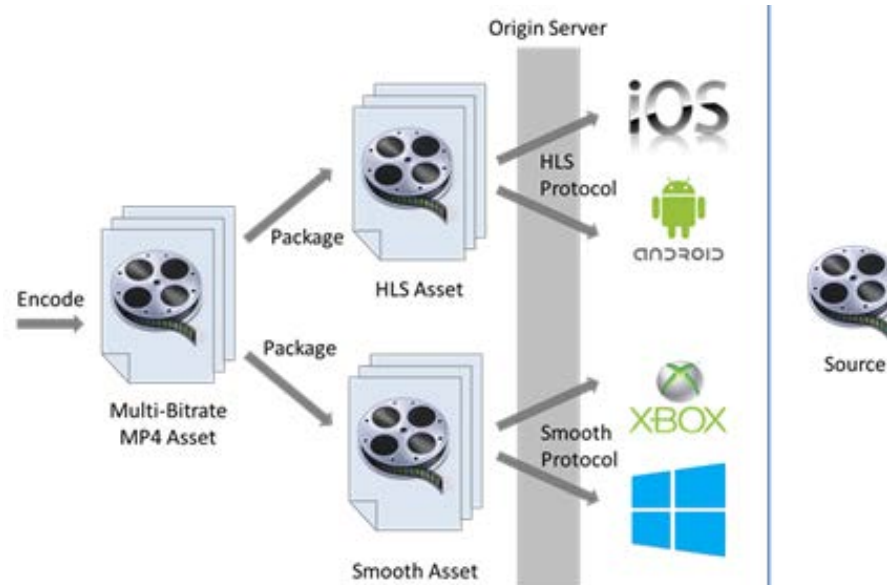
Lesson 11: Dynamic Packaging for VOD and Live

- Static vs. dynamic delivery
- Encoding for static delivery
 - Existing workflow
 - Encoding then packaging
 - Tool options
- Dynamic delivery
 - VOD
 - Live

Static vs. Dynamic Delivery

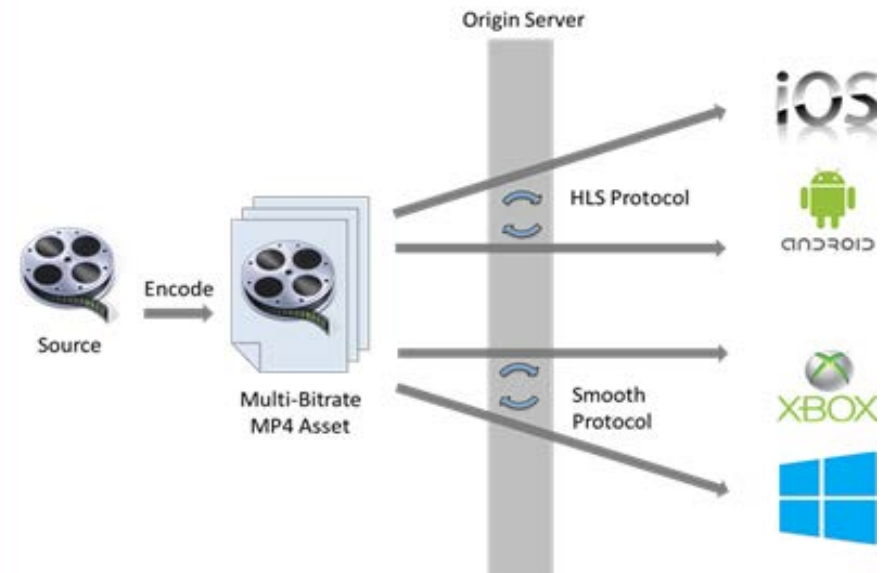
Static

- Create multi-bitrate MP4 files from mezz file
- Create ABR files from multi-bitrate files
- Upload ABR files to server
- Distribute ABR files from origin server



Dynamic

- Create multi-bitrate MP4 files and store on server
- Server dynamically creates ABR chunks and manifest files as needed



Static vs. Dynamic Delivery

Static: Pros/Cons

- Pros
 - Simple, no streaming server required
- Cons
 - Storage intensive
 - Major effort to support new formats
 - Must create new packaged files
 - Upload to servers

Dynamic: Pros/Cons

- Pros
 - Storage efficient
 - Very simple to support new formats/devices down the road
- Cons
 - More technically complex
 - May be more expensive
 - If server component costs more than extra storage + encoding

Static vs. Dynamic

- Consulting project; cloud encoding for library and ongoing
 - Static – increased encoding and storage costs
 - Dynamic – increased server costs (Wowza + cloud instance), but much cheaper overall

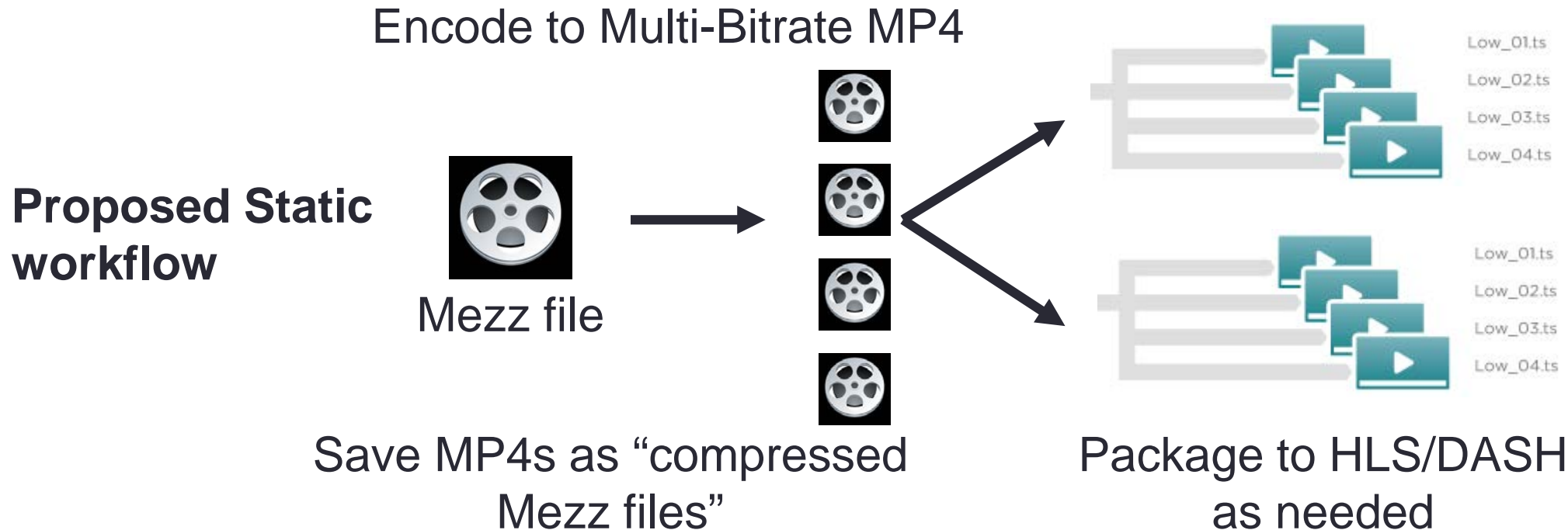
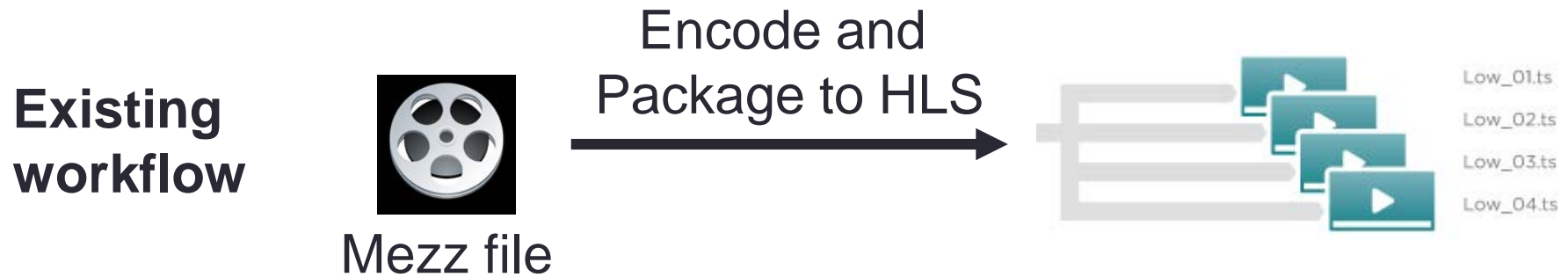
Two Year Projections	Static ABR/ Encode Library ASAP	Dynamic ABR - encode library ASAP	Delta
Ongoing cloud encode	\$68,651	\$49,914	-\$18,737
Wowza		\$20,405	\$20,405
Extra storage for HLS/DASH	\$34,009		-\$34,009
Total ongoing	\$102,660	\$70,319	-\$32,341
One Time Library Conversion	\$125,091	\$55,114	-\$69,977
Total	\$227,751	\$125,433	-\$102,318



Updating the Static File Creation Workflow

- If static selected, need to update encoding workflow to leverage similar benefits
 - Typical existing ***single-step*** workflow
 - Encode from mezz to final ABR formats
 - Complete re-encode needed to support new formats (like HLS > DASH migration)
 - More efficient ***two-step workflow***
 - ***Encode step*** - Encode from mez to MP4 (use as compressed mezz files)
 - ***Package step*** – transmux multi-bitrate MP4 files into ABR formats

Updating The Static Encoding Workflow



Updating the Static File Creation Workflow

- **Encode step** – encode mezz files into multi-bitrate MP4s:
 - Used as source for ABR files
 - This is the expensive, time-consuming step
 - Won't change when it's time to support new ABR formats
- **Package step** – transmux multi-bitrate MP4 files into ABR formats
 - This is fast and cheap
 - Easy to support new formats like DASH

Static Packaging VOD

Encoder

- Any desktop, enterprise or cloud encoder that can create MP4 files

DASH Packagers

- edash-packager
 - bit.ly/Dash_pack1
- MP4Box - <http://gpac.io>.
- Rebaca MPEG DASH Segmenter
 - http://bit.ly/Dash_pack2
- castLabs DASH Encrypt Packager
 - <https://github.com/castlabs/dashencrypt>
- Bento4 - www.bento4.com

Static Packagers

HLS Packagers

- Apple Media Stream Segmenter (MPEG-2 transport streams)
- Apple Media File Segmenter (MP4 inputs)
 - http://bit.ly/HLS_pack
- Apple Variant Playlist Creator
- FFmpeg – media playlists and packaging only
 - No master m3u8
- Bento4

Other Packagers

- Unified Packager (DASH, HLS, HDS, Smooth)
 - bit.ly/Uni_pack
- ProMedia Package (HLS, Smooth, HDS, DASH)
 - bit.ly/harm_pack

Dynamic Alternatives

DIY

- Wowza Streaming Engine
- Nimble Streamer
- Elemental Delta
- Azure Media Services
- encoding.com
- Brightcove
- Many others

Service Providers

- Akamai
- Limelight

What it Looks Like in Wowza

- Upload encoded video ladders in MP4 format
- Choose supported formats
 - DASH, HLS, RTMP, HDS, Smooth Streaming, RTSP/RTP
- Article: Dynamic Packaging with Wowza
 - http://bit.ly/wowza_dynamic

The screenshot displays the Wowza Streaming Engine Manager interface. On the left, a sidebar lists navigation options: '+ Add Application', 'SELECTED APPLICATION' (Tutorial), and categories for Monitoring, Wowza Player, Playback Security, SMIL Files, and DRM. Below these are 'LIVE APPLICATIONS' (live) and 'VOD APPLICATIONS' (Tutorial, vod, vods3). The main panel is titled 'Tutorial' and includes tabs for 'Setup', 'Properties', and 'Modules'. An 'Edit' button is visible. The 'Application Description' is set to '-Not Set-'. Under 'Playback Types', several options are checked: MPEG-DASH, Apple HLS, Adobe RTMP, Adobe HDS, Microsoft Smooth Streaming, and RTSP/RTP. To the right of these, arrows point to labels: DASH, HLS, RTMP, HDR, Smooth Streaming, and RTSP. The 'Options' section shows 'Cross-origin resource sharing (CORS)' is checked. The 'Content Directory' is set to a default path. The 'Closed Caption Sources' section shows 'Embedded 3GPP / MPEG-4 Timed Text track' is checked, while 'Timed Text (TTML / DXFP)', 'SubRip (SRT)', and 'Web Video Text Track (WebVTT)' are unchecked.

Tutorial
Video on Demand Single Server or Origin

Setup Properties Modules

Edit

Application Description
-Not Set-

Playback Types

- ✓ MPEG-DASH → DASH
- ✓ Apple HLS → HLS
- ✓ Adobe RTMP → RTMP
- ✓ Adobe HDS → HDR
- ✓ Microsoft Smooth Streaming → Smooth Streaming
- ✓ RTSP/RTP → RTSP

Options

- ✓ Cross-origin resource sharing (CORS) (for HTTP)

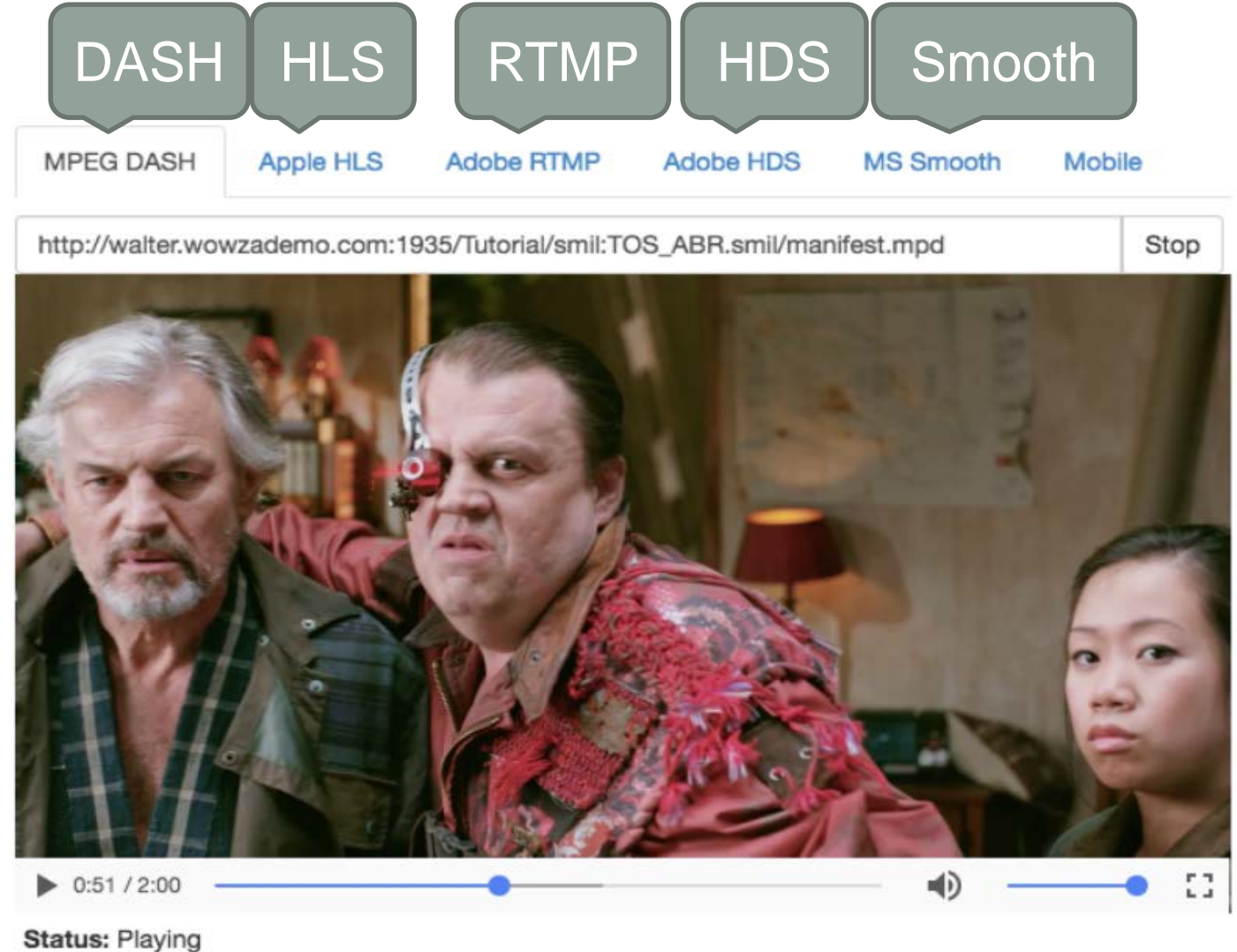
Content Directory
\${com.wowza.wms.context.VHostConfigHome}/

Closed Caption Sources

- ✓ Embedded 3GPP / MPEG-4 Timed Text track
- ✗ Timed Text (TTML / DXFP) file
- ✗ SubRip (SRT) file
- ✗ Web Video Text Track (WebVTT) file

What it Looks Like in Wowza

- Wowza supplies separate manifest URL for each
 - DASH
 - HLS
 - RTMP
 - HDS
 - Smooth Streaming
- Article: Dynamic Packaging with Wowza
 - http://bit.ly/wowza_dynamic



Live – Traditional Workflow

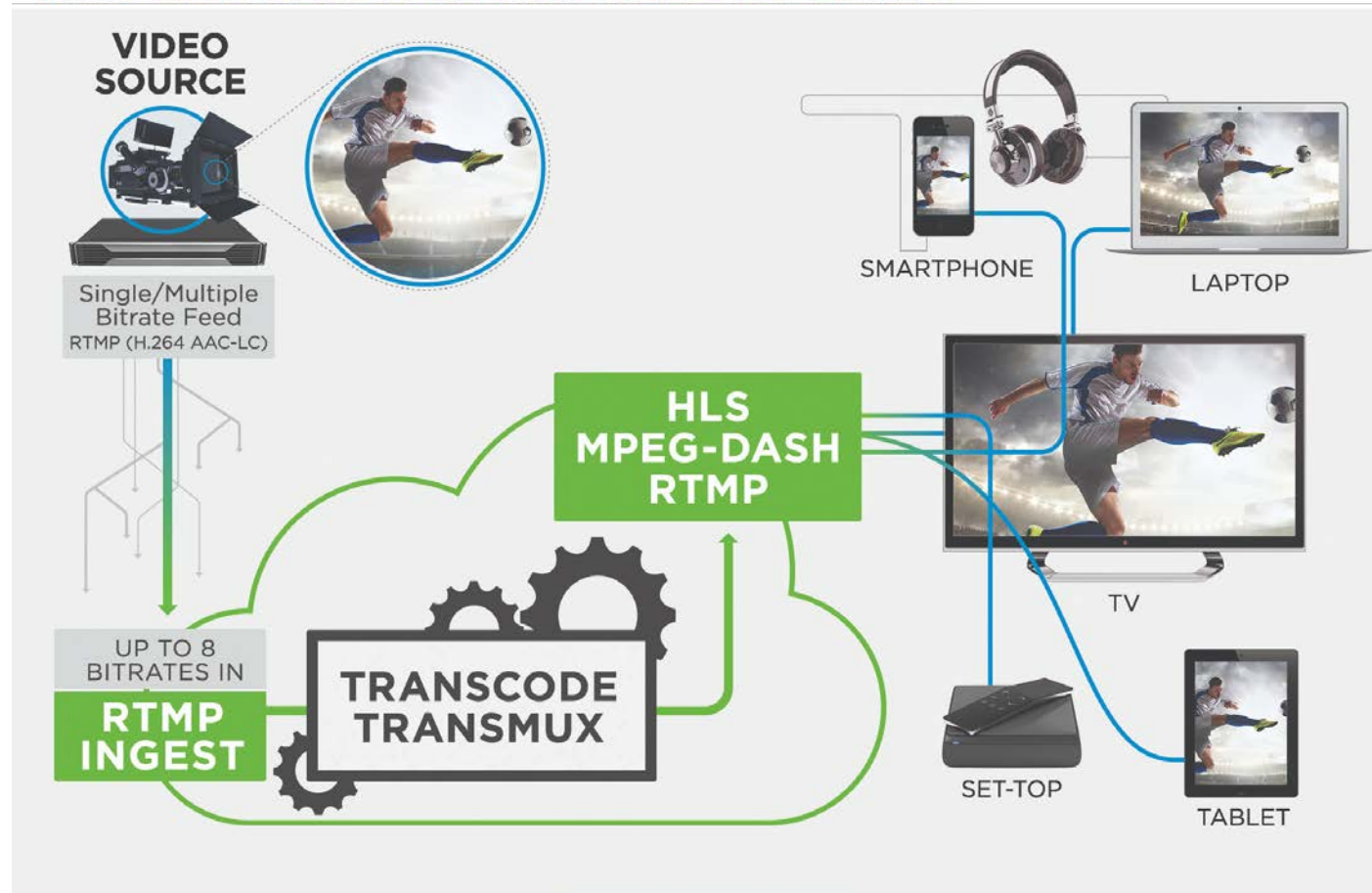


- Workflow
 - Create HLS/DASH/HDS on premise
 - Upload to CDN for distribution
- Pros
 - No transcoding fees (lower OPEX)
- Cons
 - Higher CAPX
 - Need more outbound bandwidth

Live – Transcode/Dynamic Packaging

- Workflow
 - Stream live to cloud
 - Create ladder in the cloud
 - Package as needed
- Pros
 - Lower CAPEX
 - Lower bandwidth requirements
- Cons
 - Higher OPEX
 - Software plus cloud instance

MMD LIVE SMALL CHUNK SIZE STREAMING



What it Looks Like in Wowza

- Create encoding ladder in the cloud
- Choose formats (as before)
- **Stream live video up to server**
- Get unique URL for each format (as before)

Encoding Presets



















Decoding Preset

Stream Name Groups

An encoding preset represents one resultant encoded bitrate in the output streams from the Transcoder. An adaptive bitrate stream has multiple presets. « Show

+ Add Preset

Presets

Enabled	Preset	Stream Name	Actions
<input checked="" type="checkbox"/>	source	mp4:\${SourceStreamName}_source	  
<input checked="" type="checkbox"/>	720p	mp4:\${SourceStreamName}_720p	  
<input checked="" type="checkbox"/>	360p	mp4:\${SourceStreamName}_360p	  
<input checked="" type="checkbox"/>	240p	mp4:\${SourceStreamName}_240p	  
<input checked="" type="checkbox"/>	160p	mp4:\${SourceStreamName}_160p	  
<input type="checkbox"/>	h263	mp4:\${SourceStreamName}_h263	  

Questions?