# ECE 634: Digital Video Systems
# Video transmission: 3/9/17

Professor Amy Reibman
MSEE 356
reibman@purdue.edu

http://engineering.purdue.edu/~reibman/ece634/index.html

## Video Communications Applications

- Interactive two-way visual communications
  - Low delay is essential
  - Real-time encoder/decoder essential
  - Audio/visual synchronization required to maintain lipsync
  - Some visual impairments may be acceptable
- One-way video streaming
  - Higher delay is OK
  - May not require real-time encoder
  - Many different rates and capabilities of decoders
- One-way video downloading
  - Video treated as a file; no different than file downloading

## What can go wrong? (from the video perspective)

- Buffer starvation
  - Required bits may not arrive in time, so there would not be enough bits at decoder to completely decode the current frame
- Packet loss
  - A large chunk of the bitstream is missing
    - Packet size depends on the network
- Bit error
  - One or more bits flip from 0 to 1 or vice versa

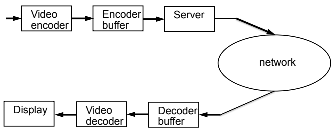
## Network characteristics and designs

- Networks are unreliable
  - Bit errors may happen; Links may fail
  - Networks are engineered to minimize this
- Networks have dynamically varying bandwidth and throughput
  - Many applications sharing a network can cause congestion
  - MAC-layer and physical layer designs increase delay
  - A mobile client/decoder may move in and out of good cell coverage
- Networks are typically designed so applications only see rare bit errors but more packet losses and/or bit-rate variations

## Network and Video Disconnect

- Many networks are engineered to reduce the high loss and error rates
  - Often this increases delay through the network
  - Even with streaming, video data is delay-sensitive
  - Once video decoding begins, it must continue or quality degrades
- Video may not be as vulnerable to packet losses and bit errors as data is
  - Data requires retransmission; any error or loss needs to be fixed
  - Video can be engineered to tolerate SOME loss and error

## Overall video transmission system



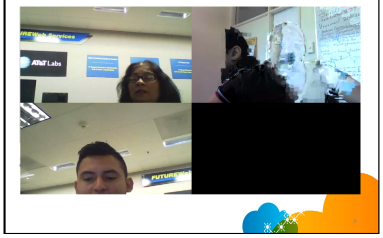
## Outline: Video transmission

- Explore impact bit errors and packet losses
- System designs:
  - What can the network do by itself
  - What can the decoder do by itself
  - What can the encoder do by itself
  - What can the encoder and decoder do together
  - What can the encoder, decoder, and network do
  - What can the decoder and the network do

## Compressed video data is sensitive to transmission errors

- Variable length coding
- Temporal predictive coding
- Spatial predictive coding
- All contribute to error propagation either within the same frame or also in following frames
- One bit error or packet loss requires careful engineering to ensure subsequently received data can still be useful

## Packet loss in SIP-based video call



## Packet losses

ARReibman, 2011

## Packet losses

ARReibman, 2011

## Packet losses

ARReibman, 2011

## Compression

ARReibman, 2011

## Bit error

ARReibman, 2011

## “Uncompressed” video

ARReibman, 2011

## Bit errors

ARReibman, 2011

## Variable-length decoding: Example revisited

- Let A={1}; B={01}; C={001}; D={0001}
- Signal to send:
  A AA C B A A D A A
  1 1 1 001 01 1 1 0001 1 1
- 1110010111000111 sent to decoder

- 1110010101000111 received by decoder
- Decoded signal:
  1 1 1 001 01 01 0001 1 1
  A A A C B B D A A

ARReibman, 2011

## Bit errors

- A single bit flips from zero to one, or one to zero
1. Variable length decoder may get lost
   - Looks similar to a packet loss
2. Decoder may not get lost
   - This may be much MUCH worse!
   - Motion vector may change sign
   - Run-length may be errored
   - DC coefficient may change sign
- Decoder may not know it has an error until much later
- Bit errors in the bitstream given to the video decoder should be avoided if at all possible!

ARReibman, 2011

## Case 1: Video decoder given all data (MPEG-2)

Frame 229

Multiple DCT errors for Design 1

ARReibman, 2011

## Case 2: Video decoder given full slices (MPEG-2)

Frame 229

Somewhat visible concealment artifacts for Design 2

ARReibman, 2011

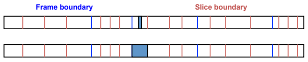
Detect a bit-error and only send video decoder "good" information

## Packet losses

- A block of bits doesn't arrive at the decoder
- Variable length coding causes major problems!
  - Don't know how much information was lost
  - Don't know where to put newly decoded information
  - Are these bits coefficients? Motion vectors?
- Solution: insert synchronization codewords
  - Easy to find: 000000000000000000000001
  - Picture_Start_Code, Slice_Start_Code
  - Thirty slices in each frame (MPEG-2)

ARReibman, 2011

## Packet losses: spatial impact



- A single loss will affect (at least) a slice of a frame
  - 16 pixels vertically, entire image horizontally
- More slice_start_codes:
  - Better quality with packet losses (less data lost)
  - Worst quality without packet losses (bits wasted)

ARReibman, 2011

## Packet losses: temporal impact

- A loss in a reference (I or P) frame will propagate with time
  - Predicting from an errored frame will propagate errors
  - Motion compensation causes errors to propagate spatially!
- Loss in a B-frame will not propagate
  - No other frames use B-frames to predict
- A correctly received I-frame will stop error propagation

ARReibman, 2011

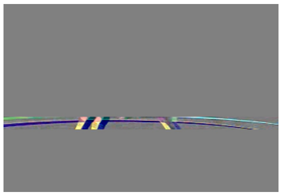
## Encoded signal (no loss)

ARReibman, 2011

## Decoded signal (with loss)

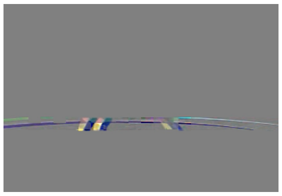
ARReibman, 2011

## Error signal (due to loss) – initial loss

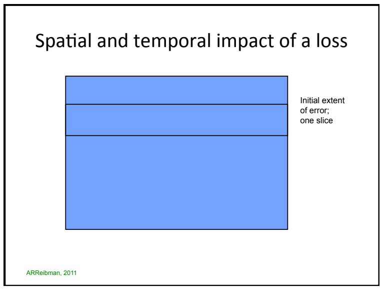


ARReibman, 2011

## Error signal (due to loss) – propagation

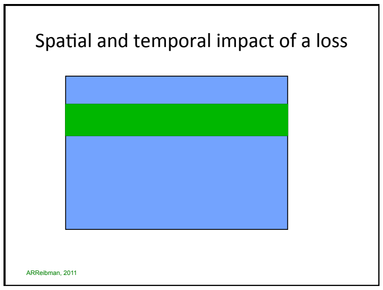


ARReibman, 2011



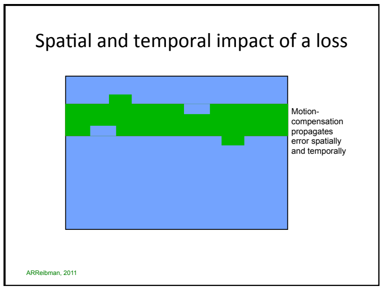
## Spatial and temporal impact of a loss

ARReibman, 2011



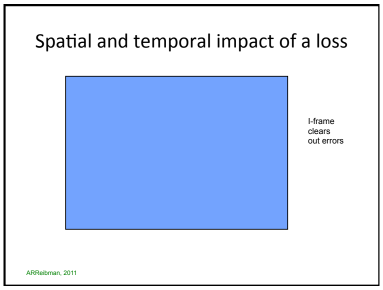
## Spatial and temporal impact of a loss

ARReibman, 2011



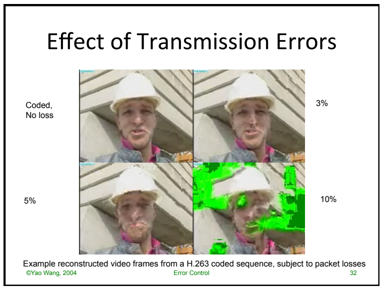
## Spatial and temporal impact of a loss

ARReibman, 2011



## Spatial and temporal impact of a loss

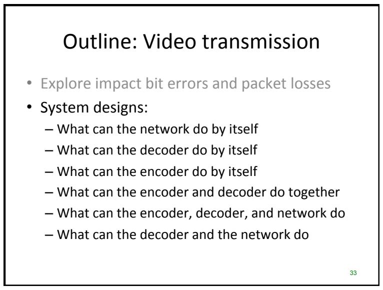
ARReibman, 2011



## Effect of Transmission Errors

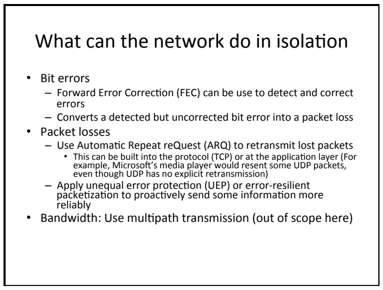
Example reconstructed video frames from a H.263 coded sequence, subject to packet losses

©Yao Wang, 2004 Error Control



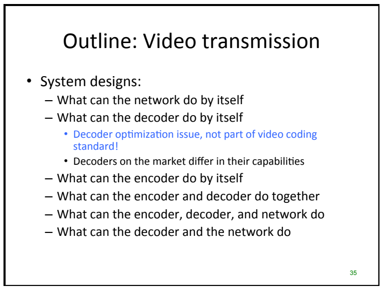
## Outline: Video transmission

- Explore impact bit errors and packet losses
- System designs:
  - What can the network do by itself
  - What can the decoder do by itself
  - What can the encoder do by itself
  - What can the encoder and decoder do together
  - What can the encoder, decoder, and network do
  - What can the decoder and the network do



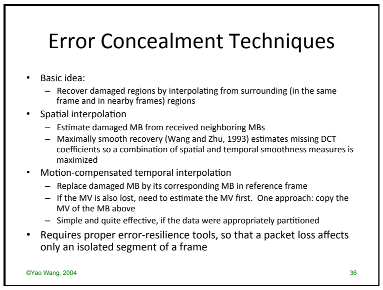
## What can the network do in isolation

- Bit errors
  - Forward Error Correction (FEC) can be use to detect and correct errors
  - Converts a detected but uncorrected bit error into a packet loss
- Packet losses
  - Use Automatic Repeat reQuest (ARQ) to retransmit lost packets
    - This can be built into the protocol (TCP) or at the application layer (For example, Microsoft's media player would resent some UDP packets, even though UDP has no explicit retransmission)
  - Apply unequal error protection (UEP) or error-resilient packetization to proactively send some information more reliably
- Bandwidth: Use multipath transmission (out of scope here)



## Outline: Video transmission

- System designs:
  - What can the network do by itself
  - What can the decoder do by itself
    - Decoder optimization issue, not part of video coding standard!
    - Decoders on the market differ in their capabilities
  - What can the encoder do by itself
  - What can the encoder and decoder do together
  - What can the encoder, decoder, and network do
  - What can the decoder and the network do



## Error Concealment Techniques

- Basic idea:
  - Recover damaged regions by interpolating from surrounding (in the same frame and in nearby frames) regions
- Spatial interpolation
  - Estimate damaged MB from received neighboring MBs
  - Maximally smooth recovery (Wang and Zhu, 1993) estimates missing DCT coefficients so a combination of spatial and temporal smoothness measures is maximized
- Motion-compensated temporal interpolation
  - Replace damaged MB by its corresponding MB in reference frame
  - If the MV is also lost, need to estimate the MV first. One approach: copy the MV of the MB above
  - Simple and quite effective, if the data were appropriately partitioned
- Requires proper error-resilience tools, so that a packet loss affects only an isolated segment of a frame

©Yao Wang, 2004

## Sample Error Concealment Results

Without concealment With concealment

## Block Matching Algorithm

- Lam, Reibman, and Liu. "Recovery of lost or erroneously received motion vectors." *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*. Vol. 5. IEEE, 1993.
- A motion-compensated concealment algorithm to estimate missing or incorrect motion vectors
- Compute the sum of the local pixel differences along the boundary of the blocks; choose the MV that minimizes this local pixel difference
- Outperforms copy-concealment (zero-motion concealment) and concealment with the immediately above MV
- Adopted (with modifications for new motion vector representations) in a non-normative annex of H.264

## Outline: Video transmission

- System designs:
  - – What can the network do by itself
  - – What can the decoder do by itself
  - – What can the encoder do by itself
    - Add redundancy to video bitstream to assist decoder recovery
  - – What can the encoder and decoder do together
    - Add redundancy to video bitstream to assist decoder recovery
  - – What can the encoder, decoder, and network do
  - – What can the decoder and the network do

## Error-Resilience – Encoder only

- Basic idea: intentionally insert redundancy in source coding to help recover from transmission errors
- Design goal: minimize the redundancy to achieve a desired level of resilience
- Error isolation (part of H.263/MPEG4 standard)
  - – Inserting sync markers
- Error resilient prediction
  - – Insert intra-mode periodically (accommodated by the standard)
  - – Independent segment prediction (part of H.263/MPEG4 standard)

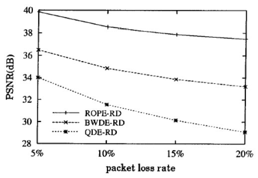
## What can the encoder do in isolation (Design issues)

- How far between each I-frame?
  - – Speed of channel changing
  - – Error resilience
  - – Compression
- How far between synchronization points?
  - – Overhead (less bit-rate for video)
  - – Better resilience given packet losses or bit-errors
- How far between each B-frame?
  - – Better compression
  - – More memory, longer delay

## RD optimized video coding with packet loss

- Goal: How to best compress video when it will be transmitted across an unreliable network
  - – Should this block be sent as an I-block or P-block?
  - – Minimize the decoder distortion due to compression AND loss, subject to total rate
- *At encoder*, for each coding option (I or P block)
  - – Compute rate
  - – Compute joint distortion *at decoder*, for encoding and packet loss
- Basic principle can be extended in many ways
  - – Include channel redundancy due to FEC/ retransmission, Scalable coding, Multiple Description Coding, etc

## Results: RD optimized video coding



- Zhang, Regunathan, and Rose, "Video coding with optimal inter/intra-mode switching for packet loss resilience", IEEE JSAC, June 2000, 18(6):966–76

## Outline: Video transmission

- System designs:
  - – What can the network do by itself
  - – What can the decoder do by itself
  - – What can the encoder do by itself
  - – What can the encoder and decoder do together
    - Add redundancy to video bitstream to assist decoder recovery
  - – What can the encoder, decoder, and network do
  - – What can the decoder and the network do

## Error-Resilience – Encoder and decoder

- Basic idea: intentionally insert redundancy in source coding to help recover from transmission errors
- Design goal: minimize the redundancy to achieve a desired level of resilience
- Error isolation (part of H.263/MPEG4 standard)
  - – Data partition (a form of scalability – frequency scalability)
- Robust binary encoding
  - – Reversible VLC (RVLC) (part of H.263/MPEG4 standard)
- Error resilient prediction
  - – Independent segment prediction (part of H.263/MPEG4 standard)
- Layered coding with unequal error protection
- Multiple description coding

## Reversible Variable Length Coding

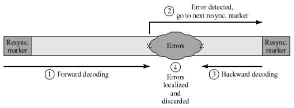


**Figure 14.7** RVLC codewords can be parsed in both the forward and backward direction, making it possible to recover more data from a corrupted data stream.

Effective primarily for bit errors. Very small additional rate.
Increased decoder complexity (when implemented).

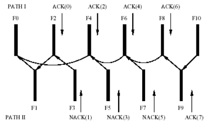
---

## Encoder-Decoder Interactive Error Control

- Reference picture selection (part of H.263/MPEG-4 standard)
  - Following a damaged frame (feedback info from receiver), use undamaged previous frame as reference frame for temporal prediction
- Error tracking
  - Determine which MBs are affected following a lost MB (feedback info), avoid using those MBs as reference pixels
- Requires a feedback channel, not necessarily involving extra coding delay

---

## Reference Picture Selection



Even/odd frames sent on separate paths. Predict damaged frames based on NACK on each path, and use undamaged frames as reference pictures.

Compatible with the RPS option in H.263+.

---

## Layered Coding with Unequal Error Protection

- LC+UEP is an attractive approach to combat channel errors
  - Base layer provides acceptable quality, enhancement layer refines the quality
  - Base layer stream is delivered through a reliable channel (by using ARQ and strong FEC)
  - Good for a network with differentiated service (Do NOT exist today over Internet, ??may become part of emerging wireless standards??)
- Problems:
  - Any error in the base layer causes severe degradation
  - Repetitive ARQ may incur unacceptable delay, strong FEC may be too complex or cause extra delay
  - The enhancement layer is useless by itself
  - The increased bit-rate from scalable coding may be too high
    - Scalability has been included in every standard since MPEG-2, which had 4 distinct methods
    - Scalability has rarely been implemented in a real system until H.264's SVC Scalable Video Coding
    - Even so, market penetration is limited

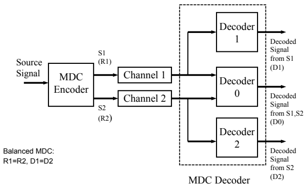
---

## Multiple Description Coding

- Assumptions:
  - Multiple channels between source and destination
  - Independent error and failure events
  - Probability that all channels fail simultaneously is low
  - All reasonable assumptions for the Internet and wireless networks, provided data are properly packetized and interleaved
- MDC: Generate multiple correlated descriptions
  - Any description provides low but acceptable quality
  - Additional received descriptions provide incremental improvements
  - No retransmission required -> low delay
  - However: correlation → reduced coding efficiency
- Design goal:
  - maximize the robustness to channel errors at a permissible level of redundancy

---

## Generic Two Description Coder



Balanced MDC:
R1=R2, D1=D2

---

## Outline: Video transmission

- System designs:
  - What can the network do by itself
  - What can the decoder do by itself
  - What can the encoder do by itself
  - What can the encoder and decoder do together
  - What can the encoder, decoder, and network do
    - Joint source/channel coding: example is layered coding with unequal error protection
    - Path diversity; example multiple descriptions sent through separate paths
    - Adapt encoding bit-rate in real-time based on network (or decoder) estimate of available bandwidth
  - What can the decoder and the network do

---

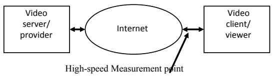
## Avoiding packet losses

- Send at the bit-rate that the network can accommodate
- First implemented/emulated by a video system in 1992

- Next few slides demonstrate how Microsoft Media Server circa 2004 used client/decoder feedback to adjust the transmitted bit-rate, and how this could be sensed inside the network

---

## Internet video streaming Microsoft Media

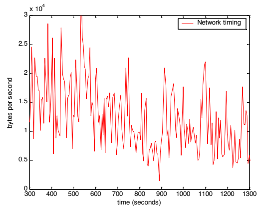


- Server may have multiple versions of same video encoded at different rates
- Server/client feedback controlled by *client*
- Client negotiates reduced-rate video from server when network gets congested
- Does the server actually respond? How? When

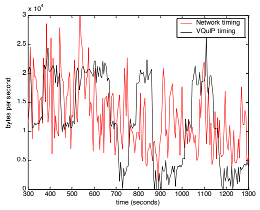
## Detecting server rate adaptation

- Server intentionally reduces transmission rate
- Switch from one video stream to another (lower-rate) stream
  - Switch from one nearly CBR stream to another
  - Detect easily using stream IDs
- Drop all non-key frames
  - Video becomes series of still frames seconds apart
  - Resulting rate very bursty
  - Look for gaps in frame IDs

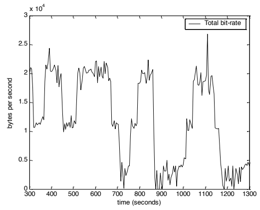
## Example: Network vs. Media Timing



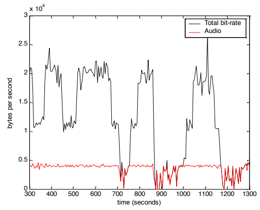
## Example: Network vs. Media Timing



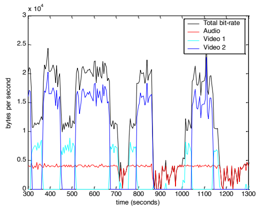
## Example: Stream analysis



## Example: Stream analysis



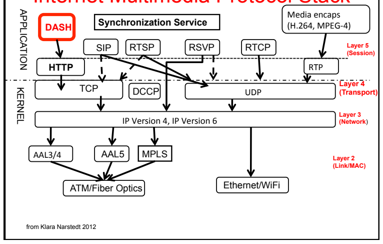
## Example: Stream analysis



## Outline: Video transmission

- System designs:
  - What can the network do by itself
  - What can the decoder do by itself
  - What can the encoder do by itself
  - What can the encoder and decoder do together
  - What can the encoder, decoder, and network do
  - What can the decoder and the network do
    - HTTP adaptive streaming

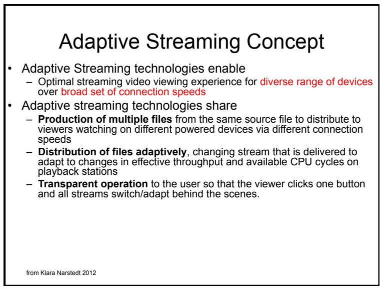
## Internet Multimedia Protocol Stack



from Klara Narstedt 2012

## Video Streaming Technologies

- Adobe Flash progressive download
- Apple HTTP Live Streaming (HLS)
- Microsoft Smooth Streaming (Netflix)
- Adobe HTTP Dynamic Streaming
- DASH – Dynamic Adaptive Streaming over HTTP (an MPEG standard)

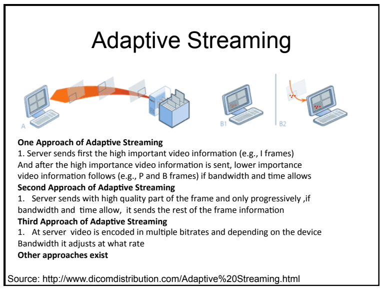


## Adaptive Streaming Concept

- Adaptive Streaming technologies enable
  - Optimal streaming video viewing experience for diverse range of devices over broad set of connection speeds
- Adaptive streaming technologies share
  - **Production of multiple files** from the same source file to distribute to viewers watching on different powered devices via different connection speeds
  - **Distribution of files adaptively**, changing stream that is delivered to adapt to changes in effective throughput and available CPU cycles on playback stations
  - **Transparent operation** to the user so that the viewer clicks one button and all streams switch/adapt behind the scenes.

from Klara Narstedt 2012

---



## Adaptive Streaming

**One Approach of Adaptive Streaming**

1. Server sends first the high important video information (e.g., I frames) And after the high importance video information is sent, lower importance video information follows (e.g., P and B frames) if bandwidth and time allows

**Second Approach of Adaptive Streaming**

1. Server sends with high quality part of the frame and only progressively ,if bandwidth and time allow, it sends the rest of the frame information

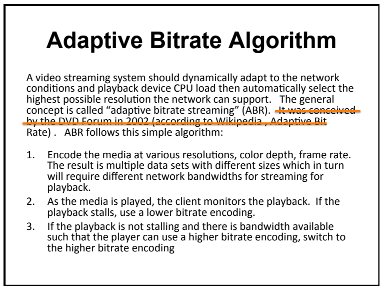
**Third Approach of Adaptive Streaming**

1. At server video is encoded in multiple bitrates and depending on the device Bandwidth it adjusts at what rate

**Other approaches exist**

Source: http://www.dicomdistribution.com/Adaptive%20Streaming.html

---

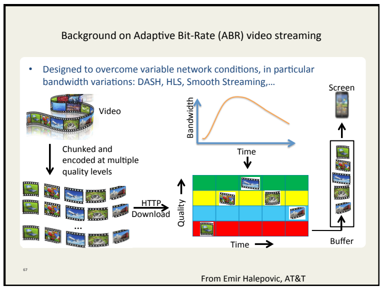


## Adaptive Bitrate Algorithm

A video streaming system should dynamically adapt to the network conditions and playback device CPU load then automatically select the highest possible resolution the network can support. The general concept is called "adaptive bitrate streaming" (ABR). ~~It was conceived by the DVD Forum in 2002 (according to Wikipedia , Adaptive Bit~~ Rate) . ABR follows this simple algorithm:

1. Encode the media at various resolutions, color depth, frame rate. The result is multiple data sets with different sizes which in turn will require different network bandwidths for streaming for playback.
2. As the media is played, the client monitors the playback. If the playback stalls, use a lower bitrate encoding.
3. If the playback is not stalling and there is bandwidth available such that the player can use a higher bitrate encoding, switch to the higher bitrate encoding

---

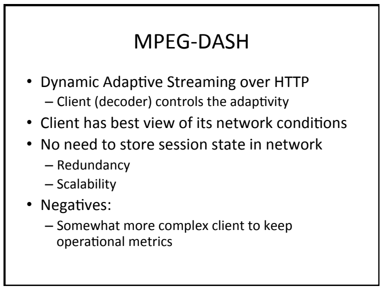


## Background on Adaptive Bit-Rate (ABR) video streaming

- Designed to overcome variable network conditions, in particular bandwidth variations: DASH, HLS, Smooth Streaming,…

From Emir Halepovic, AT&T

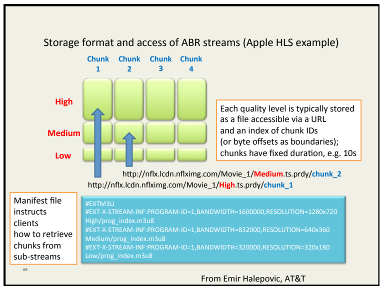
---



## MPEG-DASH

- Dynamic Adaptive Streaming over HTTP
  - Client (decoder) controls the adaptivity
- Client has best view of its network conditions
- No need to store session state in network
  - Redundancy
  - Scalability
- Negatives:
  - Somewhat more complex client to keep operational metrics

---



## Storage format and access of ABR streams (Apple HLS example)

Each quality level is typically stored as a file accessible via a URL and an index of chunk IDs (or byte offsets as boundaries); chunks have fixed duration, e.g. 10s

http://nflx.lcdn.nflximg.com/Movie_1/Medium.ts.prdy/chunk_2

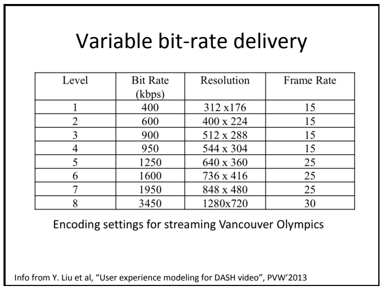
http://nflx.lcdn.nflximg.com/Movie_1/High.ts.prdy/chunk_1

Manifest file instructs clients how to retrieve chunks from sub-streams

```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1600000,RESOLUTION=1280x720
High/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=832000,RESOLUTION=640x360
Medium/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=320000,RESOLUTION=320x180
Low/prog_index.m3u8
```

From Emir Halepovic, AT&T

---



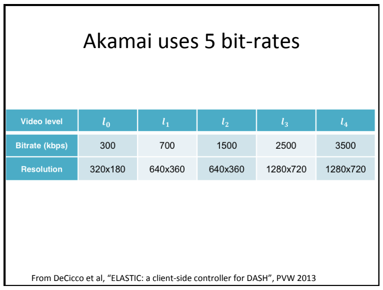
## Variable bit-rate delivery

| Level | Bit Rate (kbps) | Resolution | Frame Rate |
|---|---|---|---|
| 1 | 400 | 312 x176 | 15 |
| 2 | 600 | 400 x 224 | 15 |
| 3 | 900 | 512 x 288 | 15 |
| 4 | 950 | 544 x 304 | 15 |
| 5 | 1250 | 640 x 360 | 25 |
| 6 | 1600 | 736 x 416 | 25 |
| 7 | 1950 | 848 x 480 | 25 |
| 8 | 3450 | 1280x720 | 30 |

Encoding settings for streaming Vancouver Olympics

Info from Y. Liu et al, "User experience modeling for DASH video", PVW'2013

---

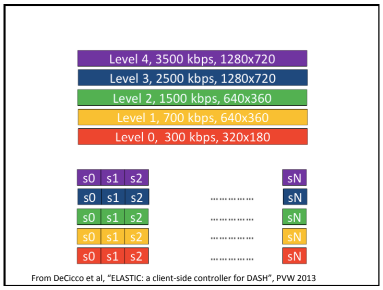


## Akamai uses 5 bit-rates

| Video level | $l_0$ | $l_1$ | $l_2$ | $l_3$ | $l_4$ |
|---|---|---|---|---|---|
| Bitrate (kbps) | 300 | 700 | 1500 | 2500 | 3500 |
| Resolution | 320x180 | 640x360 | 640x360 | 1280x720 | 1280x720 |

From DeCicco et al, "ELASTIC: a client-side controller for DASH", PVW 2013

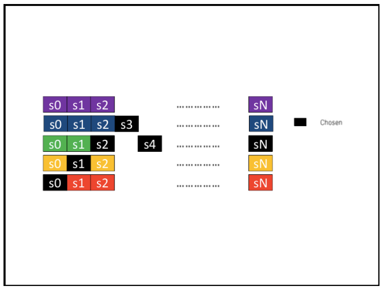
---



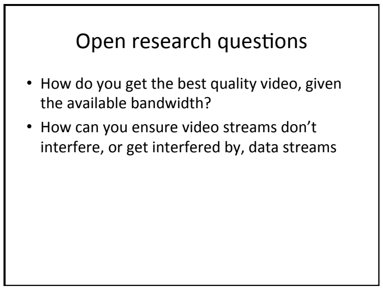
Level 4, 3500 kbps, 1280x720
Level 3, 2500 kbps, 1280x720
Level 2, 1500 kbps, 640x360
Level 1, 700 kbps, 640x360
Level 0, 300 kbps, 320x180

s0 s1 s2 …………… sN
s0 s1 s2 …………… sN
s0 s1 s2 …………… sN
s0 s1 s2 …………… sN
s0 s1 s2 …………… sN

From DeCicco et al, "ELASTIC: a client-side controller for DASH", PVW 2013

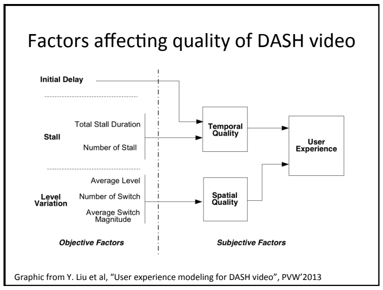

s0 s1 s2 ………… sN
s0 s1 s2 s3 ………… sN
s0 s1 s2 s4 ………… sN
s0 s1 s2 ………… sN
s0 s1 s2 ………… sN

■ Chosen



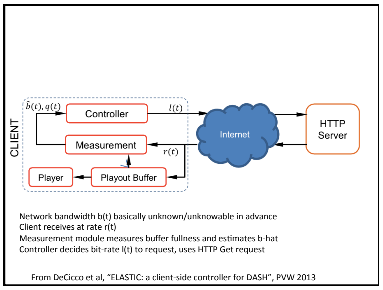
## Open research questions

- How do you get the best quality video, given the available bandwidth?
- How can you ensure video streams don't interfere, or get interfered by, data streams



## Factors affecting quality of DASH video

Graphic from Y. Liu et al, "User experience modeling for DASH video", PVW'2013



Network bandwidth b(t) basically unknown/unknowable in advance
Client receives at rate r(t)
Measurement module measures buffer fullness and estimates b-hat
Controller decides bit-rate l(t) to request, uses HTTP Get request

From DeCicco et al, "ELASTIC: a client-side controller for DASH", PVW 2013