Encoding for performance on multiple devices

Fabio Sonnati | Adobe Community Professional

Introduction

H.264 | Encoding for performance on multiple devices

- **Fabio Sonnati** - Freelance media applications architect, ACP since 2006 FMS developer and beta tester since 2003 with expertise in Video Encoding optimizations. I develop or optimize video encoding & delivery platforms for big media clients.

Success Story:

- Designed and developed the encoding pipeline of La7.tv (Telecom Italia), the first Catch-up TV in Italy.
  (Flash, iOS, DGTVi, ConnectedTV)

Blog: http://sonnati.wordpress.com
Email: sonnati@progettosinergia.com
Twitter: @sonnati

H.264 | Encoding for Performance on multiple devices

- Encoding is “**Art**” as well as ”**Science**”
- A challenge to find a balance between **Business needs** and **Customer expectactions**
- The market is today much more complex due to multiple devices and higher expectations.

The quest for performance

Why the “performance” is so important ?

An encoding workflow has to satisfy Business and Customers needs

Business needs

Engage Customers

Monetize

Control delivery costs

Customer expectations

Video Quality

Flowless Experience

My recipe

How to maximize the performance ?

1. Understand Customers' Expectations
2. Analize all the boundary conditions
3. Leverage the Flash Video Ecosystem
4. Apply Encoding Best Practices
5. Apply Delivery Best Practices

Be ***creative***, ***adaptive***, ***dynamic*** to overcome limitations in each point above.

Know your audience

1. Understand your customers' expectations

Video quality often achieved with High Bitrate ?

↓

Higher engagement

↓

Better monetization

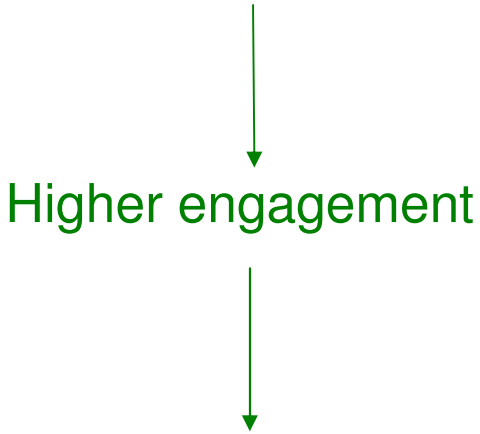
Know your audience

1. Understand your customers' expectations

Know your audience

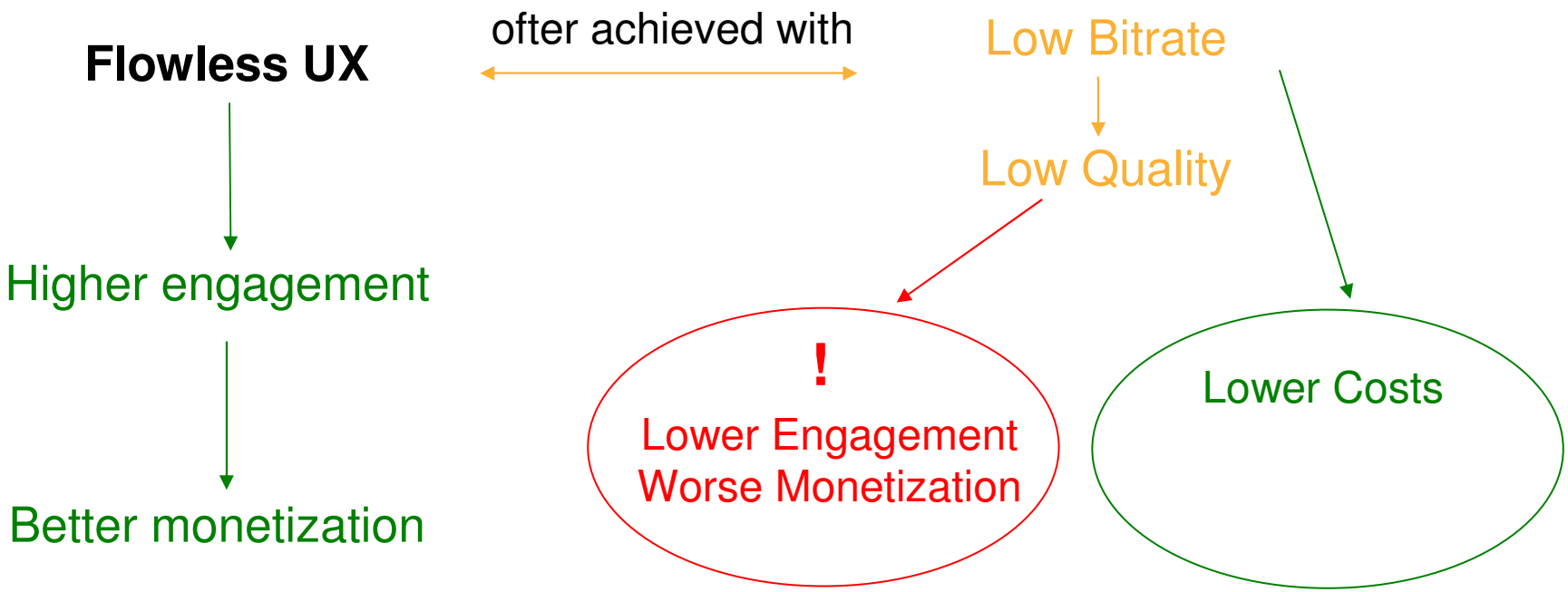
1. Understand your customers' expectations

Flowless UX often achieved with Low Bitrate ?



Know your audience

1. Understand your customers' expectations



Boundary conditions

2. Analyze all the boundary conditions

Now you have to address multiple devices and platforms:

Each with different levels of **processing power**, **HW acceleration**, **bandwidth**, **screen size** and **user interaction model**.

Boundary conditions

2. Analyze all the boundary conditions

Deskstops & Laptop

- Hi power
- Medium to Big screen
- Hi-bandwidth
- Mouse (accurate)

Smart phones & Tablets

- Low power
- Small to Medium screen
- Low to med bandwidth
- Touch (less accurate)

TV sets & STBs

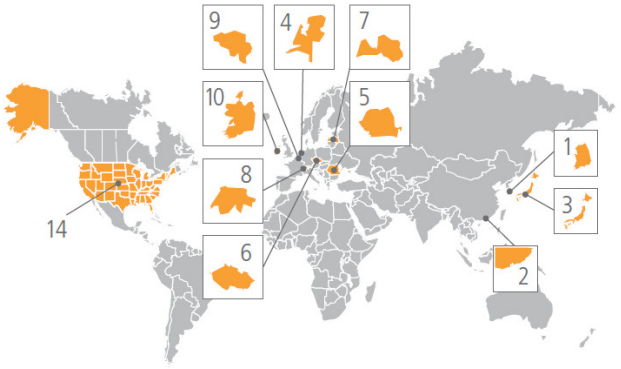
- Low power (HW Acc.)
- Big screen
- Hi bandwidth
- Remote control (coarse)

Boundary conditions

2. Analyze all the boundary conditions

User Bandwidth is a limited and expensive resource:

	Country/Region	Q1'11 Avg. Mbps	QoQ Change	YoY Change
–	Global	2.1	9.7%	23%
1	South Korea	14.4	5.0%	20%
2	Hong Kong	9.2	-1.7%	2.1%
3	Japan	8.1	-2.7%	2.7%
4	Netherlands	7.5	7.6%	25%
5	Romania	6.6	-4.9%	4.9%
6	Czech Republic	6.5	14%	19%
7	Latvia	6.3	6.7%	0.4%
8	Switzerland	6.2	10%	17%
9	Belgium	6.1	11%	29%
10	Ireland	5.6	16%	14%
…				
14	United States	5.3	4.7%	14%



It's constantly increasing (and cost decreasing) but :

Source: Akamai 2011

- Also quality expectations are growing (competition).
- Mobile bandwidth is much more limited and oscillating.
- 23% of users are under 2Mbit/s in USA and 38% Globally (61% below 5Mbit/s in USA)
- TLC Operators are setting data transfer limits.

3. Leverage the Flash Video Ecosystem

What delivery technique to maximize QoS ?

Flash Player 11 supports several delivery techniques that can match any cost / security / portability / QoS requirements.

- **Progressive Download**
- **RTMP(E) Streaming**
- **RTMP Dynamic Streaming**
- **HTTP Dynamic Streaming**
- **Peer Assisted Streaming**
- **HLS for iOS devices (new FMS4.5 feature)**

3. Leverage the Flash Video Ecosystem
is dynamic streaming the solution ?

Dynamic Streaming can help in maximizing **Quality of Service** allowing the player to switch from a bitrate to another depending by the network conditions.

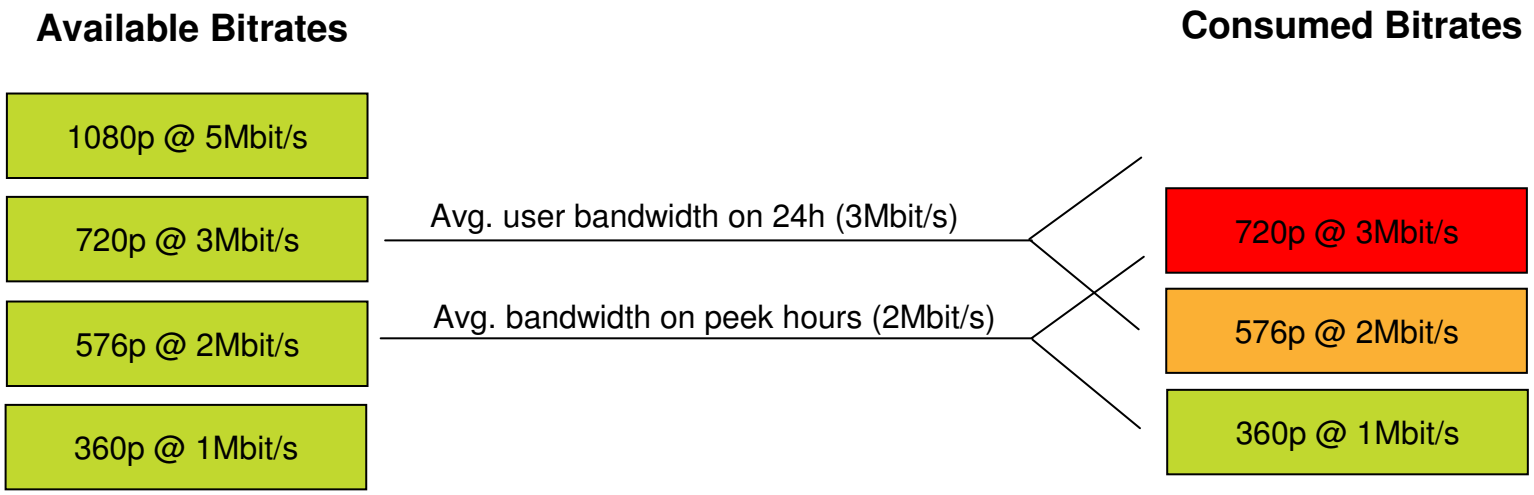
But it cannot be used as an “alibi” for un-optimized encodings.

The temptation is to say:

“Now that I have Dynamic Streaming I can simply encode enough different bitrates to serve from HD to low bitrates for mobile!”

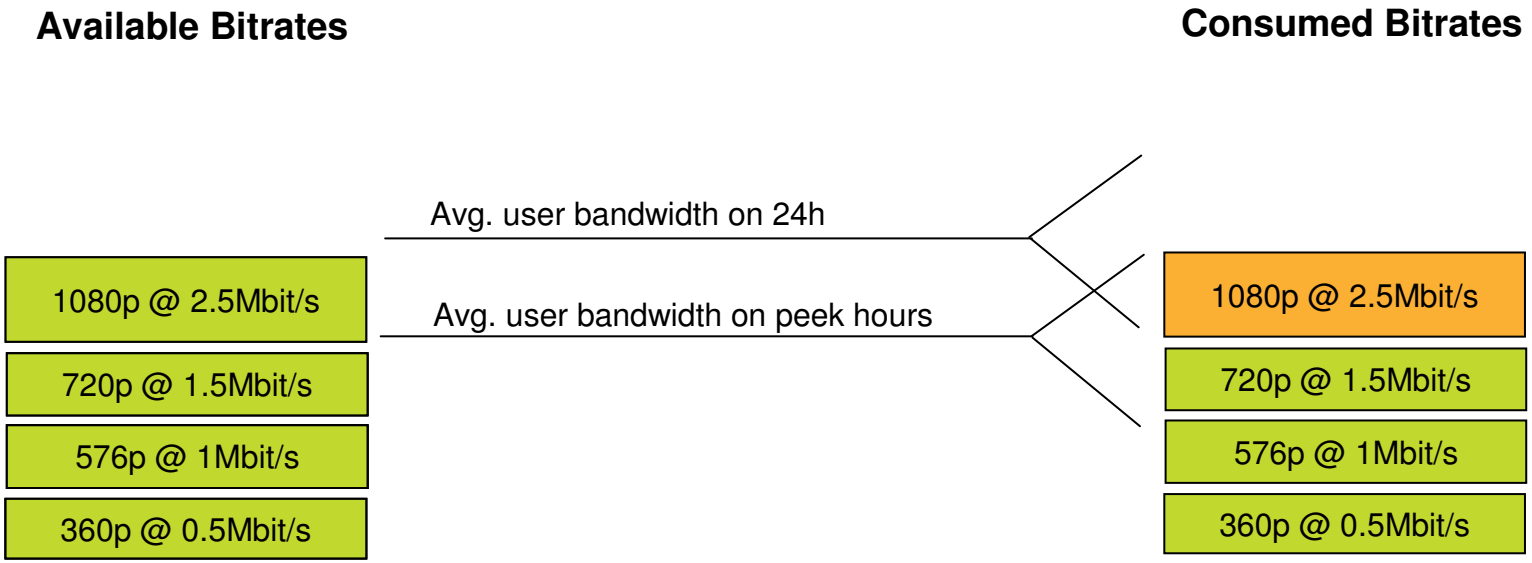
Flash Video Ecosystem

(H)DS with un-optimized encoding



In this scenario only a limited subset of users are able to watch 1080p and 720p and during peak hours even to stream the 576p or (480p) may be difficult.

(H)DS with un-optimized encoding



In this scenario the most users are able to stream 720p even during peek hours and have good chances to stream 1080p.

“Success is in the Details”

Don’t limit to a good setup, instead **monitor performance** continuously to know even more about the *boundary conditions* and the *UX* and how they are changing over time. **Leverage Flash Player QoS API** to collect UX informations.

Encoding best practices: Encoding Strategy

Define a *flexible* **Encoding Strategy** to optimize each rendition and the whole DS set:

- **Encoding best-practices**
  - **Optimize the encoding process**
  - **Choose a balanced set of resolutions and bitrates**
  - **Eventually use an adaptive encoding workflow**
- **Delivery best-practices**
  - **Use dynamic streaming (Strobe - OSMF - Custom)**
  - **Optimize UI for each device's category**
- **Be creative to overcome Flash Video Ecosystem's limits**

Flash Video Ecosystem

Build an encoding workflow

Adobe Media Encoders

Open Source

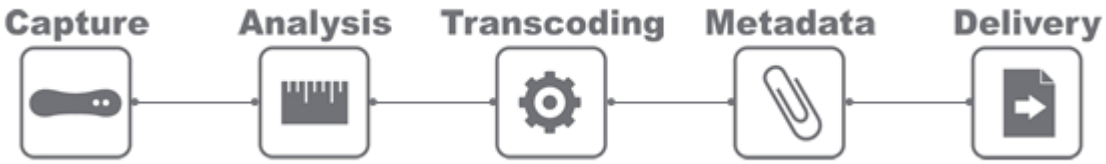
FFMPEG

Partners

dicas · grab networks · multicast · digital rapids · INLET technologies · sorenson media · iStreamPlanet · KULAbyte · envivio · media excel · ateme · RHOZET Universal Media Transcoding · ripcode · On2 technologies · TELESTREAM · ViewCast corporation

Optimize the encoding process

To optimize a **linear encoding workflow** for quality :



1. Use pristine sources
2. Find a balance between parameters of H.264 and encoding time
3. Understand the constraints of each class of devices (profile and level)
4. Pre-process the video source

Encoding best practices: H.264

H.264 contains several new features that allow it to compress video much more effectively than older H.26x standards or other modern codecs like VC-1 and VP8.

Maximize the parameters

↓

Maximize the Quality/Bitrate ratio

↓

Maximize the Encoding Time

Depending by your business and technical constraints, you many need to find your balance between encoding time and encoding efficiency.

Key Parameters

Most important parameters in H.264 encoding:

- **Frame resolution & Bitrate**
- **Bitrate allocation:** CBR, VBR or ABR
- **Multi-pass encoding:** 1 pass or multi pass
- **FrameRate:** usually same as source
- **IDR Interval or GOP size:** variable up to 10s for progressive, fixed 2-4s for DS
- **Profile and Level:** depend by target device

- **Number of B-Frames*:** 0-16 (recommended 3-5)
- **Number of Reference Frames*:** 1-16 (recommended 3-5)
- **Motion estimation and compensation parameters * & ****
- **RDO (Rate Distortion Optimization)****
- **PSY Optimizations****

* = may depend by “profile” and “level”

** = may be set by pre-defined encoding accuracy preset

Key Parameters: Profile & Level

- The H.264 decoder (software) implemented in Flash Player is very good. Supports **baseline**, **main, high, high10** profiles and level up to **5.1.** In the past you might simply target *high profile* and *4.1*…

- But with the rapid spread of Flash on multiple screens (Desktop, TV, Tablet, Mobile), you need to define a **dedicated encoding and delivery strategy** for each scenario especially to leverage HW acceleration on mobile:

Encoding for Desktops :

High profile, level 4.1

Higher resolutions (up to FullHD)

Higher bitrates (megabits)

Encoding for Mobile :

Baseline profile, level 3.1

Lower resolutions (sub HD)

Lower bitrates (hundreds of Kbps)

Pre-filtering

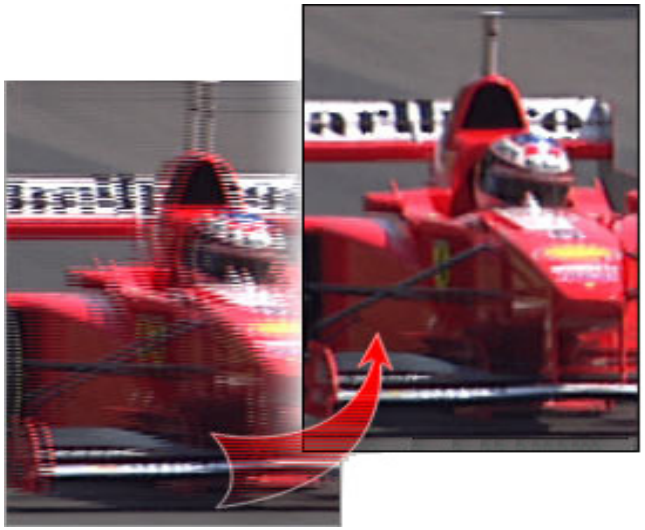
Denoising

Source frame with noise

Frame processed by MSU Denoising Filter

Resizing

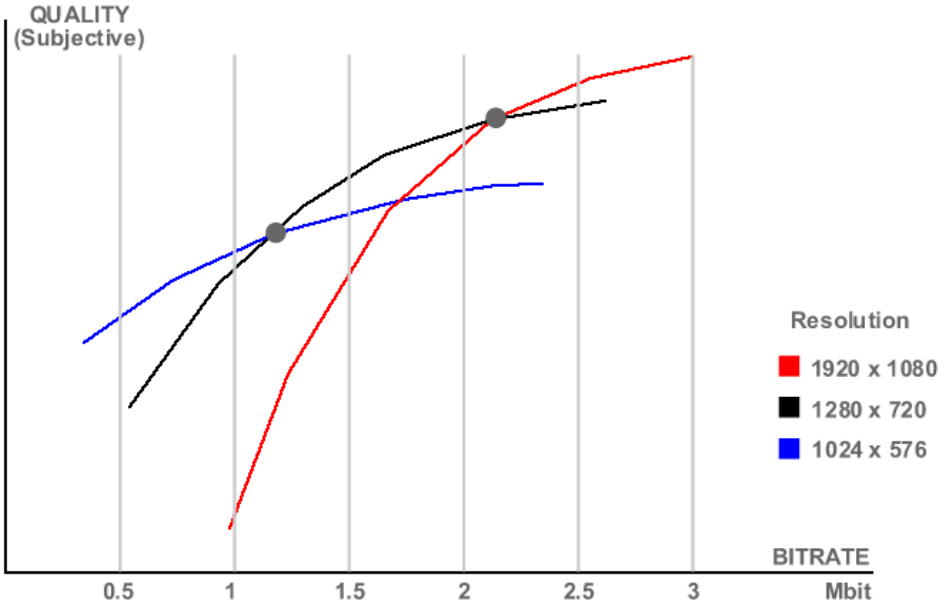
Deinterlacing



Resolution-Bitrate Mix

Choose the best resolution-bitrate mix

The two most important parameters in encoding are **frame resolution** and **bitrate**. The importance of a good balance between resolution and bitrate is often underestimated. Understand **rate distortion curves.**



Rate distortion curves depend by source complexity and resolution so you have a problem with up to 4 degrees of freedom:

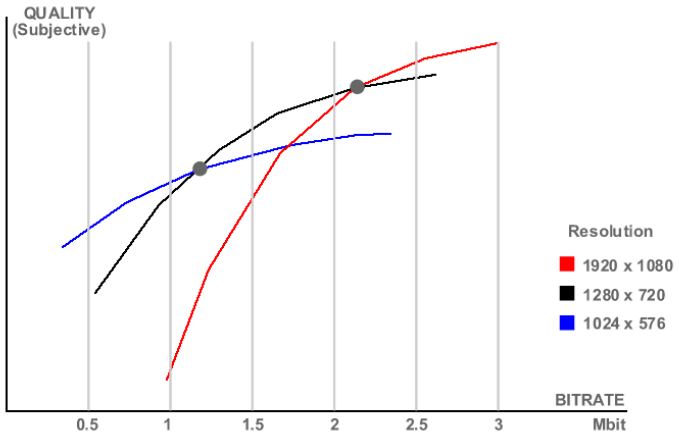
- **Frame resolution**
- **Average bitrate**
- **Video complexity**
- **Desired level of quality**

Resolution-Bitrate Mix

Subjective or Objective quality control (Worst Case)

Perform **tests of subjective** (focus test) and/or **objective quality** (PSNR o SSIM metrics) on your workflow to find an optimal set of resolution and bitrates for **Desktop+TV** and **Mobile+Tablet** using worst cases.

Note: Perceived quality may also depend by DPI of the display



4:3	16:9
320x180	320x240
480x360	480x272
512x384	512x288
576x432	640x360
640x480	1024x576
704x528	1280x720
768x576	1920x1080

Note: keep dimensions divisable by 16

Resolution-Bitrate Mix

Choose the best resolution-bitrate mix

Scenario: **Encode for the Desktop**

Big Screen, 720p High@4.1L content easily decoded by old HW, broadband

Aspect ratio 16:9	
Resolution	Bitrate
1920 x 1080 *	2-**4** Mbit/s
1280 x 720	1-**2** Mbit/s
1024 x 576	0.8-**1.5** Mbit/s
848 x 480	0.6-**1** Mbit/s
640 x 360	0.4-**0.7** Mbit/s

Aspect ratio 4:3	
Resolution	Bitrate
720 x 576	0.6-**1** Mbit/s
640 x 480	0.5-**0.8** Mbit/s
480 x 360	0.3-**0.6** Mbit/s
320 x 240	0.2-**0.4** Mbit/s

Resolution-Bitrate Mix

Choose the best resolution-bitrate mix

Scenario: **Encoding for Mobile**

Small Screen, heterogeneous support for HW acceleration, narrowband.
I suggest to use Baseline@3.1 for maximum support to HW acceleration.

Aspect ratio 16:9	
Resolution	Bitrate
848 x 480 *	0.7-**1.2** Mbit/s
640 x 360	0.5-**0.8** Mbit/s
512 x 288	0.4-**0.6** Mbit/s
480 x 272	0.3-**0.4** Mbit/s
320 x 180	0.2-**0.3** Mbit/s

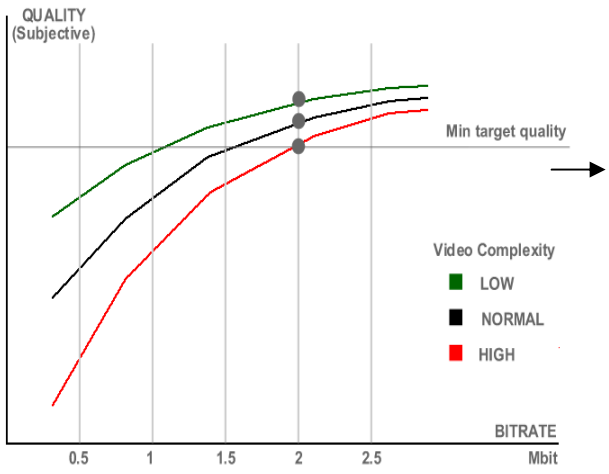
Aspect ratio 4:3	
Resolution	Bitrate
640 x 480	0.6-**1** Mbit/s
512 x 384	0.5-**0.8** Mbit/s
480 x 360	0.3-**0.6** Mbit/s
320 x 240	0.2-**0.4** Mbit/s

Note: in mobile scenario, at low bitrates it's possible to halve FPS

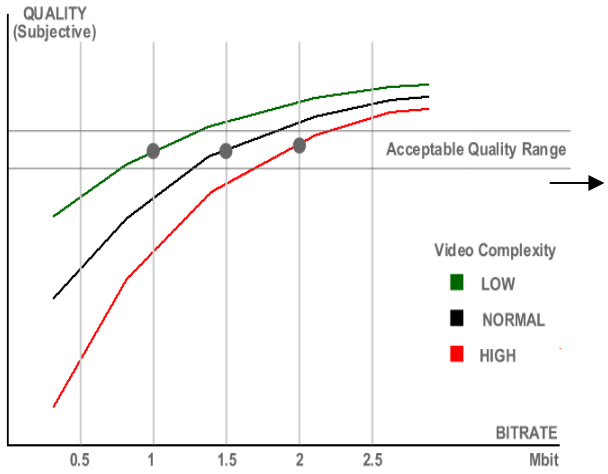
Resolution-Bitrate Mix

Adaptive encoding workflows - case 1

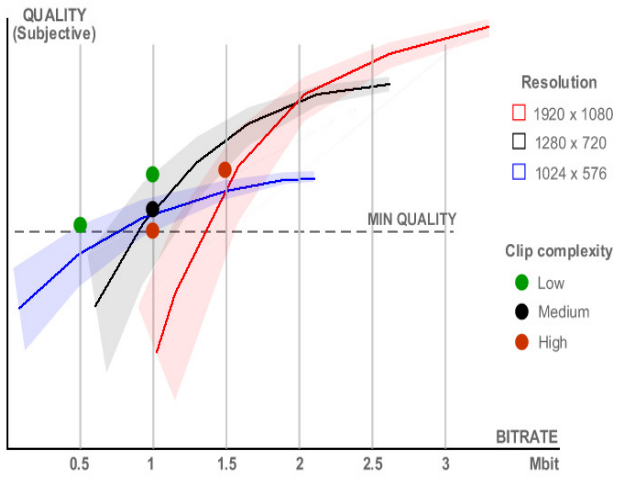
Use a content adaptive encoding to define encoding parameters on the fly depending by content complexity. Useful to optimized overall bandwidth usage.



Static strategy: fixed bitrate for every complexity



Simple adaptive: bitrate changes according to complexity

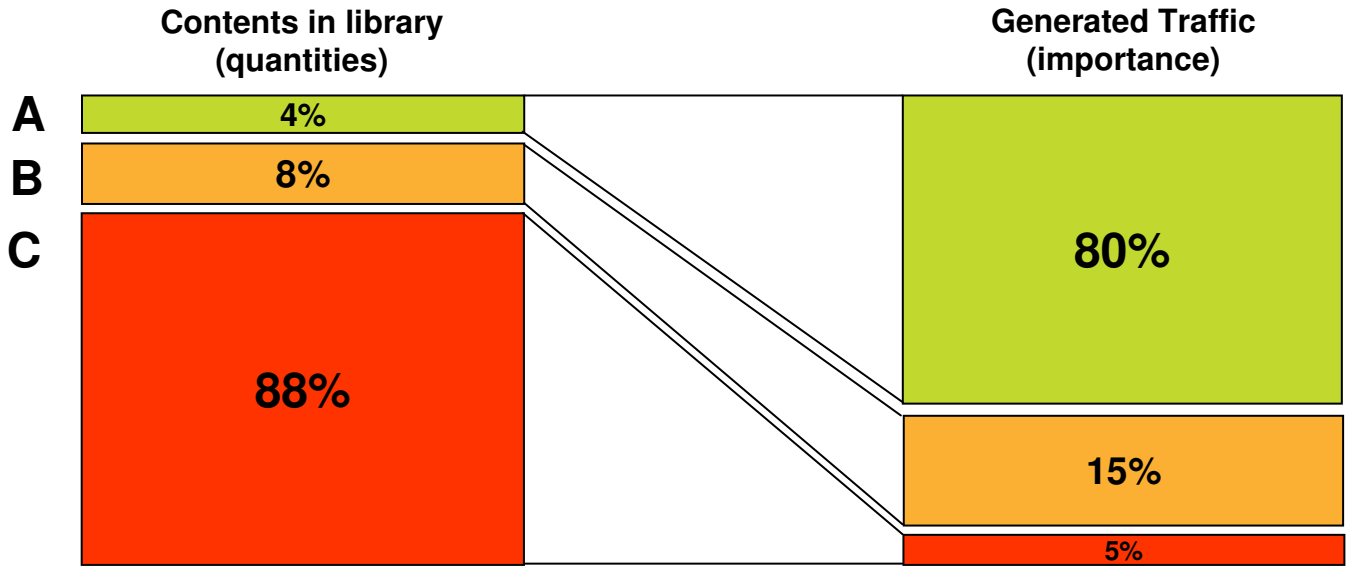


Complex adaptive: bitrate, resolution, parameters and filters change

Adaptive encoding workflows - case 2

Use statistic laws like the famous **“Pareto principle”** (or **80-20** rule or **ABC**) to encode more accurately only relevant contents.

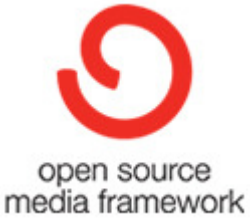
Useful to optimize limited encoding resources.



Playback recommendations

Which PLAYER ?

- Completely custom (low level Flash Player API)
- OSMF (Framework)
- Strobe or Flash Media Player (turn-key solutions)



What Objectives ?

- Maximize **UX** and Quality of Service (**QoS**) in every scenario and on every device

Playback recommendations

User Experience is King

Remember that the success is in the details. Optimize UX for desktop, mobile and TV separately, each have quite different interaction model.

- Detect platform and offer a dedicated UI **Mandatory!**
- Use Tablet&Mobile as advanced remote for video apps on TV

Dynamic Streaming

Maximize QoS: Use (RTMP/HTTP) Dynamic Streaming + HLS (iOS)

- Use fixed length GOP (2-4 seconds) to easily align keyframes
- Use 2-pass CBR (or “light” VBR es: VBV buffer = gop length)
- Use same audio settings for each stream (HE-AAC v2)
- Find a balanced set of resolutions - bitrates

Desktop 16:9

Resolution	Bitrate
1280 x 720	1-**2** Mbit/s
1024 x 576	0.8-**1.5** Mbit/s
848 x 480	0.6-**1** Mbit/s
640 x 360	0.4-**0.7** Mbit/s

High@4.1

Baseline@3.1

Mobile 16:9

Resolution	Bitrate
640 x 360	0.5-**0.8** Mbit/s
512 x 288	0.4-**0.6** Mbit/s
480 x 272	0.3-**0.4** Mbit/s
320 x 180	0.2-**0.3** Mbit/s

→

Multiscreen 16:9

Resolution	Bitrate
1280 x 720	1-**2** Mbit/s
1024 x 576	0.8-**1.5** Mbit/s
848 x 480	0.6-**1** Mbit/s
640 x 360	0.5-**0.8** Mbit/s
512 x 288	0.4-**0.6** Mbit/s
480 x 272	0.3-**0.4** Mbit/s
320 x 180	0.2-**0.3** Mbit/s

FFmpeg – Swiss Army Knife of Internet Streaming

- Open source universal transcoder (CLI or Lib)
- One of the “pillars” of Internet streaming used by ***YouTube***, ***Hulu***, ***Vimeo***, ***Facebook*** and many others.
- Low level tool useful for server side batch encoding
- Almost 70% of all consumed content is encoded with FFmpeg (mainly because of Hulu and YouTube)

Upsides:

- Very good H.264 encoder (x264 lib)
- Plenty of parameters, flexibility
- Supports RTMP protocol
- Free

Downsides:

- Reliability, compatibility, integration costs
- Licensing issues

FFmpeg – Swiss Army Knife of Internet Streaming

Remember: Be ***creative***, ***adaptive***, ***dynamic*** to overcome limitations …

FFmpeg can be used inside **adaptive encoding pipelines** as **glue logic**

FFmpeg can:

- Encode single file very fast to very accurate.
- Encode multi-bitrate sets.
- Extract thumbs, Audio or Video tracks, slices, remux contents.
- Encode live from a local source.
- Trascode live from/to RTMP sources/destinations.
- Repurpose existing streams to different classes of devices.
- Compress images using H.264

Very useful to enhance the new FMS 4.5 capability to stream to iOS device

FFmpeg – Swiss Army Knife of Internet Streaming

Example of H.264 encoding:

Encode expliciting a wide set of parameters:

ffmpeg -i INPUT ***-r 25 -b 1000k –s 640×360*** *-vcodec libx264 -flags +loop -me_method hex* ***-g 250*** *-qcomp 0.6 -qmin 10 -qmax 51 -qdiff 4* ***-bf 3*** *-b_strategy 1 -i_qfactor 0.71 -cmp +chroma -subq 8 -me_range 16 -coder 1 -sc_threshold 40 -flags2 +bpyramid+wpred+mixed_refs+dct8x8+fastpskip -keyint_min 25* ***-refs 3*** *-trellis 1* ***-level 30*** *-directpred 1 -partitions +parti8x8+parti4x4+partp8x8+partb8x8 -threads 0 -acodec libfaac -ar 44100 -ab 96k -y OUTPUT.mp4*

(Did I mention that FFmpeg offers really plenty of parameters ?)

Encode using profiles:

ffmpeg -i INPUT -an -vcodec libx264 ***-vpre fast*** *-b 1000k -s 640×360 OUTPUT.mp4*

FFmpeg – Swiss Army Knife of Internet Streaming

Encode for multiple-devices:

Encode for Desktop/TV: **High@4.1, Slow 2-pass encode = high quality, VBR**

FFmpeg –i INPUT –pass 1 –an –vcodec libx264 –b 1500k –s 1280x720 –vpre slower_fastfirstpass -level 41 OUTPUT.mp4

FFmpeg –i INPUT –pass 2 –acodec libfaac –ab 128k –ar 44100 –b 1500k –s 1280x720 –vpre slower –level 41 OUTPUT.mp4

Encode for Mobile: **Baseline@3.0, Slow 2-pass encode = high quality, VBR**

FFmpeg –i INPUT –pass 1 –an –vcodec libx264 –b 500k –s 640x360 –vpre slower_fastfirstpass ***–vpre baseline –level 30*** *OUTPUT.mp4*

FFmpeg –i INPUT –pass 2 –acodec libfaac –ab 64k –ar 44100 –b 500k –s 640x360 –vpre slower ***–vpre baseline –level 30*** *OUTPUT.mp4*

FFmpeg – Swiss Army Knife of Internet Streaming

Encode for multi-bitrate

Requesites: Aligned keyframes and CBR
Trick: force FFmpeg to align keyframes re-using a common first pass.

FFmpeg –i IN ***–pass 1*** *–an –vcodec libx264 –r 30 –b 1500k* ***–bufsize 1500k*** ***–keyint_min 60 –g 120*** *–s 1280x720 –vpre slower_fastfirstpass O_1500.mp4*

A *FFmpeg –i IN –pass 2 –an –vcodec libx264 –r 30 –b 1500k* ***–bufsize 1500k*** **–keyint_min 60 –g 120** *–s 1280x720 –vpre slower O_1500.mp4*

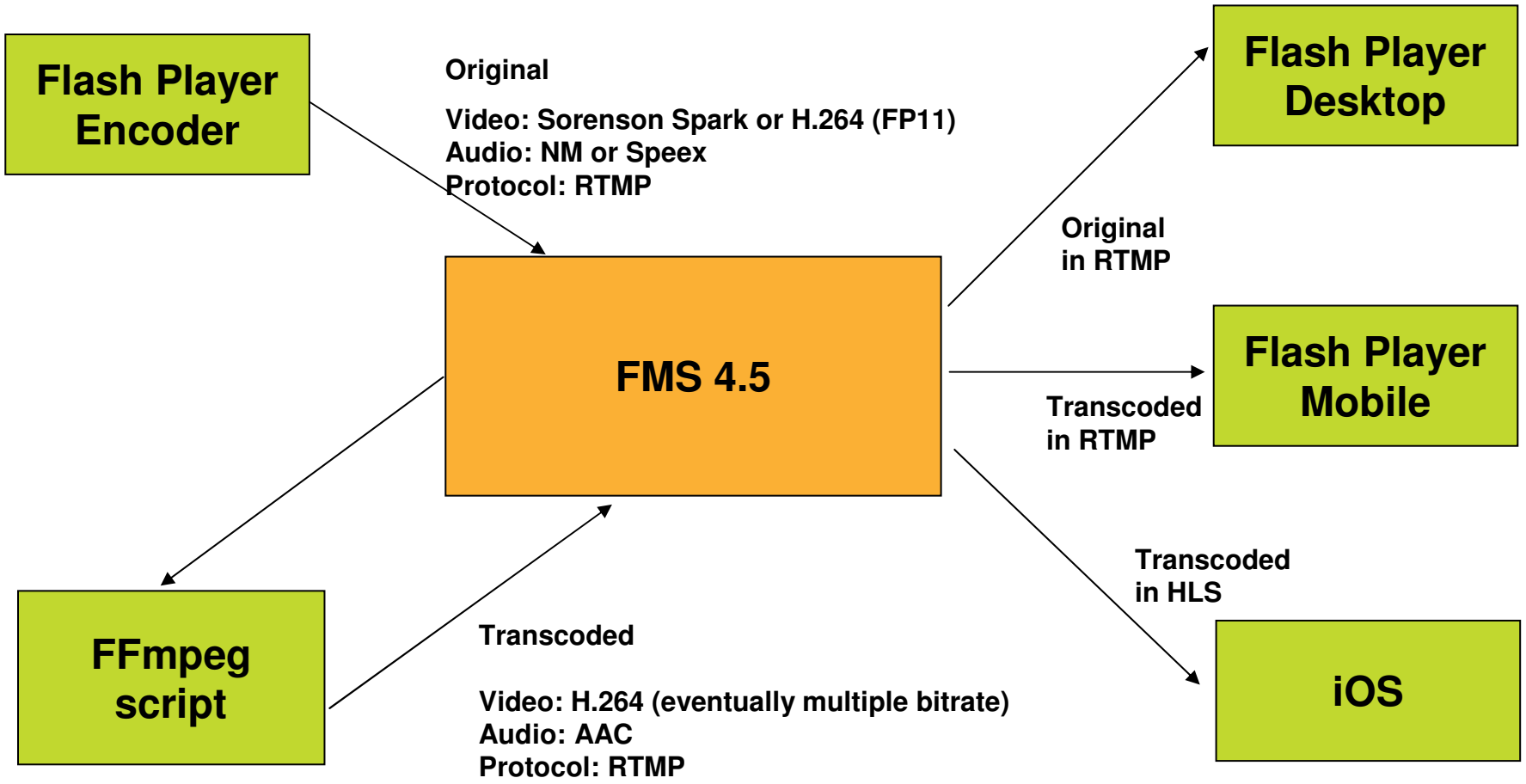
B *FFmpeg –i IN –pass 2 –an –vcodec libx264 –r 30 –b 1000k* ***–bufsize 1000k*** **–keyint_min 60 –g 120** *–s 854x480 –vpre slower O_1000.mp4*

C *FFmpeg –i IN –pass 2 –an –vcodec libx264 –r 30 –b 500k* ***–bufsize 500k*** **–keyint_min 60 –g 120** *–s 640x360 –vpre slower O_500.mp4*

FFmpeg – Swiss Army Knife of Internet Streaming

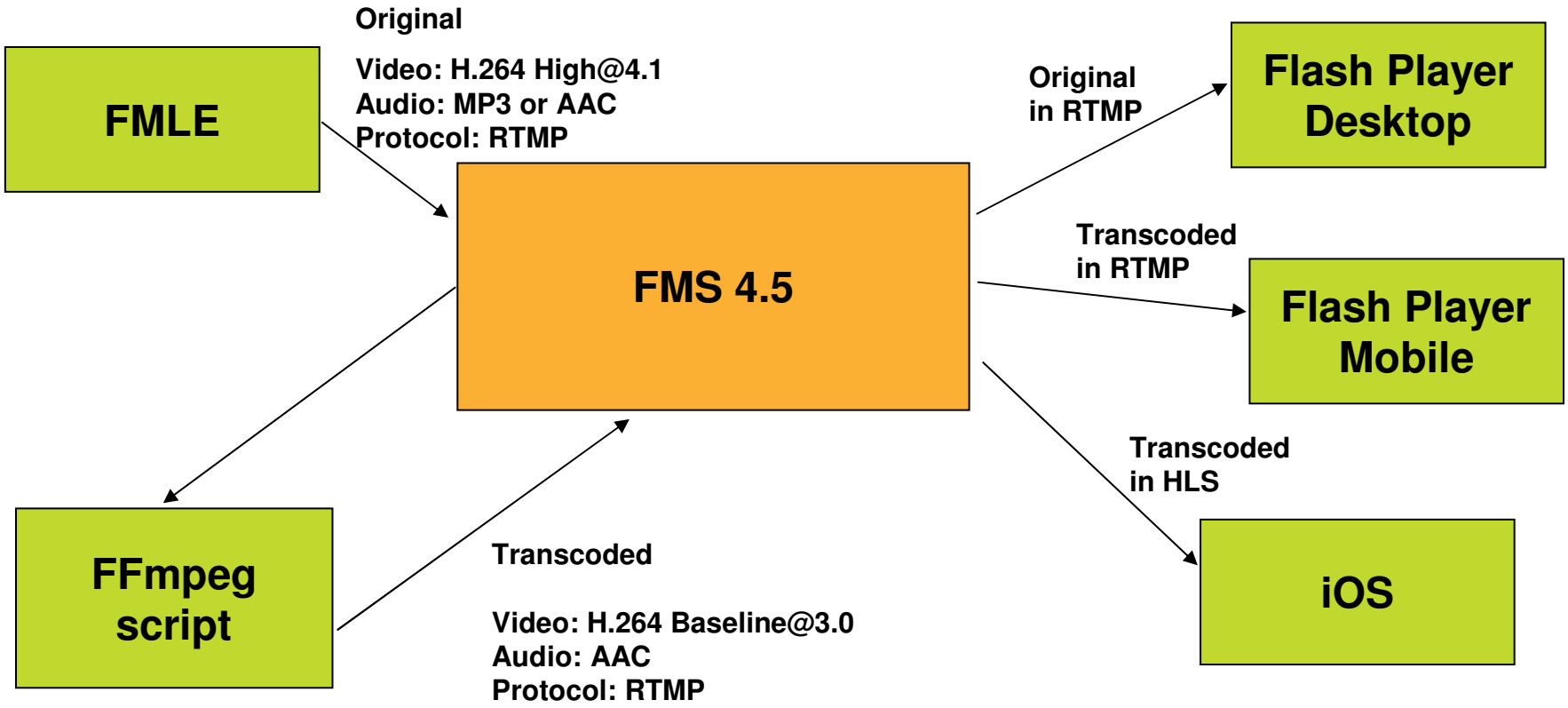
Transcoding live from RTMP to RTMP – Case 1

ffmpeg –re -i rtmp://server/live/originalStream -acodec libfaac -ar 44100 -ab 48k -vcodec libx264 -vpre slow -vpre baseline -f flv rtmp://server/live/h264Stream



Transcoding live from RTMP to RTMP – Case 2

ffmpeg -re -i rtmp://server/live/high_FMLE_stream -acodec copy -vcodec x264lib -s 640×360 -b 500k -vpre medium -vpre baseline rtmp://server/live/baseline_500k -acodec copy -vcodec x264lib -s 480×272 -b 300k -vpre medium -vpre baseline rtmp://server/live/baseline_300k -acodec copy -vcodec x264lib -s 320×200 -b 150k -vpre medium -vpre baseline rtmp://server/live/baseline_150k -acodec libfaac -vn -ab 48k rtmp://server/live/audio_only_AAC_48k



Turn in your surveys for a chance to WIN!

Adobe Press

- Hand in your surveys to the room monitors
- One survey per session will be selected as a winner of an Adobe Press e-book or Video

  Introduction to Adobe Edge

  Web Design with Muse (code name) from Adobe

  Android App Development and Design: Learn by Video

- Winners will be notified via e-mail at the end of each day