

Tools that enable streaming

Anil Kokaram

Reference Material

Background on Digital Media

<http://xiph.org/video/vid1.shtml> <http://xiph.org/video/vid2.shtml>

Excellent Overview of Media Compression at

http://people.xiph.org/~tterribe/pubs/lca2012/auckland/intro_to_video1.pdf ;

<https://www.xiph.org/daala/>

HEVC Information <http://hevc.hhi.fraunhofer.de/>

<http://www.atlanta-smppte.org/HEVC-Tutorial.pdf>

H.264 Information <http://www.itu.int/rec/T-REC-H.264>

Tools : www.ffmpeg.org <http://www.videolan.org/>

VP9 Presentation at Google IO 2013 : <http://www.youtube.com/watch?v=K6JshvbllcM>

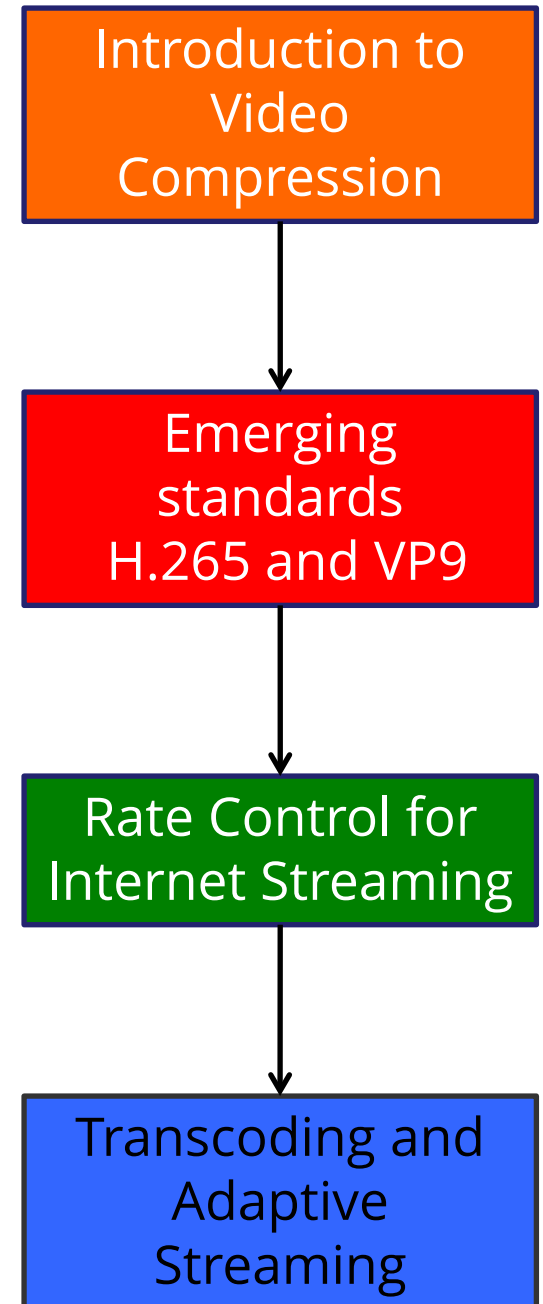
Wikipedia actually quite good on this stuff

Rate Control in H.264 : http://www.pixeltools.com/rate_control_paper.html

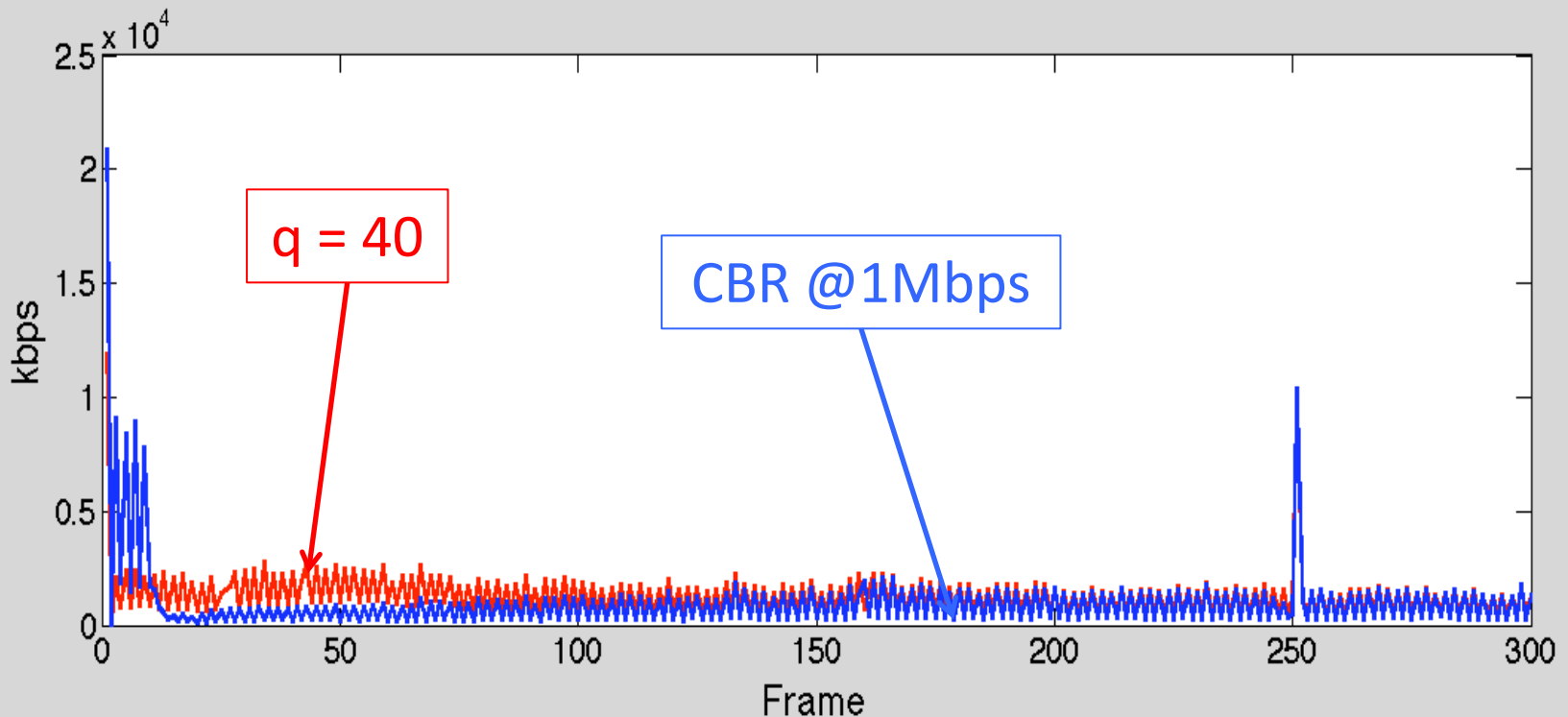
Content

Lab report on rate distortion
and using ffmpeg

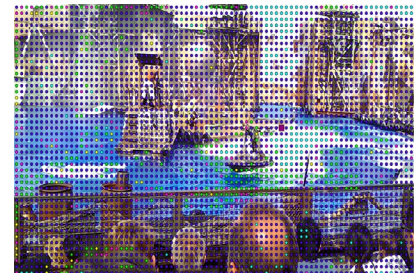
Then final assignment on making
Representations for Adaptive Streaming



H.264 Constant Q Versus CBR



You can manually choose the “right” q to get about 1Mbps. But the CBR algorithm works out what it should be adaptively.



Enabling Video over the internet

<https://developer.apple.com/streaming/>

Streaming Protocols in the Browser/Device : DASH, HLS

Transcoding : Bit rate adaptation for devices

Network Transport TCP/IP RTMP UDP CDN's

Software Encryption/Common Encryption/Digital Rights Management DRM

Enabling Video over the internet

Different devices have different capabilities [speed, hardware, storage, display resolution]

Different networks have different capacities [bandwidth] at different times

People want more than one application ON at the same time

How can we convert between different bitrates and different formats?

Transcoding/Transrating

How do we adapt bitrates to changing network conditions?

Adaptive Streaming/DASH

Why do movie studios trust video streaming?

Digital Rights Management (DRM), Common Encryption Formats (Widevine)

How do we _actually_ stream video over the internet?

Network Protocols : RTMP, UDP, TCP/IP

Transcoding

To change the bitrate of an incoming stream into a distributable stream

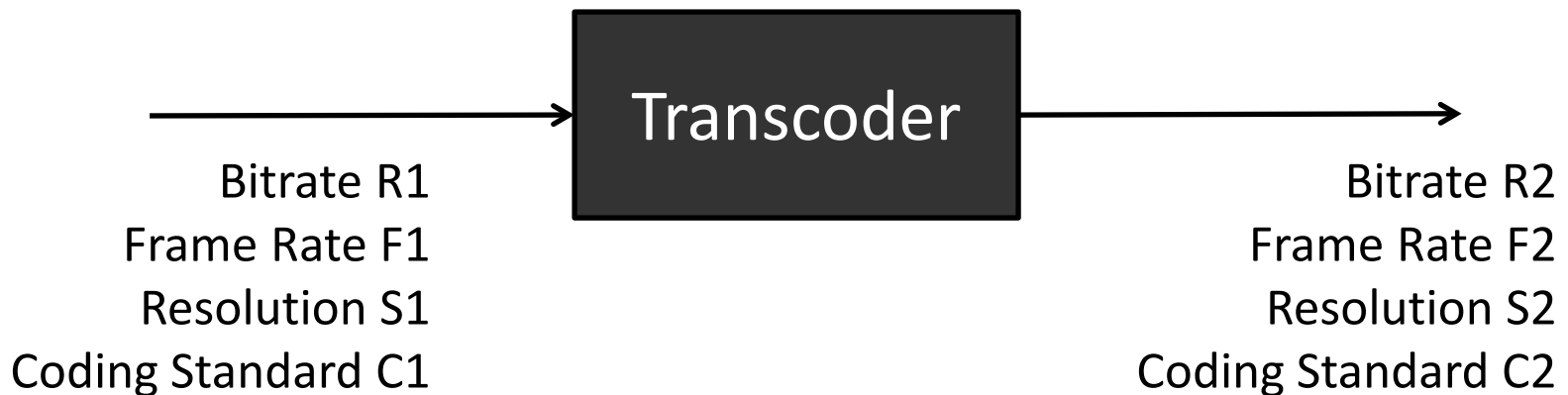
Netflix/YouTube/GooglePlay Ingest

Live streaming on ESPN, Olympics, NBC etc

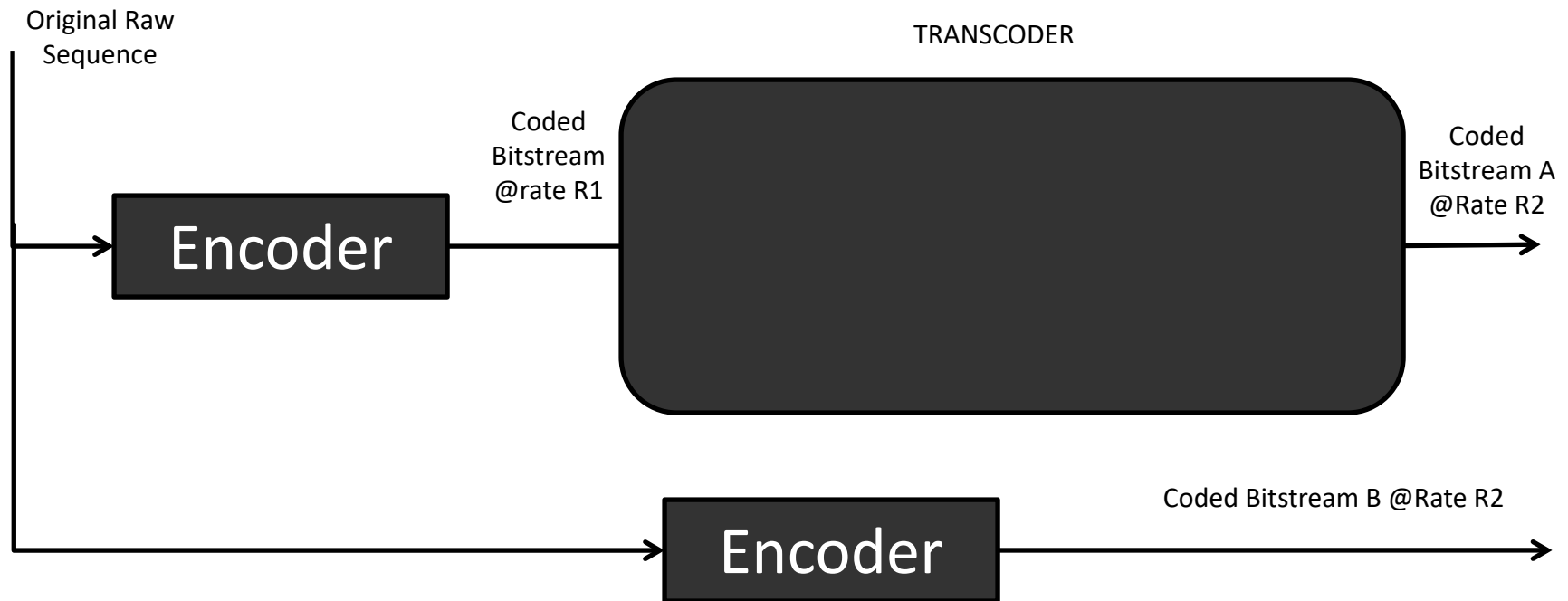
To change the format of a stream so that other devices can use it

MPEG2 (DTV) -> H.264 (Mobile, Desktop)

Change framerate, resolution, bitrate, format in response to network/device changes

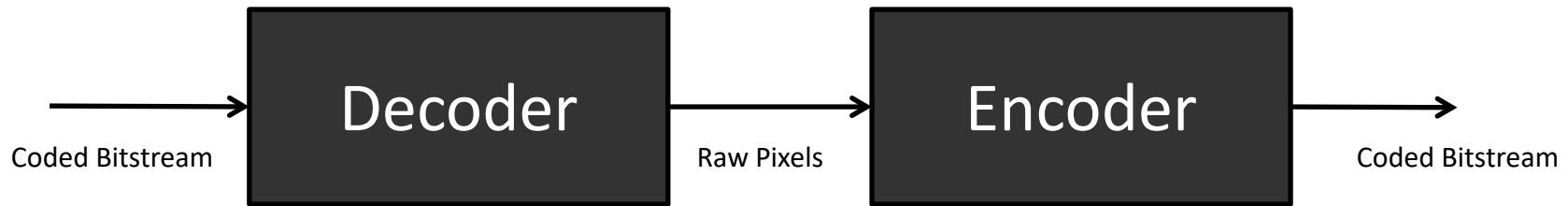


The ideal Transcoder



Creates the same bitstream as if the original raw data was available.
Bitstream A would be the same as Bitstream B.

Transcoding: Strategy 1



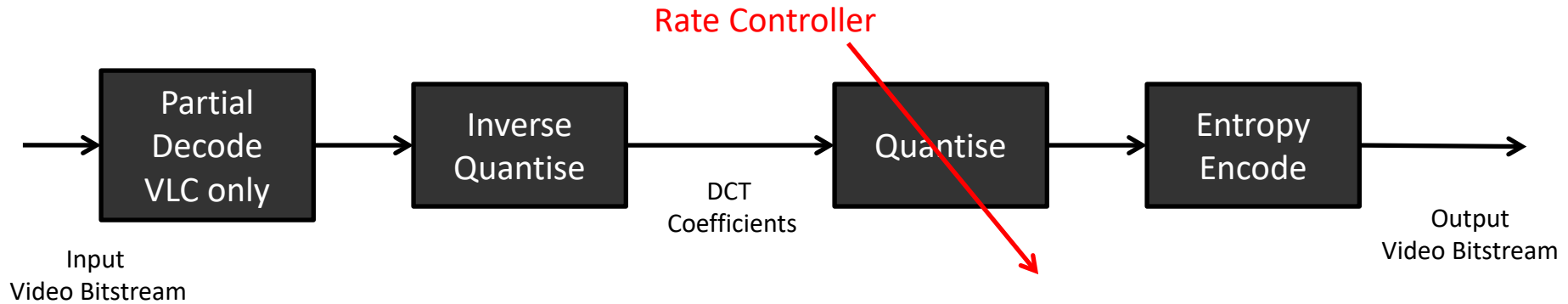
Cascade a Decoder and Encoder

Works well at high bitrates

At low input bitrates the “raw pixels” may exhibit so many artefacts that the Encoder Motion Estimator cannot predict frames well enough, so the output picture quality is worse than the input quality.

Also very cpu intensive : Encoder implies re-generating much information already available in the input bitstream e.g. motion and block mode decisions e.g. INTRA/INTRA-8x8/PREDICTED/SKIPPED etc etc

Transcoding: Strategy 2



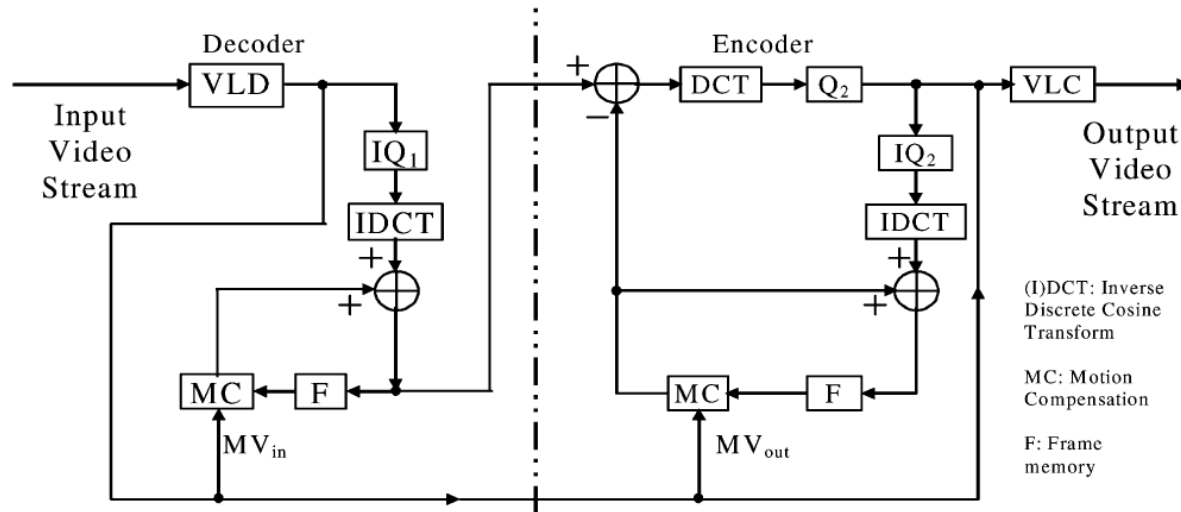
Open Loop “Requantisation”

Re-quantise the DCT coefficients by just deleting more of the hi-frequency coefficients until you hit the required rate.

Simple, very fast

Picture quality gets worse through the GOP till the next I frame : Why?

Transcoding: Strategy 3



Closed Loop Decode/Reuse/Encode : Closed Loop Transcoder

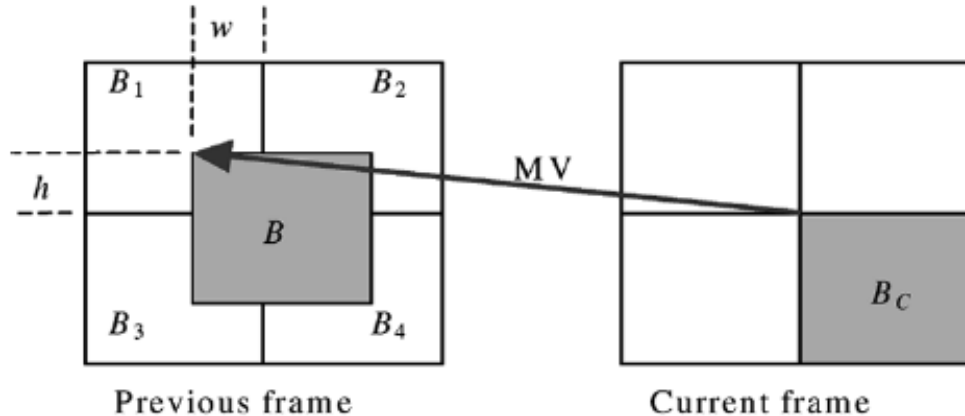
Decode to raw data

Re-use the motion information from the Partial Decoder

Calculate/update the motion compensation error in the partial encoder

Requantise the “true” prediction error in the light of the previous requantisation of coefficients.

Transcoding: Even faster



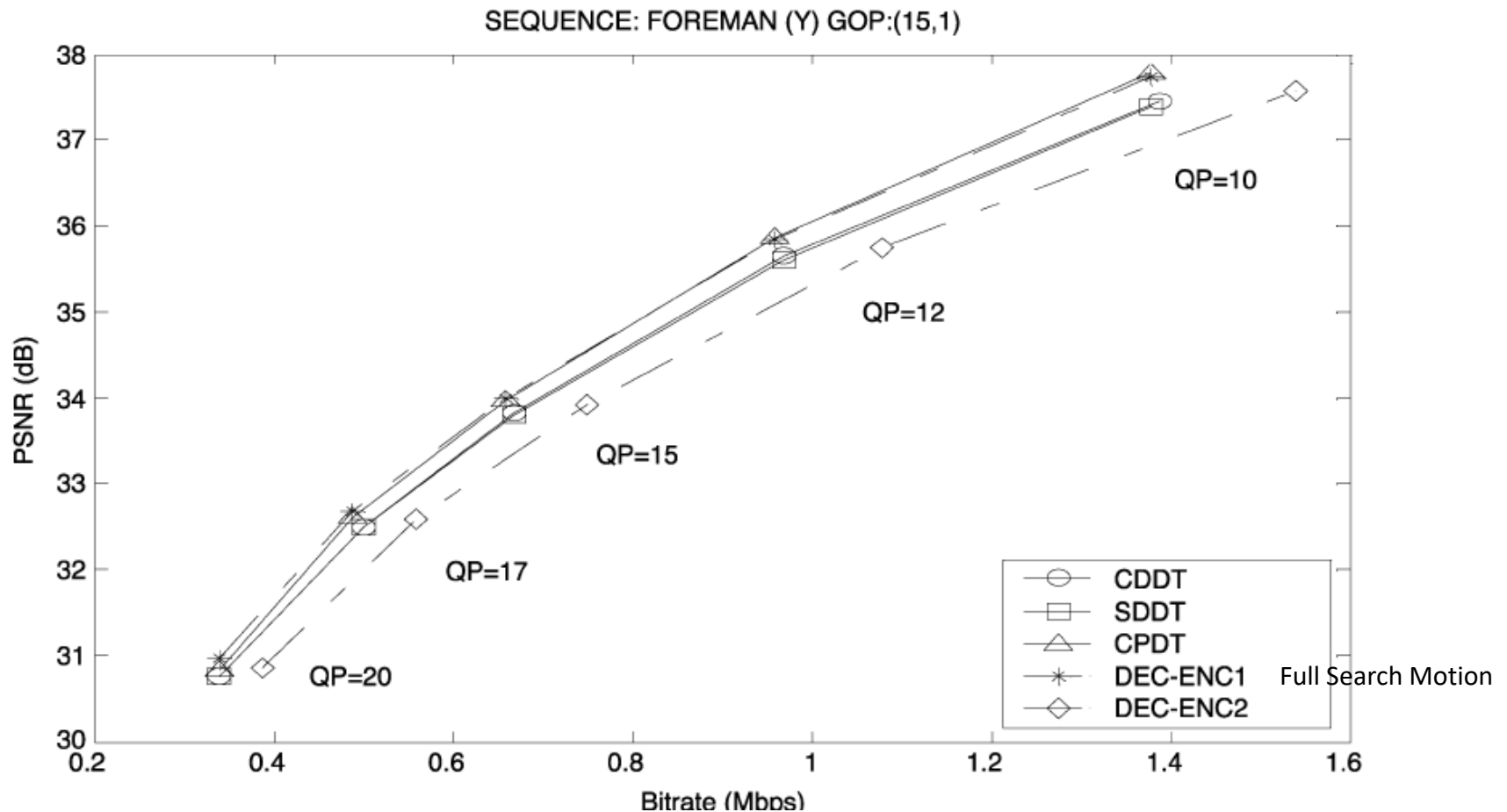
Try to do more in the DCT domain

Avoid decoding to Raw pixels!

Key DSP issues to solve

- : How to do motion compensation in the DCT domain?
- : How to extract DCT of blocks “between” existing ones
- : How to do linear interpolation in the DCT domain?

The surprise result



CPDT : Cascaded Pixel Domain Xcoder (Closed Loop Xcoder)
SDDT : Simplified DCT Domain Xcoder (Closed Loop Xcoder in DCT Domain)

What about format changes?

MPEG2 -> H.264

H.264 -> VP9 -> HEVC

Much more interesting.

Can re-use the motion information, but what about block patterns?

VP9 -> HEVC probably simpler than VP9 -> H.264 for that reason.

Block Modes also very different.

Alot of interesting work yet to be done here

More Information

Akamai

www.akamai.com

<http://www.beet.tv/2012/09/akamaicloud.html>

Elemental

<http://www.elementaltechnologies.com/>

Amazon Transcoding

<http://aws.amazon.com/elastictranscoder/>

FFMPEG Open Source Transcoding

www.ffmpeg.org

Review Papers

Digital Video Transcoding, Jun Xin et al, Proceedings of the IEEE, Vol 93, No 1, Jan 2005, pp 84-97

Video Transcoding Architectures and Techniques: An Overview, Anthony Vetro et al, IEEE Signal Processing Magazine, March 2003, pp 18-29

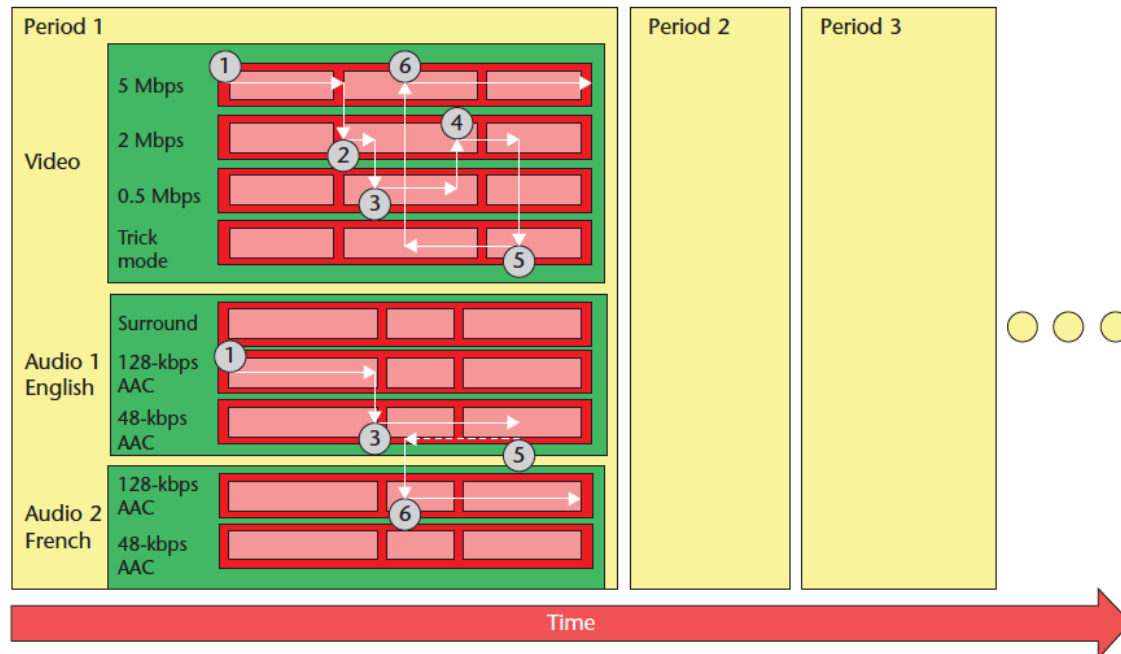
Video Transcoding: an overview of various techniques and research issues, Ishfaq Ahmad et al, IEEE Transactions on Multimedia , Vol 7 No 5, Oct 2005, pp 793-804

Adaptive Streaming

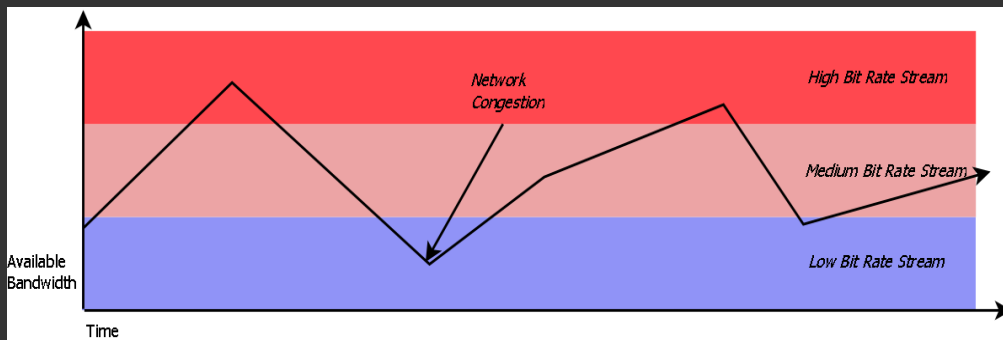
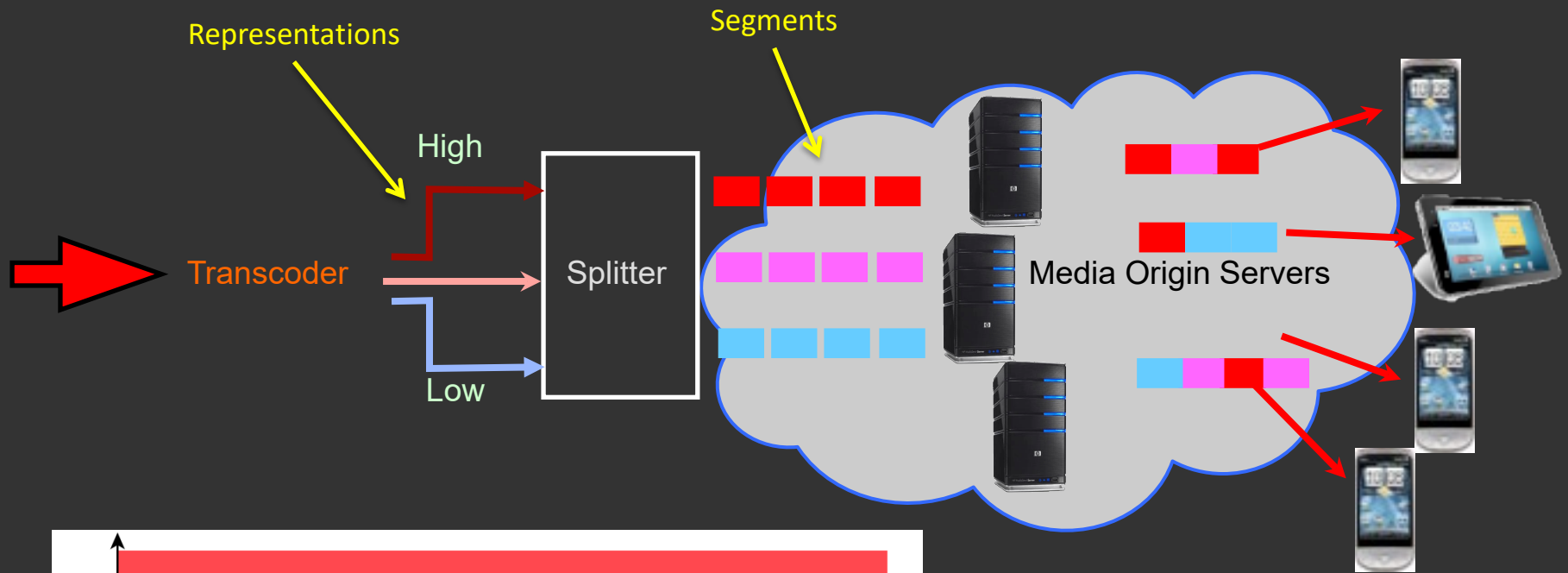
As old as online video streaming .. From about 1999

The MPEG-DASH Standard for Multimedia Streaming Over the Internet, Anthony Vetro, IEEE Multimedia Magazine, Oct-Dec 2011, pp 62-67

MPEG DASH = ISO/IEC 23009-1



DASH: Dynamic Adaptive Streaming over HTTP



Implemented as XML in the Client and Server

```
<Period>
  <!-- Video -->
  <AdaptationSet
    mimeType="video/mp4"
    codecs="avc1.4D401F"
    frameRate="30000/1001"
    segmentAlignment="true"
    startWithSAP="1">
    <BaseURL>video/</BaseURL>
    <SegmentTemplate timescale="90000" initialization="$Bandwidth%/init.mp4v"
media="$Bandwidth%/$Time$.mp4v">
      <SegmentTimeline>
        <S t="0" d="180180" r="432"/>
      </SegmentTimeline>
    </SegmentTemplate>
    <Representation id="v0" width="320" height="240" bandwidth="250000"/>
    <Representation id="v1" width="640" height="480" bandwidth="500000"/>
    <Representation id="v2" width="960" height="720" bandwidth="1000000"/>
  </AdaptationSet>
  <!-- English Audio -->
  <AdaptationSet mimeType="audio/mp4" codecs="mp4a.0x40" lang="en" segmentAlignment="0"
startWithSAP="1">
    <SegmentTemplate timescale="48000" initialization="audio/en/init.mp4a"
media="audio/en/$Time$.mp4a">
      <SegmentTimeline>
        <S t="0" d="96000" r="432"/>
      </SegmentTimeline>
    </SegmentTemplate>
    <Representation id="a0" bandwidth="64000" />
  </AdaptationSet>
  <!-- French Audio -->
  <AdaptationSet mimeType="audio/mp4" codecs="mp4a.0x40" lang="fr" segmentAlignment="0"
startWithSAP="1">
    <SegmentTemplate timescale="48000" initialization="audio/fr/init.mp4a"
media="audio/fr/$Time$.mp4a">
```

Special Sauce

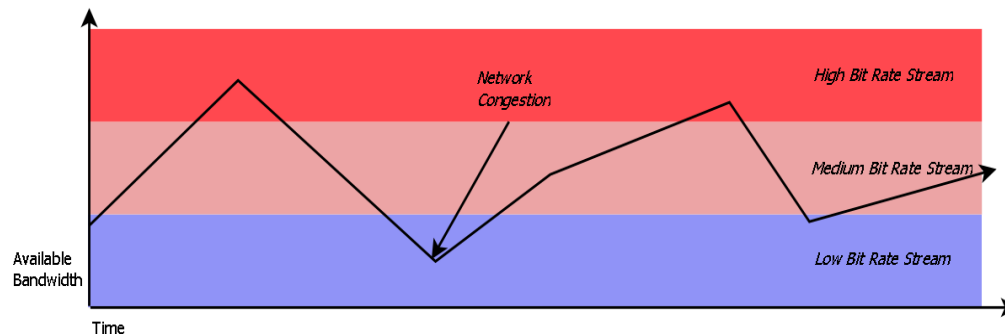
What is the best switching strategy?

How often should we switch? [Netflix and YT use of the order of 10secs]

How do we measure the bandwidth of the link? [WebRTC public source]

How do we know what bitrate each “representation” should be?

What is the metric we are trying to optimise?



Switching Strategies

No details in this lecture but two papers are an interesting read

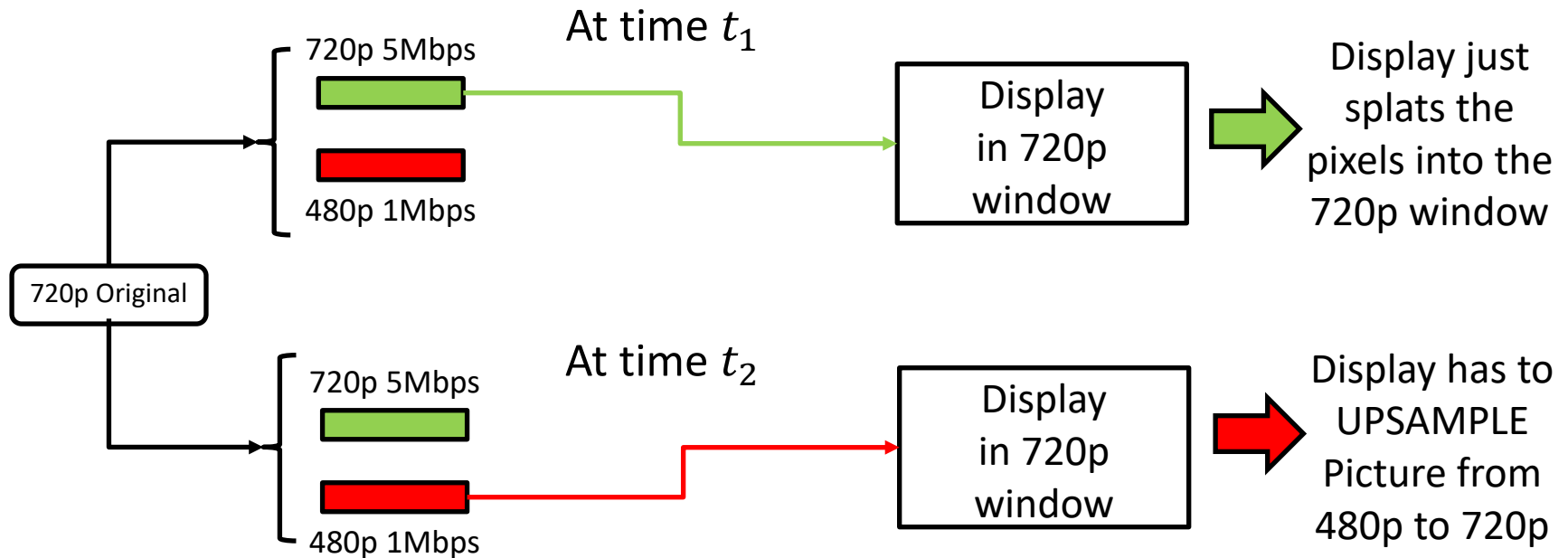
Rate Adaptation for Adaptive HTTP Streaming, Chenghao Liu, Imed Bouazizi, Moncef Gabbouj, ACM Conference on MMSys 2011, Feb 23-25, 2011, San Jose CA, pp169-174

Adaptation Algorithm for Adaptive Streaming over HTTP, Konstantin Miller et al, Proceedings of 2012 IEEE 19th International Packet Video Workshop, May 10-11, 2012, Munich, Germany, pp 173-178

Metrics : Mean Buffering Time, Speed of Adaptation

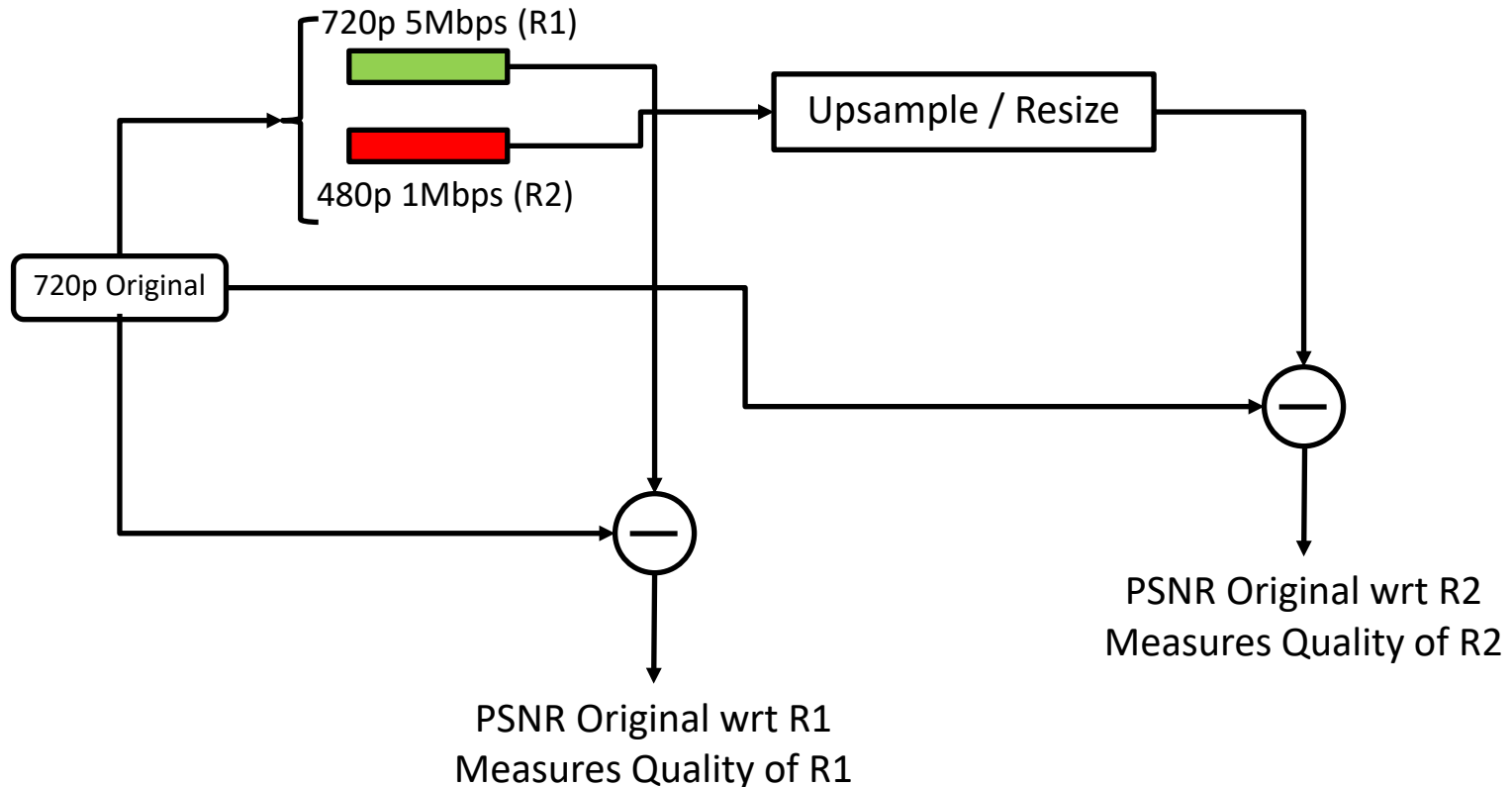
How do you measure the perceived quality?

The interaction between received resolution and display resolution



To compare quality of DASH representations we compare with the original resolution. That means upsampling before PSNR calculation.

Comparing representations

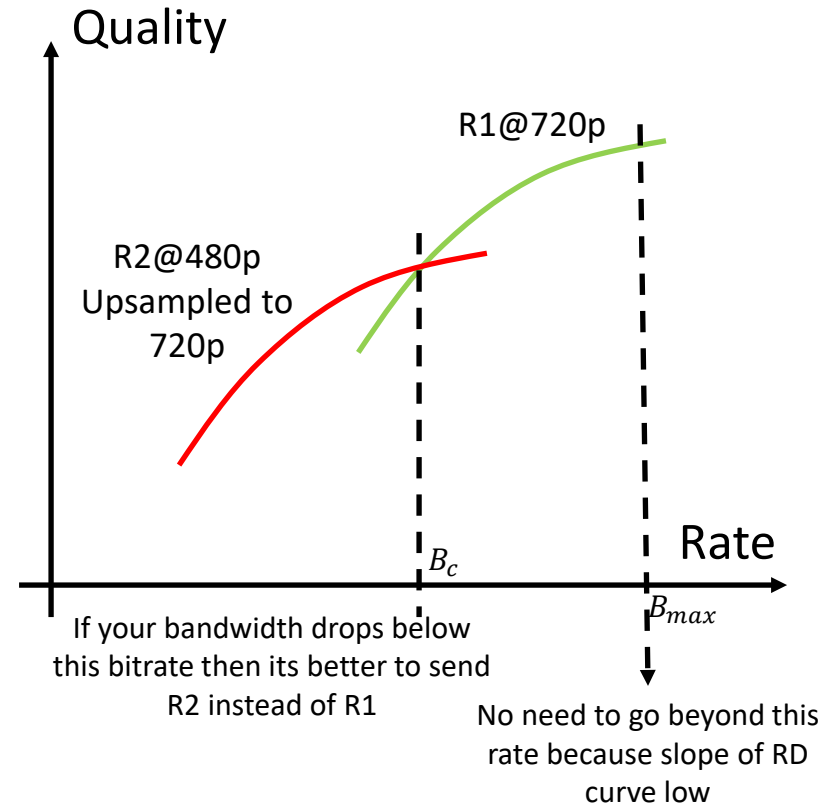


But how did we pick those bitrates of 5 Mbps and 1Mbps for those transcodes? Is it just that it “looked” good wrt measured quality?

How to pick target bitrates?

The bigdata approach

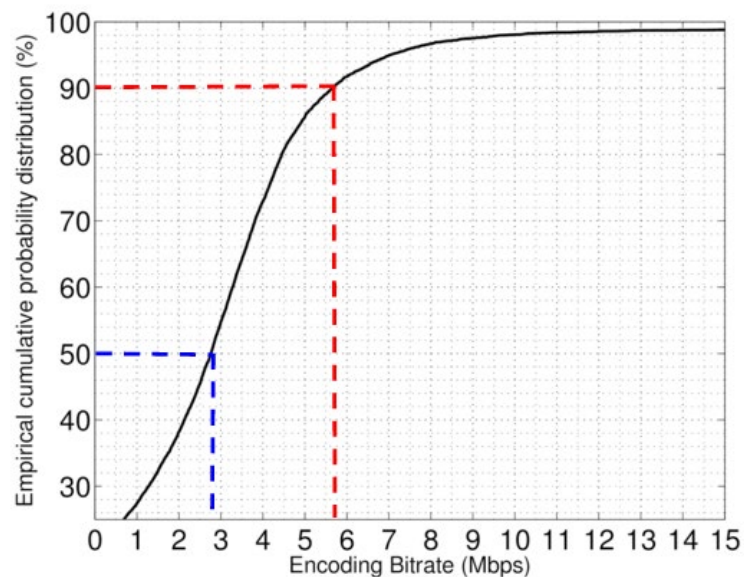
- Think of just 1 720p clip
- Encode using 2 pass CBR (constant bitrate encoding), 9.5 – 15.5 Mbps. Plot RD Curve 1
- Downsample to 480p, transcode to create R2 and upsample to 720p, then do the same. Plot RD Curve 2
- Record the “crossover bitrate” B_c
- Target bitrate of R1 is $(B_c + B_{max})/2$
- Target bitrate of R2 is kB_c where k is some fraction



How to pick target bitrates?

The bigdata approach : find some bitrates which are mostly right for everybody

- Selected 7966 720p clips from YouTube
- Measure the two RD curves for the two representations
- Record the “crossover bitrate” B_c for each clip
- Measure the CuF of the crossover bitrates and select the 90% level as the *prototype* crossover bitrate which is expected to be mostly right for 720p to 480p
- Target bitrates then use the same calculation as previously i.e. Bitrate for R2 is some fraction of that
- Bitrate of R1 is calculated similarly using the CuF of max bitrates over the corpus



A Subjective Study for the Design of ABR Streams, Chao Chen, Sasi Inguva, Andrew Rankin and Anil Kokaram, *IS&T Visual Communications and Image Processing Conference*, San Francisco Feb 2016

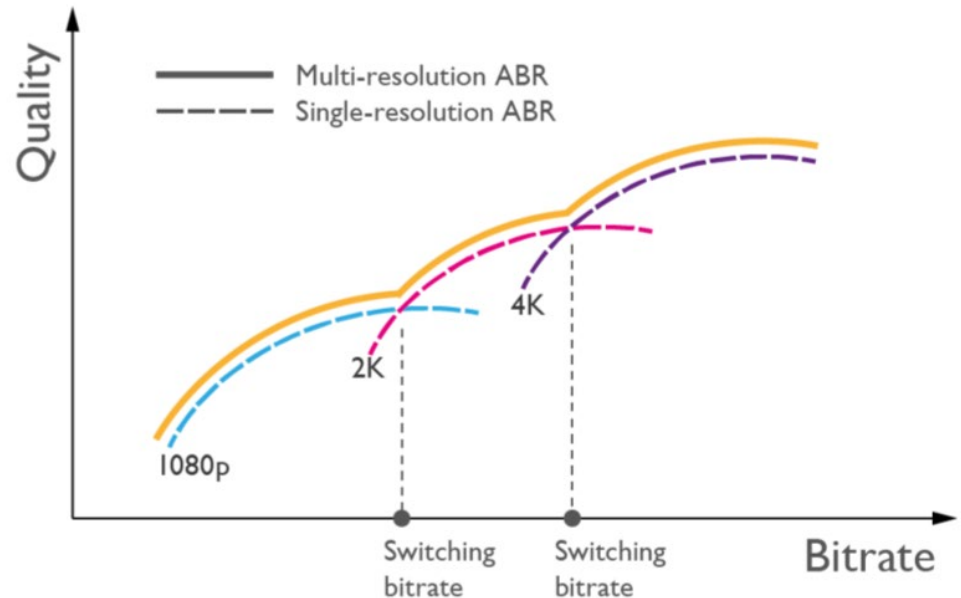
Netflix, YouTube and Per Title Encoding

Everybody uses multiresolution ABR

Everybody was using something like that bigdata recipe up till 2015

In 2012 YouTube started to experiment with per-title ideas. Based on a thing called “popular reruns”.

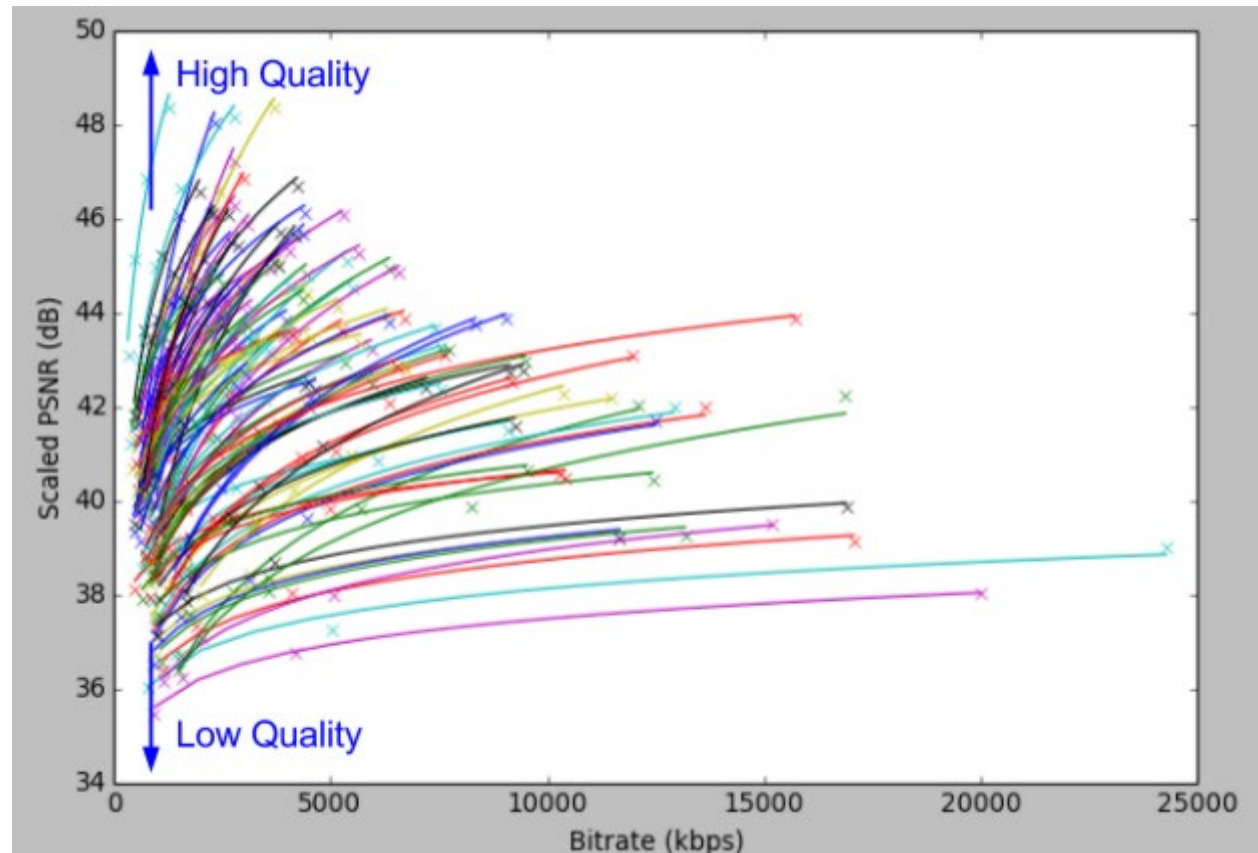
In 2015 Netflix weaponized the idea for ABR and now everyone uses this new idea for DASH representations.



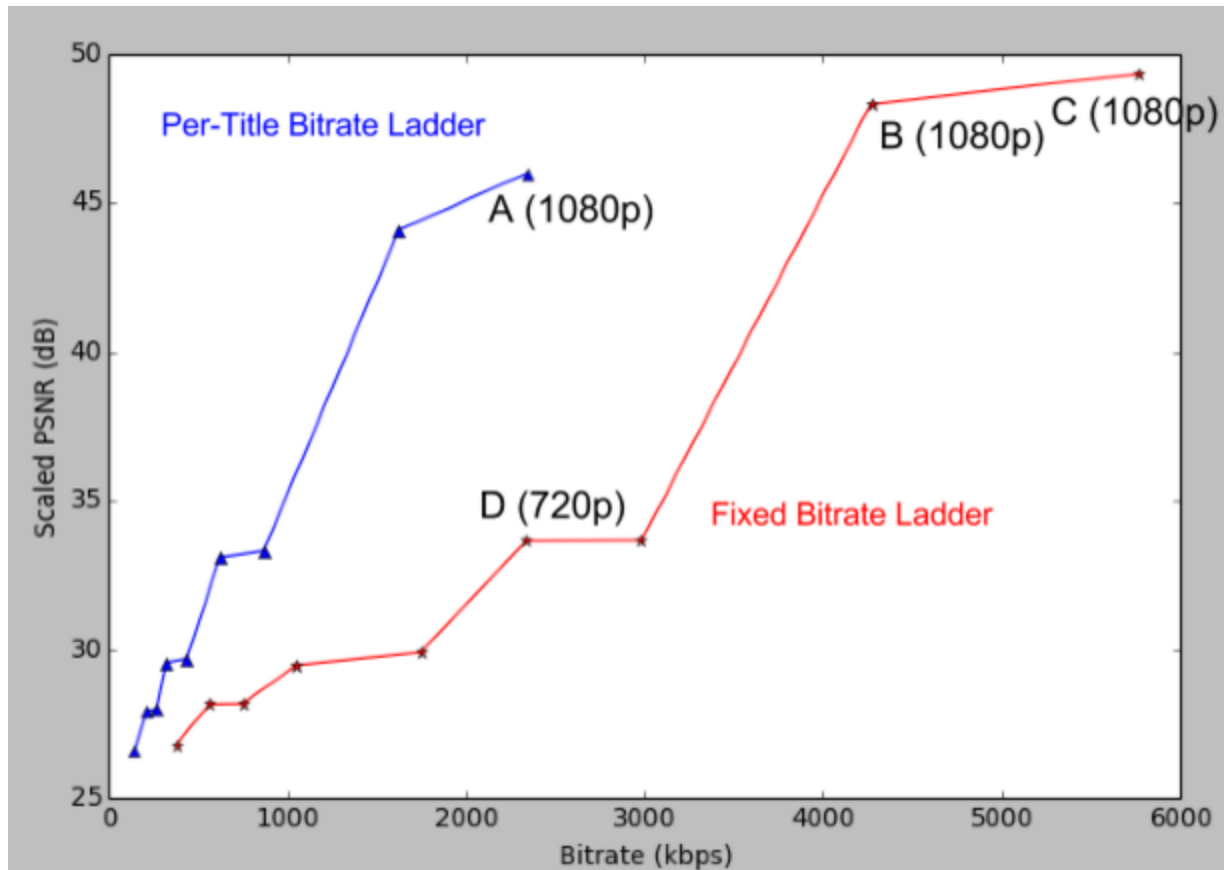
The Netflix idea

“Per Title Encoding”

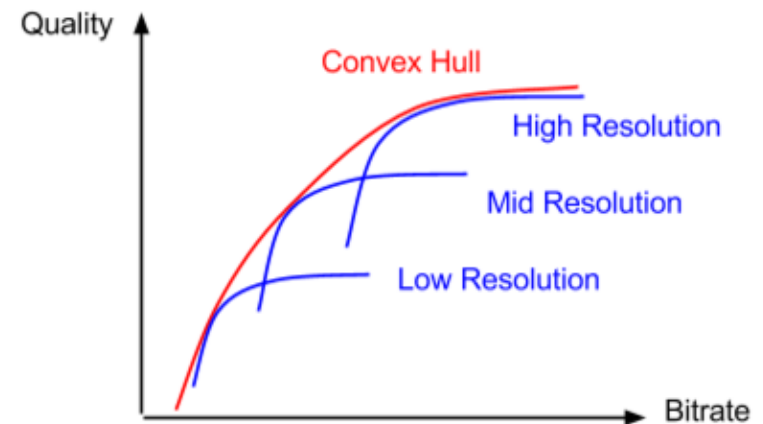
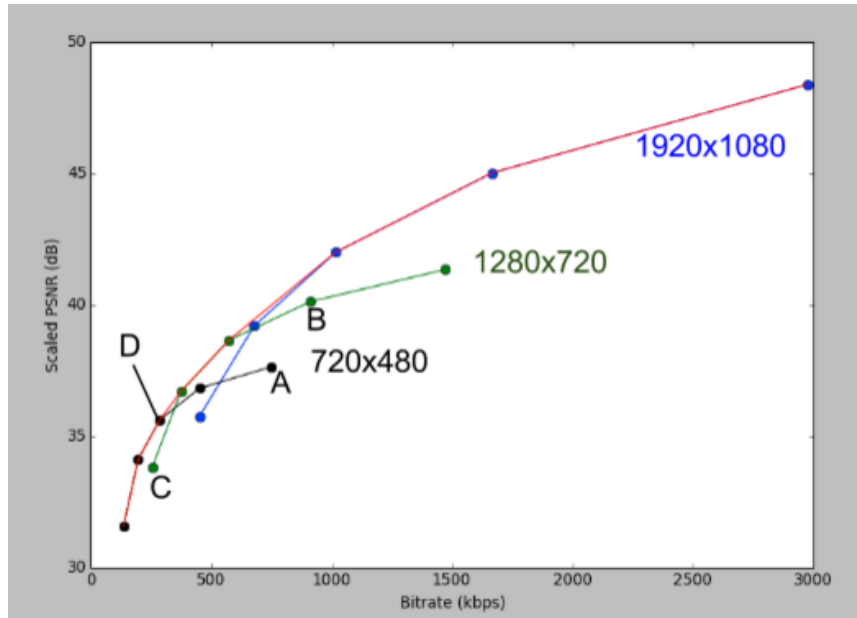
You can't really assume every clip behaves the same. The bigdata approach fudges it using that CuF recipe.



Look at what you could gain



So Netflix used brute force



Choose target bitrates “per clip” based on same RD curves as before but now use the convex hull to find the best target bitrate.

And it works

1.75 Mbps @ 480p



1.54 Mbps @ 1080p



How do you hit a “Target Bitrate”?

YouTube uses ML / DNNs

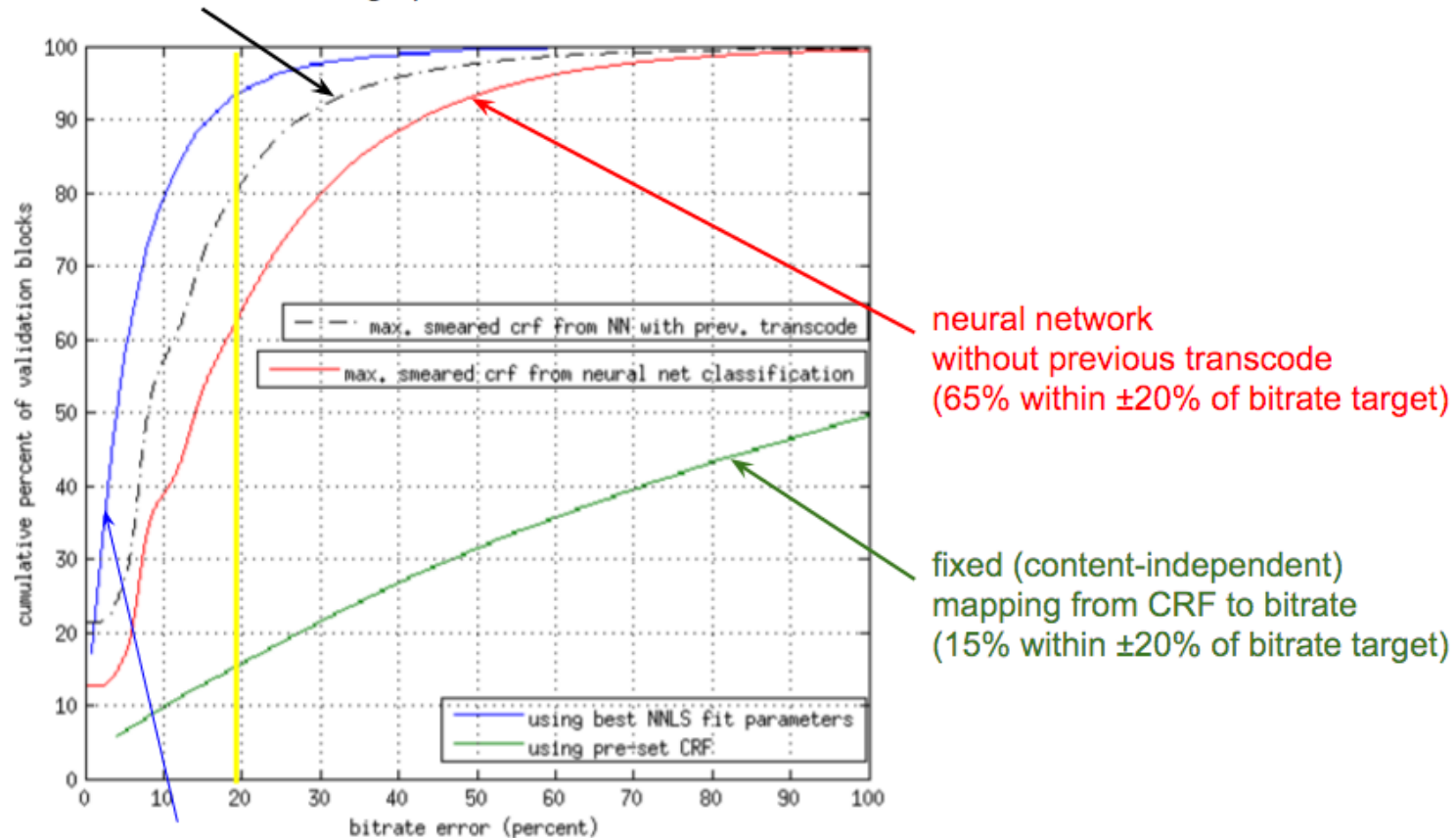
- Our problem is to choose the ffmpeg parameter (CRF) to give the bitrate we want.
- 400 hours video uploaded per minute means you can't use brute force
- So what if you could model the rate-parameter curve for a clip with a few parameters?
- Turns out you can

$$\log R(c, t, h, v) = \log K(v) - \underbrace{a(v)}_{\text{CRF}} c + \underbrace{b(v)}_{\text{fps}} \log t + \underbrace{d(v)}_{\text{Frame height}} \log h$$

- So can we predict what these parameters are (K,a,b,d) using some features measured about the video signal?
- If we can, then we can plot the rate-parameter curve and work out what CRF we need to hit the right bitrate

Machine Learning (DSP)

neural network with previous transcode
(81% within $\pm 20\%$ of bitrate target)



mapping using NNLS fit of $\log K$, a ,
and d to 145 transcodes/segment
(95% within $\pm 20\%$ of bitrate target)

References

Optimizing transcoder quality targets using a neural network with an embedded bitrate model, *Michele Covell¹, Martin Arjovsky², Yao-Chung Lin¹, and Anil Kokaram¹*; ¹Google, Inc (United States) and ²University of Buenos Aires (Argentina); SPIE Conference on Visual Information Processing and Communication, Electronic Imaging Symposium 2016, San Francisco, USA, Feb 2016

Multipass encoding for reducing pulsing artefacts in cloud based video transcoding, *Yao Chung Lin, Hugh Denman and Anil Kokaram*; Google, Inc (United States); IEEE International Conference on Image Processing, Quebec, Canada, Oct 2015

Digital Rights Management (DRM)

How iTunes, GooglePlay and Netflix exist.

Old style Satellite/DTV/Cable : Conditional Access (CA)

Have BOX will get TV

DRM = Have encryption key, can de-encrypt received file, and watch it.

Chrome uses Widevine CE (Common Encryption)

https://www.widevine.com/wv_drm.html

See “DRM”, EBU Technical Review Jan 2007, R. Leeming Red Bee Media
[OLD]

Digital Rights Management (DRM)

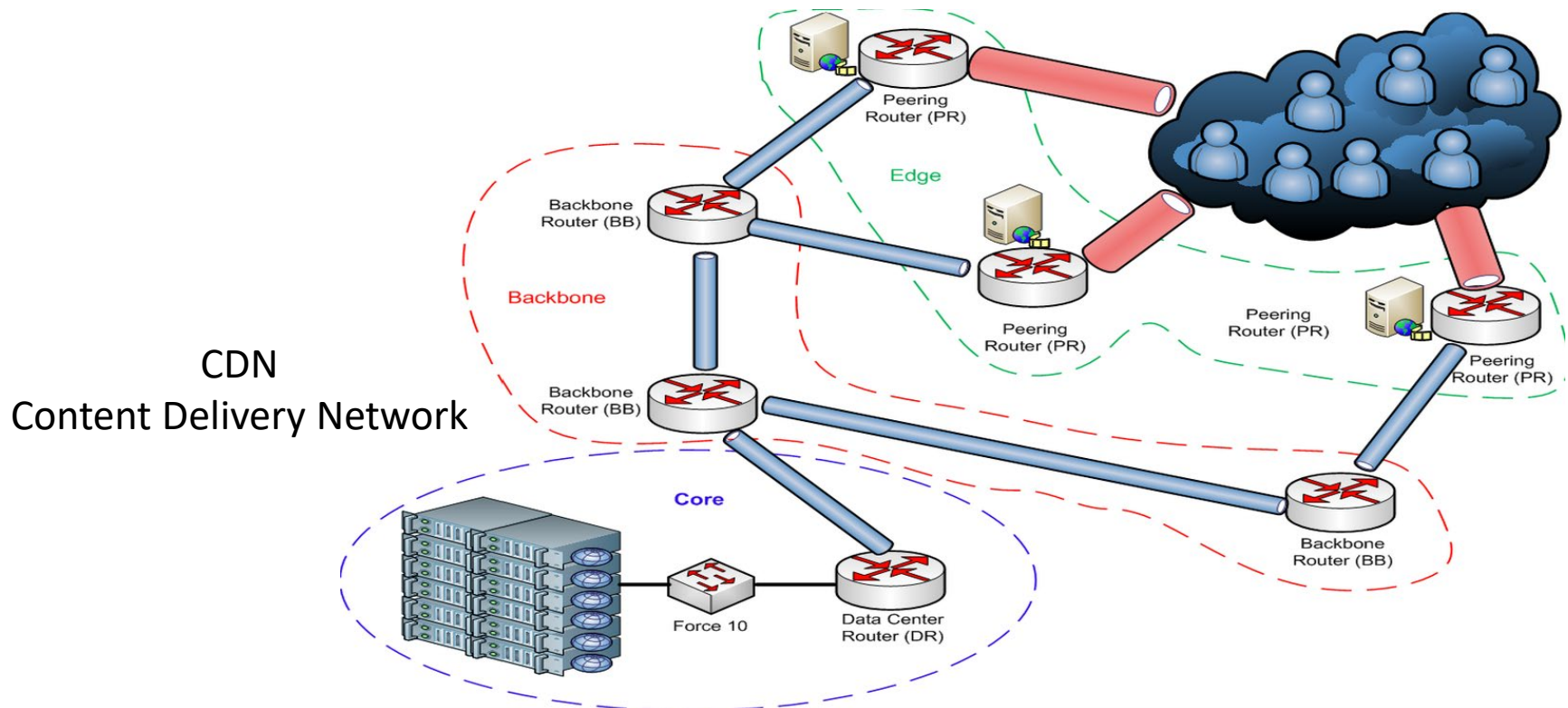
The Primary School Algorithm

1. Encrypt all DASH streams on server with a public key
2. User/Client Requests to play a movie
3. Client requests decryption private key from 3rd party server
4. 3rd party server checks that Client should have that key
5. Sends it to the client
6. Client can now unlock and view.

Network Protocols

No time for retransmit! So TCP/IP is OUT!

RTP (Real Time Protocol) , UDP (User Datagram Protocol)
Used for streaming media



Final Comments

- Many tools surround the core compression technologies that make streaming work. We didn't talk about FEC Forward Error Correction!
- DASH Adaptive Bitrate Streaming only “recent” ... 2013 Netflix and YouTube
- Transcoding is an important “glue” tool. Still unexplored in the academic community.