Video

Lecture 9

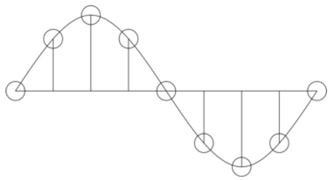
http://www.cs.rutgers.edu/~sn624/352-F24

Srinivas Narayana

Rutgers University | New Brunswick

Digital representation of audio and video

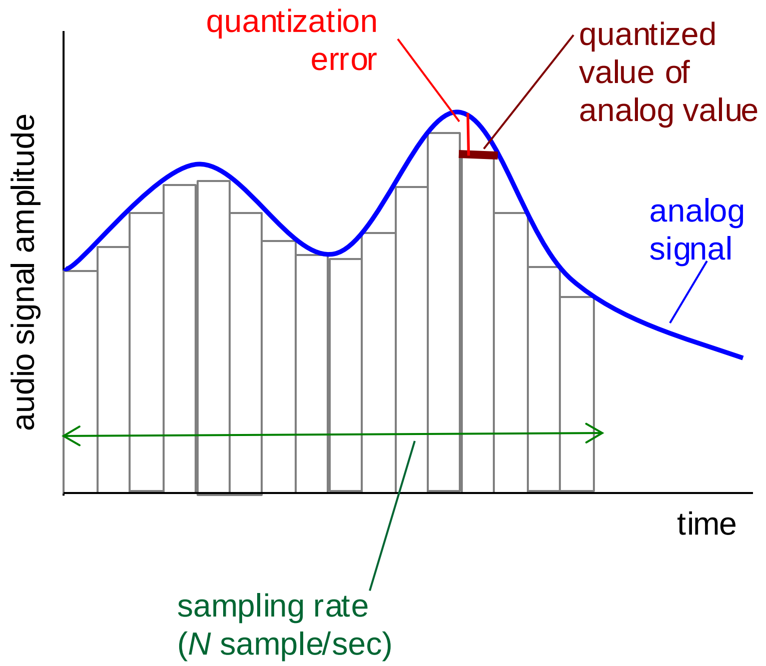
Digital representation of audio



- Must convert analog signal to digital representation
- Sample
  - How many times (twice the max frequency in the signal)
- Quantize
  - How many levels or bits to represent each sample
  - More levels ➔ more accurate representation of signal
  - More levels ➔ more bits to store & need more bandwidth to transmit
- Compress
  - Compact representation of quantized values

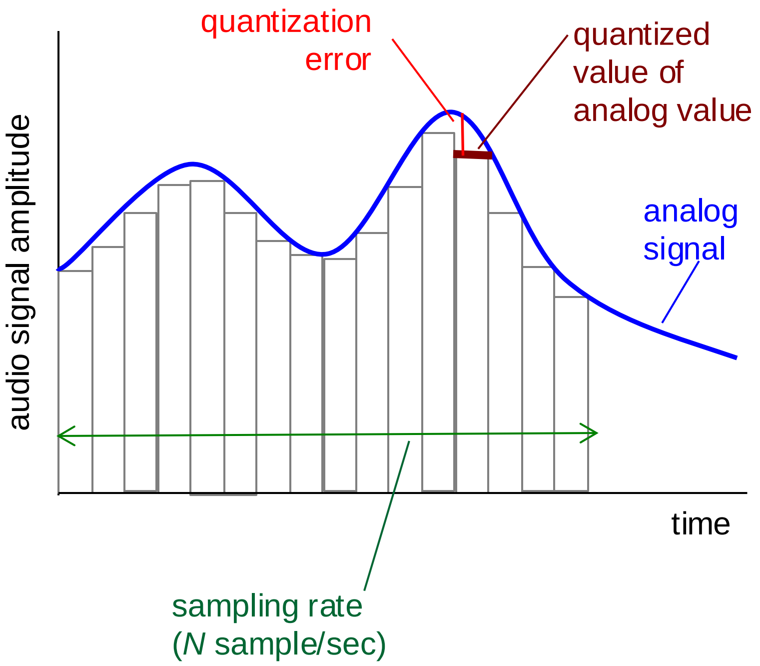
Audio representation

- analog audio signal sampled at constant rate
  - telephone: 8,000 samples/sec
  - CD music: 44,100 samples/sec
- each sample quantized, i.e., rounded
  - e.g., 2^8=256 possible quantized values
  - each quantized value represented by bits, e.g., 8 bits for 256 values



Audio representation

- example: 8,000 samples/sec, 256 quantized values
- Bandwidth needed: 64,000 bps
- receiver converts bits back to analog signal:
  - some quality reduction



Example rates

- CD: 1.411 Mbps
- MP3: 96, 128, 160 Kbps
- Internet telephony: 5.3 Kbps and up

Video representation

- Video: sequence of images displayed at constant rate
  - e.g., 30 images/sec
  - Appear continuous due to the stroboscopic effect

frame *i*

frame *i+1*

Video representation

- Digital image: array of pixels
  - each pixel represented by bits
  - Encode luminance and color
  - Number of pixels: resolution
- Coding: use redundancy *within* and *between* images to decrease # bits used to encode image
  - spatial (within image)
  - temporal (from one image to next)
- Encoding/decoding algorithm often called a codec

spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of repeated values* (N)

frame *i*

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i (motion vectors)

frame *i+1*

Video codecs: terminology

- Video *bit rate*: effective number of bits per second of the video after encoding
- It depends on many factors
  - Resolution of each image: more pixels = more bits
  - Detail per pixel: more luminance & color detail = more bits
  - Amount of movement in the video. More movement = more bits
  - Quality of overall compression in the codec
- Video bit rate is typically correlated with quality of perception
  - Higher bit rate == better to perceive

Bit-rates: terminology

- Bit-rate of a video changes over the duration of the video
- CBR: (constant bit rate): fixed bit-rate video
- VBR: (variable bit rate): different parts of the video have different bit rates, e.g., changes in color, motion, etc.
    - For VBR, we talk about average bit-rate over video's duration
- Examples of average video bit-rates
    - MPEG 1 (CD-ROM) 1.5 Mbps. MPEG2 (DVD) 3-6 Mbps
    - MPEG4 (often used in Internet, < 1 Mbps)
    - In general, one Internet video stream takes up a few Mbit/s (more for HD)

https://blog.video.ibm.com/streaming-video-tips/what-is-video-encoding-codecs-compression-techniques/

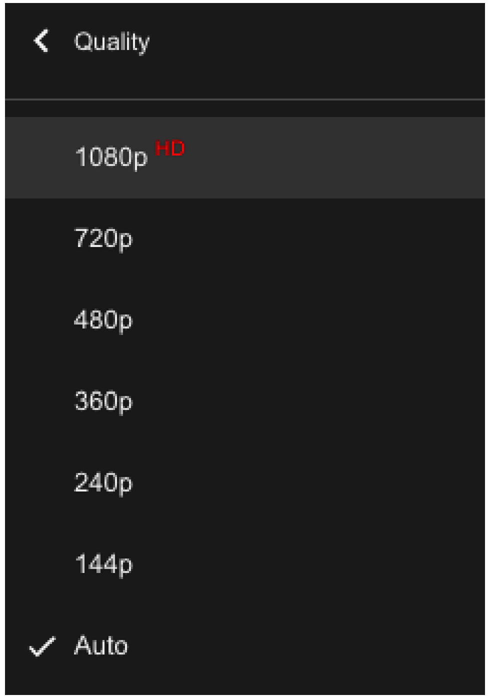
Networking multimedia: 3 types

- On-demand streamed video/audio
  - Can begin playout before downloading the entire file
  - Ful video/audio stored at the server: able to transmit faster than audio/video will be rendered (with storing/buffering at client)
  - e.g., Spotify, YouTube, Netflix
- Conversational voice or video over IP
  - interactive human-to-human communication limits delay tolerance
  - e.g., Zoom
- Live streamed audio, video
  - e.g, sporting event on sky sports
  - Can delay a little, but must be close to the “live edge” of content

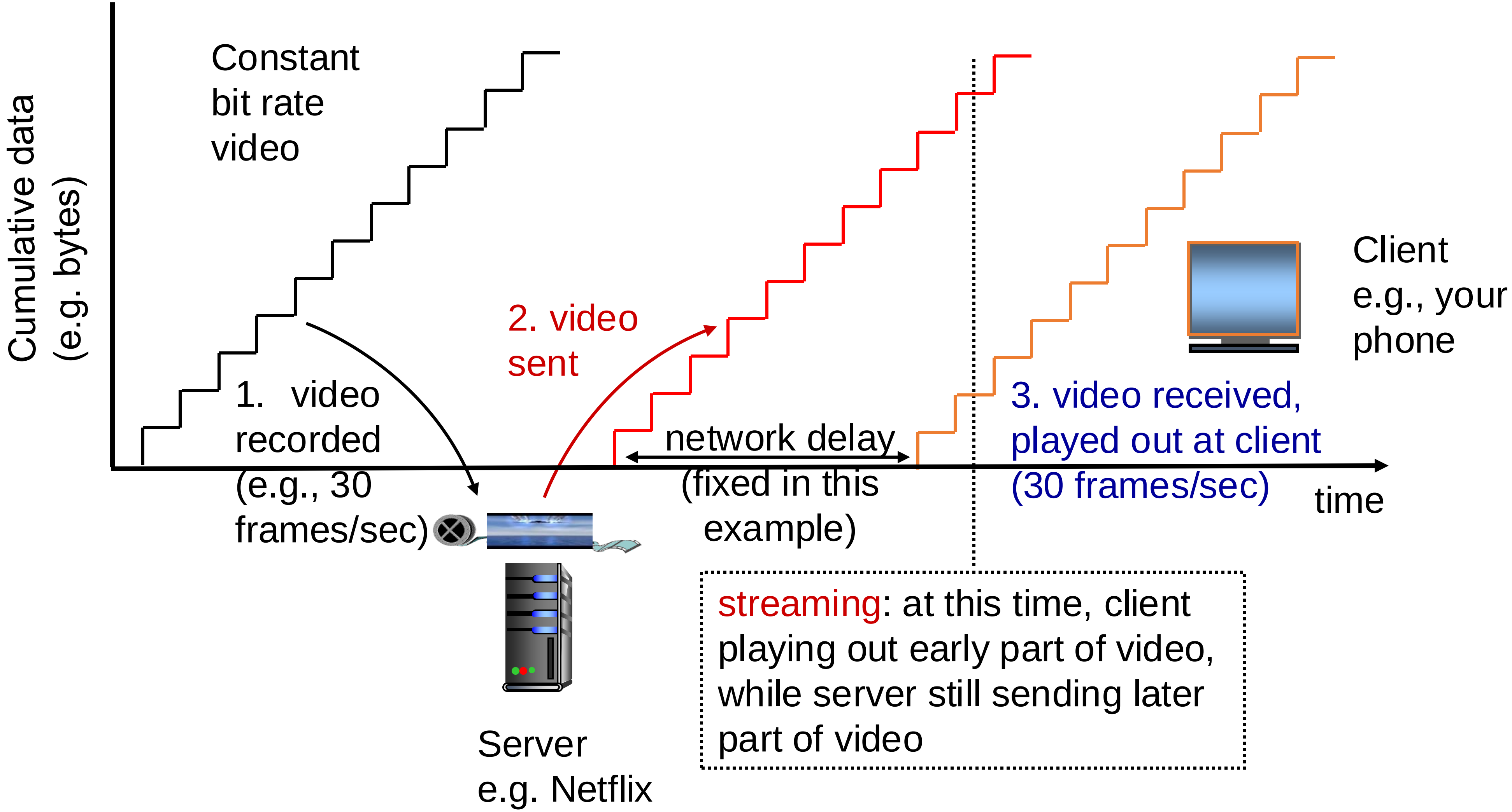
On-demand Video Streaming

Streaming (stored) video

- Media is prerecorded at different qualities
  - Available in storage at the server
- Client downloads an initial portion and starts viewing
  - The rest is downloaded as time progresses
  - No need for user to wait for entire content to be downloaded (streaming)
- Can change the quality of the content and where it's fetched mid-stream

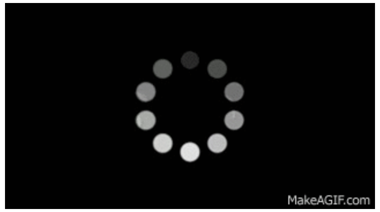


Streaming stored video



Streaming stored video: challenges

- Continuous playout constraint: once video playout begins at client, time gap between frames must match the original time gap in the video (why?)

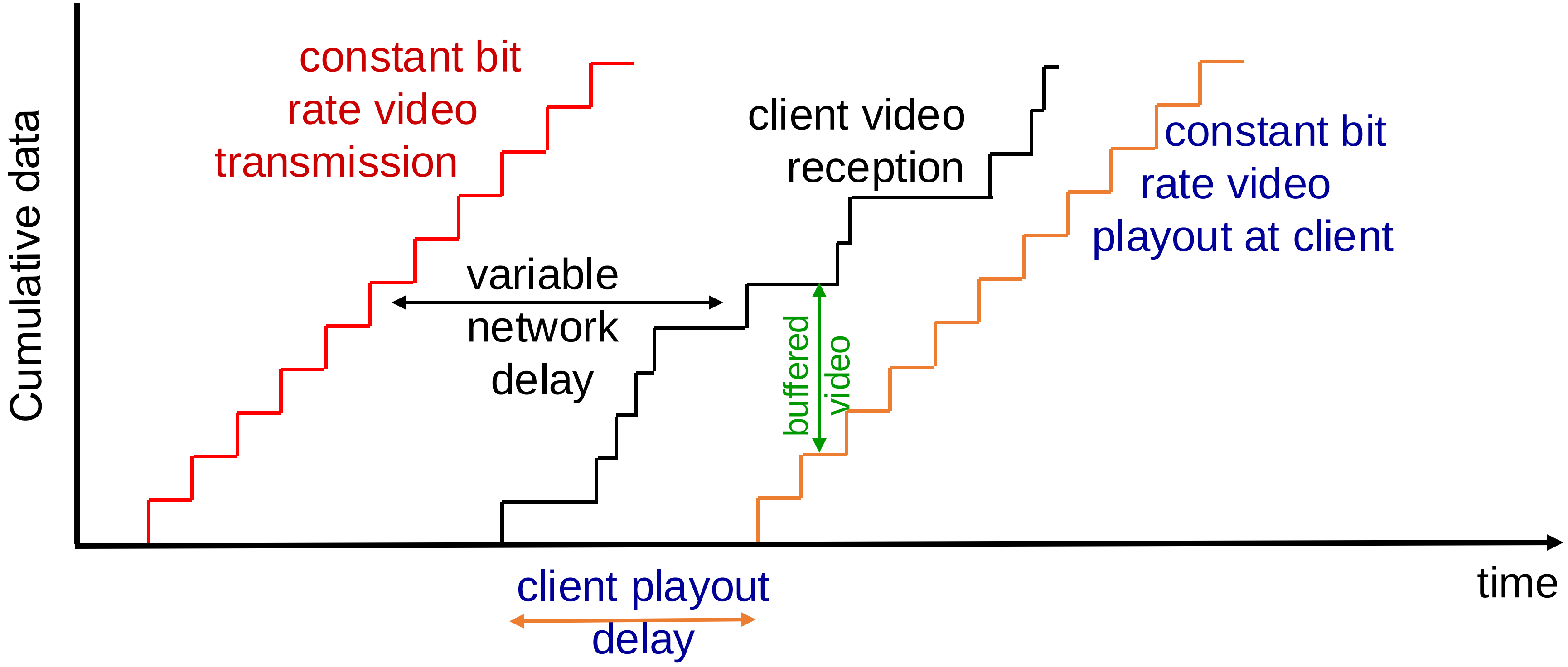


- But network delays are variable!

- Clients have a client-side buffer of downloaded video to absorb variation in network delay, available bandwidth

- The video buffer also helps with user interactions: pause, fast-forward, rewind, jump through video

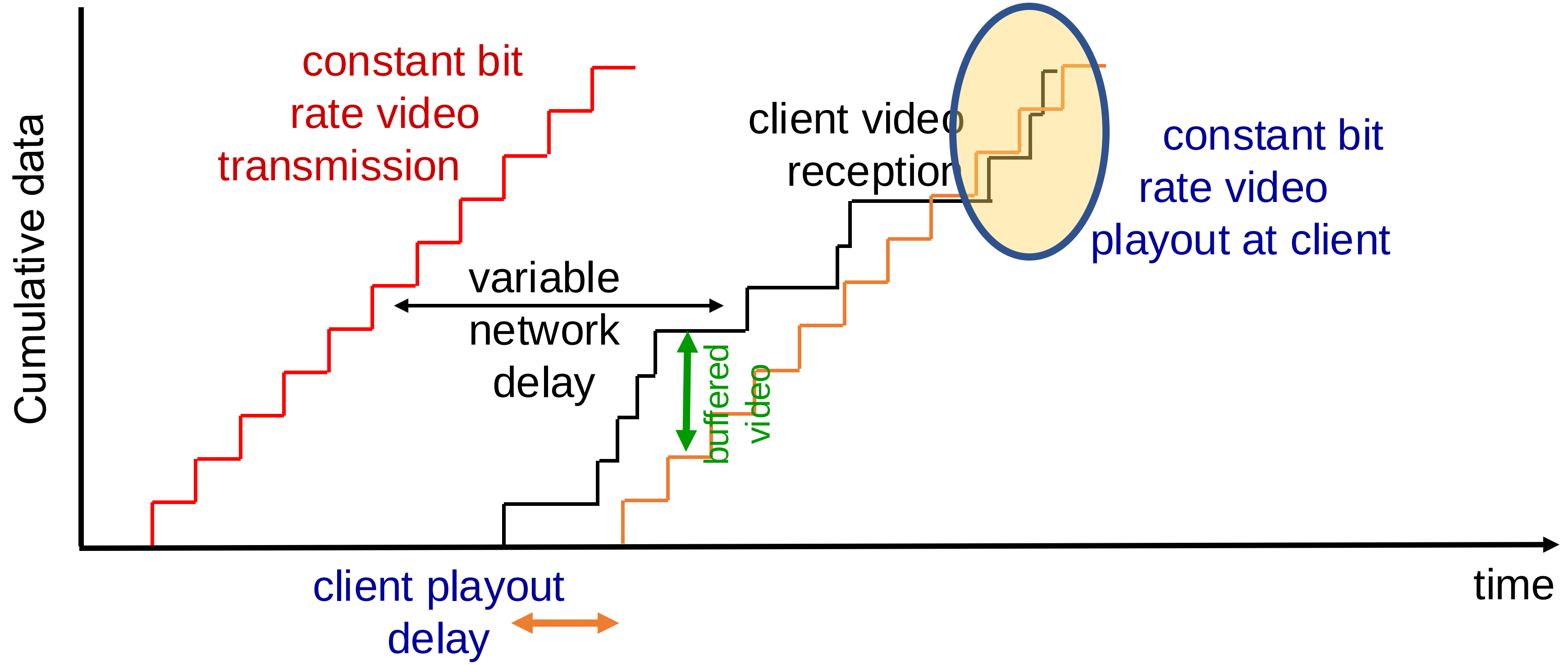
Introduce a delay for smooth playout



Client-side buffering with playout delay:

compensate for variations in the network delay

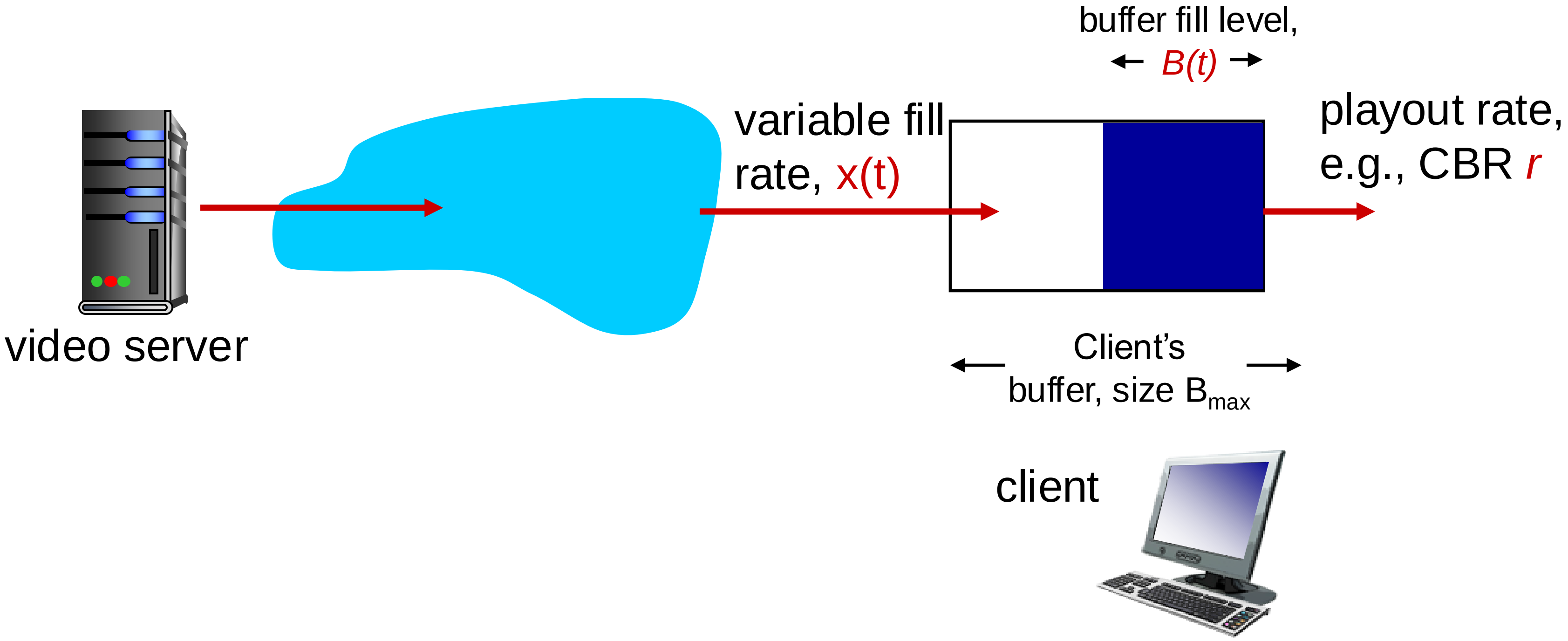
But not too small a delay



Playout delay that's too small can cause stalls

There's nothing in the buffer to show to the user

Client-side buffering, playout

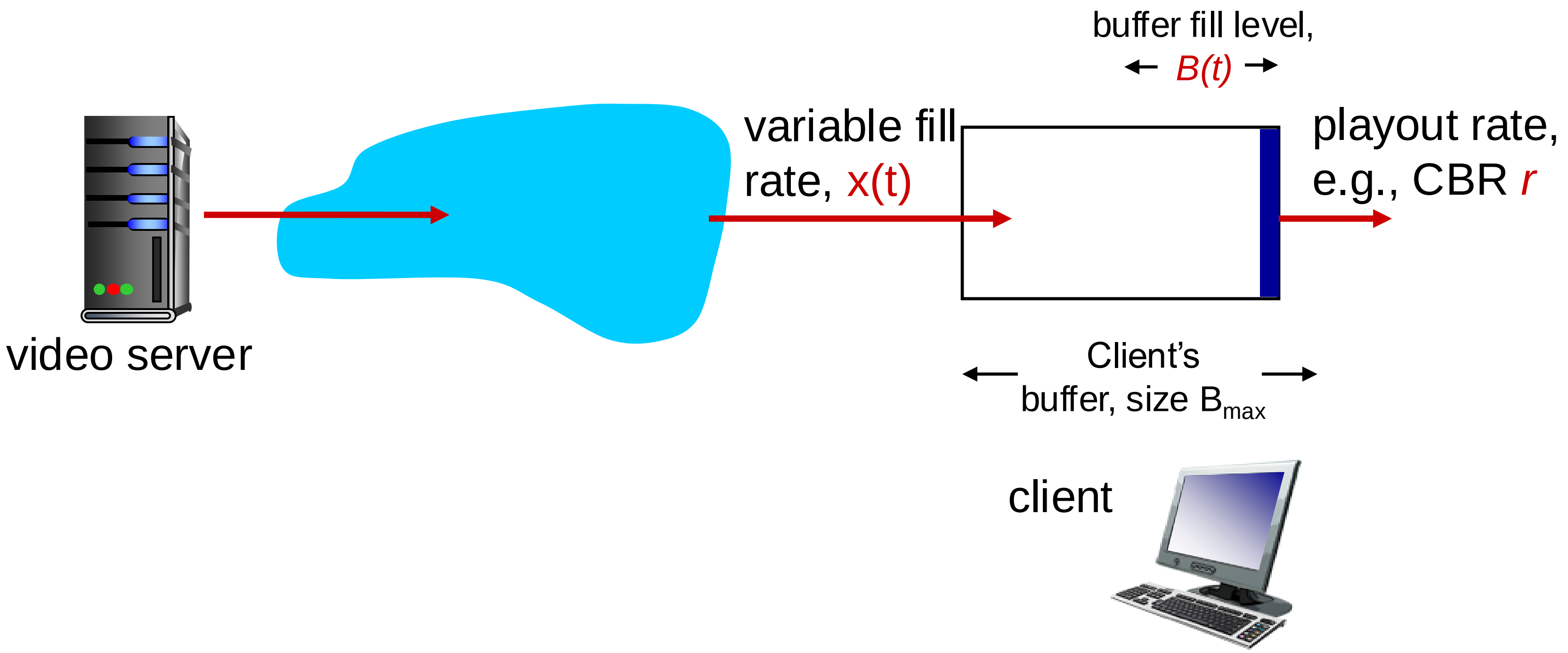


Most video is broken up in time into multiple segments

Client downloads video segment by segment

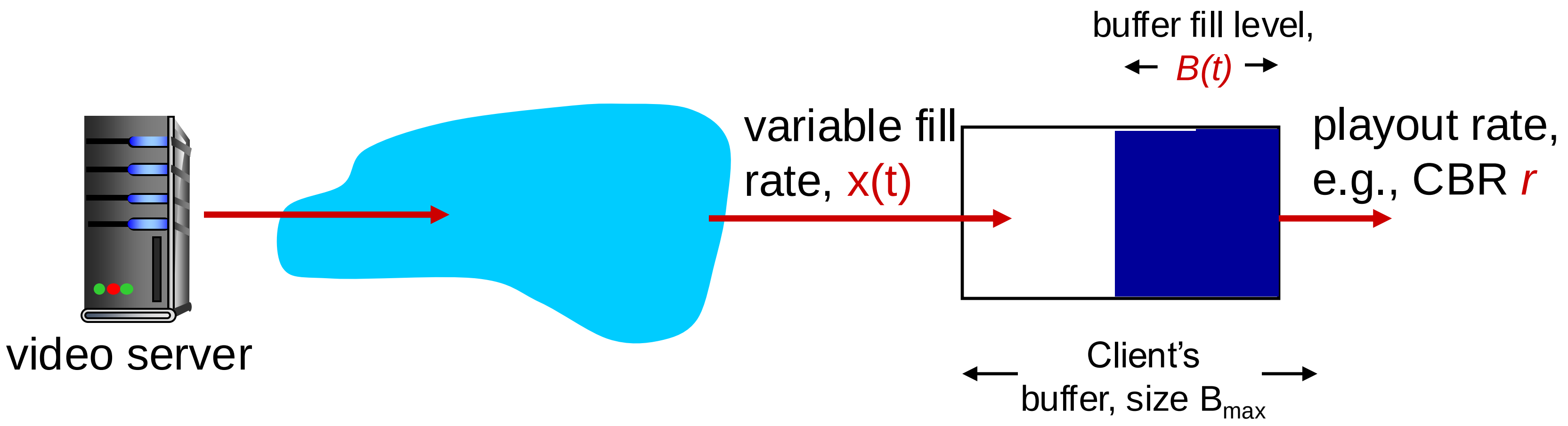
For example: a segment might be 4 seconds worth of video.

Client-side buffering, playout



1. Initial fill of buffer until playout begins at t_p
2. playout begins at t_p
3. buffer fill level varies over time as fill rate x(t) varies (assume playout rate r is constant for now)

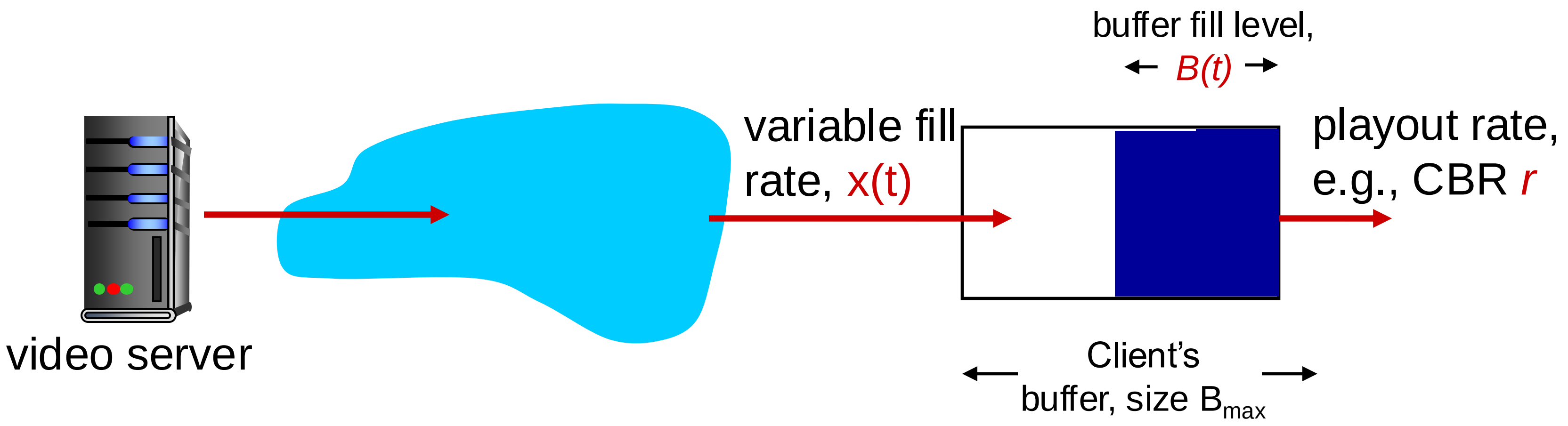
Client-side buffering, playout



playout buffering: average fill rate ($\bar{x}$), playout rate (r):

- $\bar{x} < r$: buffer eventually empties for a sufficiently long video. Stall and rebuffering
- $\bar{x} > r$: buffer will not empty, provided the initial playout delay is large enough to absorb variability in x(t)
  - *initial playout delay tradeoff:* buffer starvation less likely with larger delay, but also incur a larger delay until the user begins watching

Client-side buffering, playout

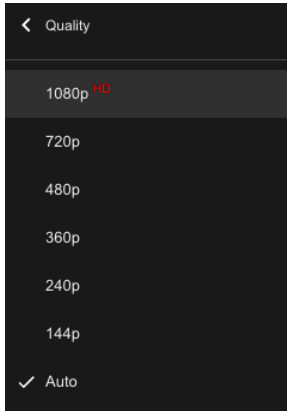


playout buffering: average fill rate ($\bar{x}$), playout rate (r):

- is $\bar{x}$ < r or $\bar{x}$ > r for a given network connection?
- It can be hard to control x or even predict it in general
  - Best-effort network inflicts long queues, low bandwidth, loss, etc.
- How to set the playout rate r?
  - Too low a bit-rate r: video has poorer quality than needed
  - Too high a bit-rate r: buffer might empty out. Stall/rebuffering!

Adaptive bit–rate video

- Motivation: Want to provide high quality video experience, without stalls
- Observations:
  - Videos come in different qualities (average bit rates)
  - Versions of the video for different quality levels readily available
  - Different segments of video can be downloaded separately



- Adapt bit rate per segment through collaboration between the video client (e.g., your browser) and the server (e.g., @ Netflix)
- Adaptive bit-rate (ABR) video: change the bit-rate (quality) of next video segment based on network and client conditions
- A typical strategy: Buffer-based rate adaptation

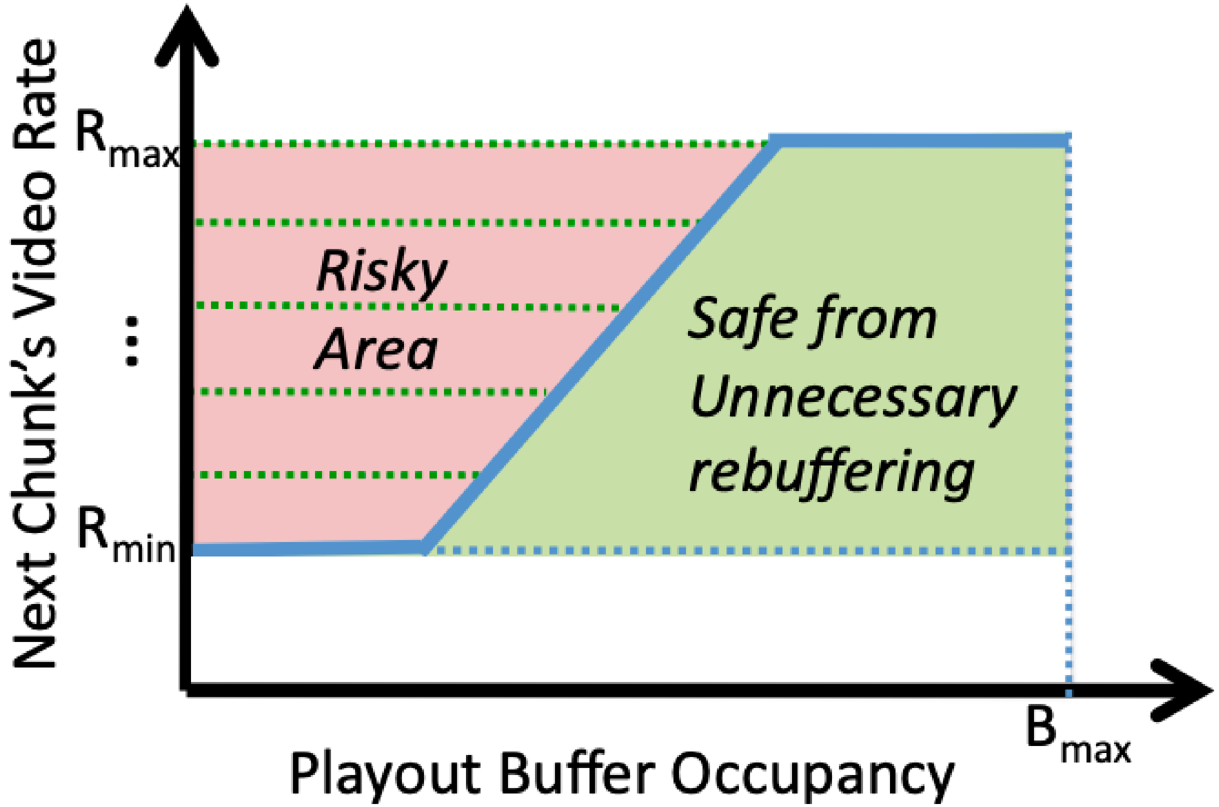
Buffer-based bit-rate adaptation

- Key idea: If there is a large stored buffer of video, optimize for video quality, i.e., high bit rates

- Else (i.e., buffer has low occupancy), avoid stalls by being conservative and ask for a lower quality (bit-rate)
  - Hope: lower bandwidth requirement of a lower quality stream is satisfiable more easily

- Buffer is measured in seconds of playout left before stalling

Buffer-based bit-rate adaptation



A highly effective method to provide high video quality despite variable and intermittently poor network conditions.

Used by Netflix.

http://yuba.stanford.edu/~nickm/papers/sigcomm2014-video.pdf

A Buffer-Based Approach to Rate Adaptation

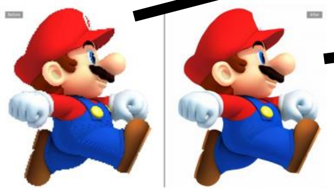
Dynamic Adaptive Streaming over HTTP (DASH)

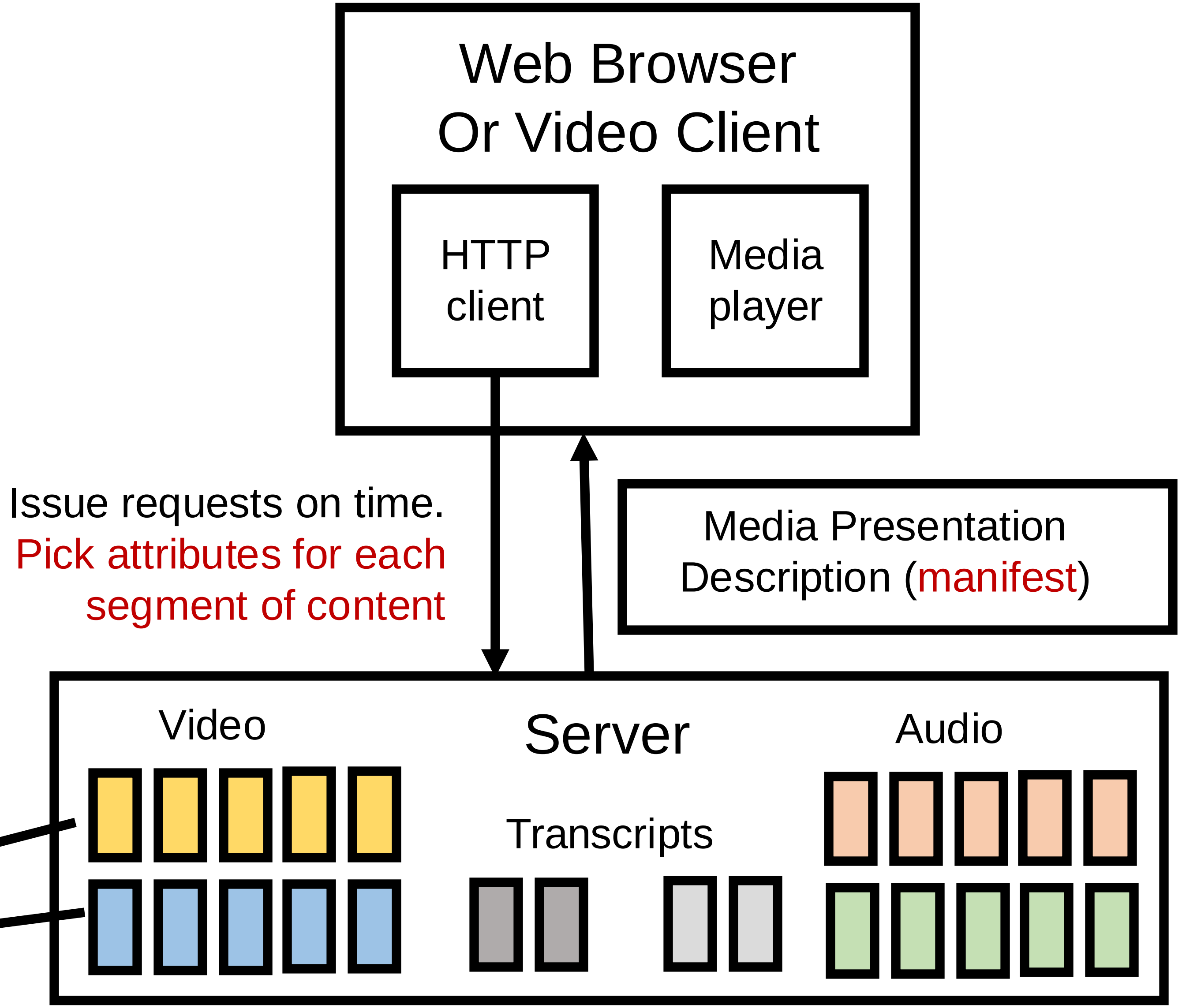
Streaming multimedia with DASH

- Dynamic Adaptive Streaming over HTTP
  - Used by Netflix and most popular video streaming services
- Adaptive: Perform video bit rate adaptation
  - It can be done on the client or the server (with client feedback)
- Dynamic: Retrieve a single video from multiple sources
- The DASH video server is just a standard HTTP server
  - Provides video/audio content in multiple formats and encodings
- Leverage existing web infrastructure
  - CDN
  - DNS

DASH: Key ideas

- Content (video, audio, transcript, etc.) divided into segments (time)
- Algorithms to determine and request varying attributes (e.g., bitrate, language) for each segment
- Goal: ensure good quality of service, match user prefs, etc.





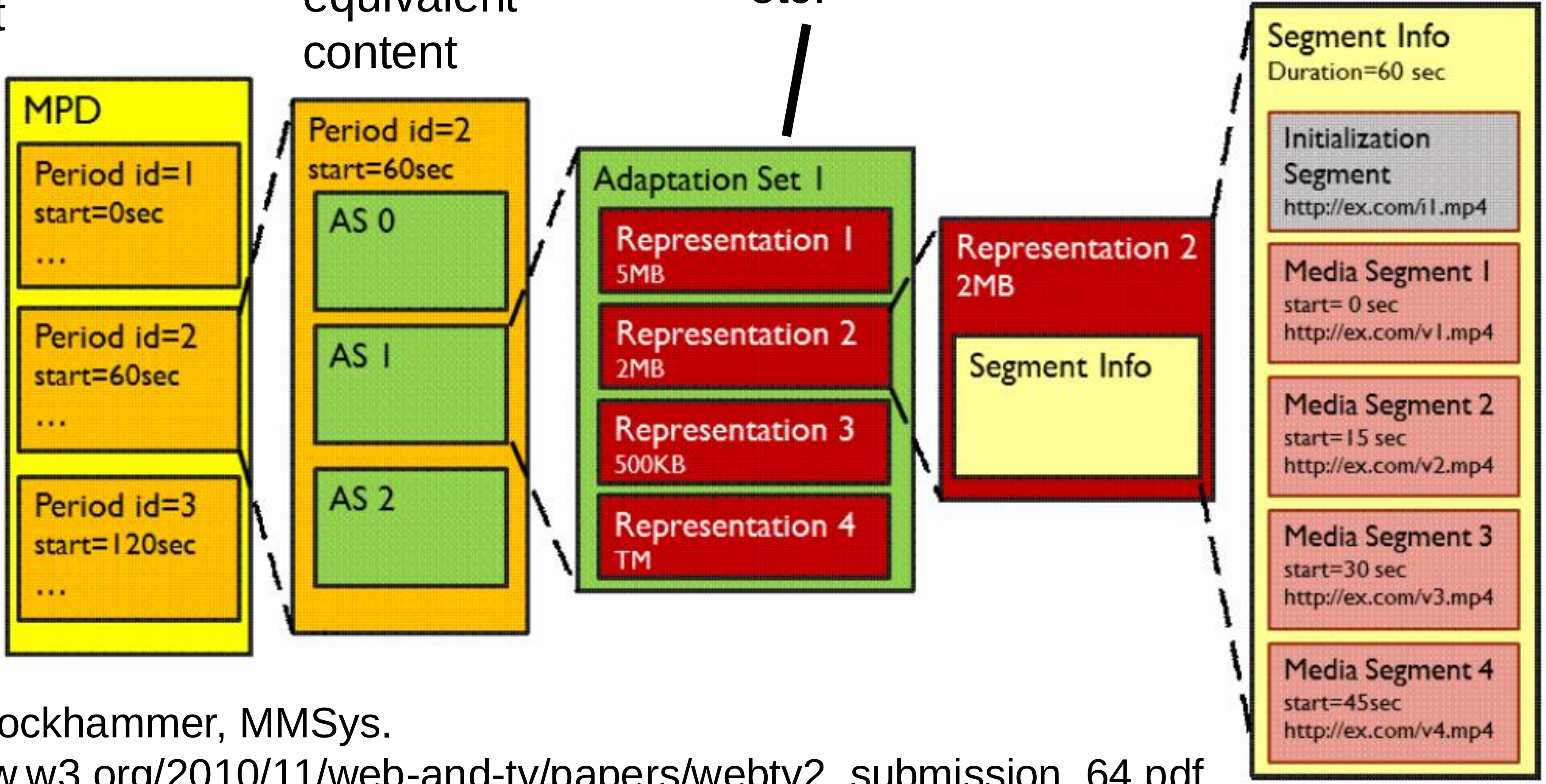
What does the manifest contain?

Periods: Durations of content

Adaptation set: functionally equivalent content

Representations: codecs, bit rates, etc.

Functionally equivalent: RSes of given AS
Functionally different: different ASes



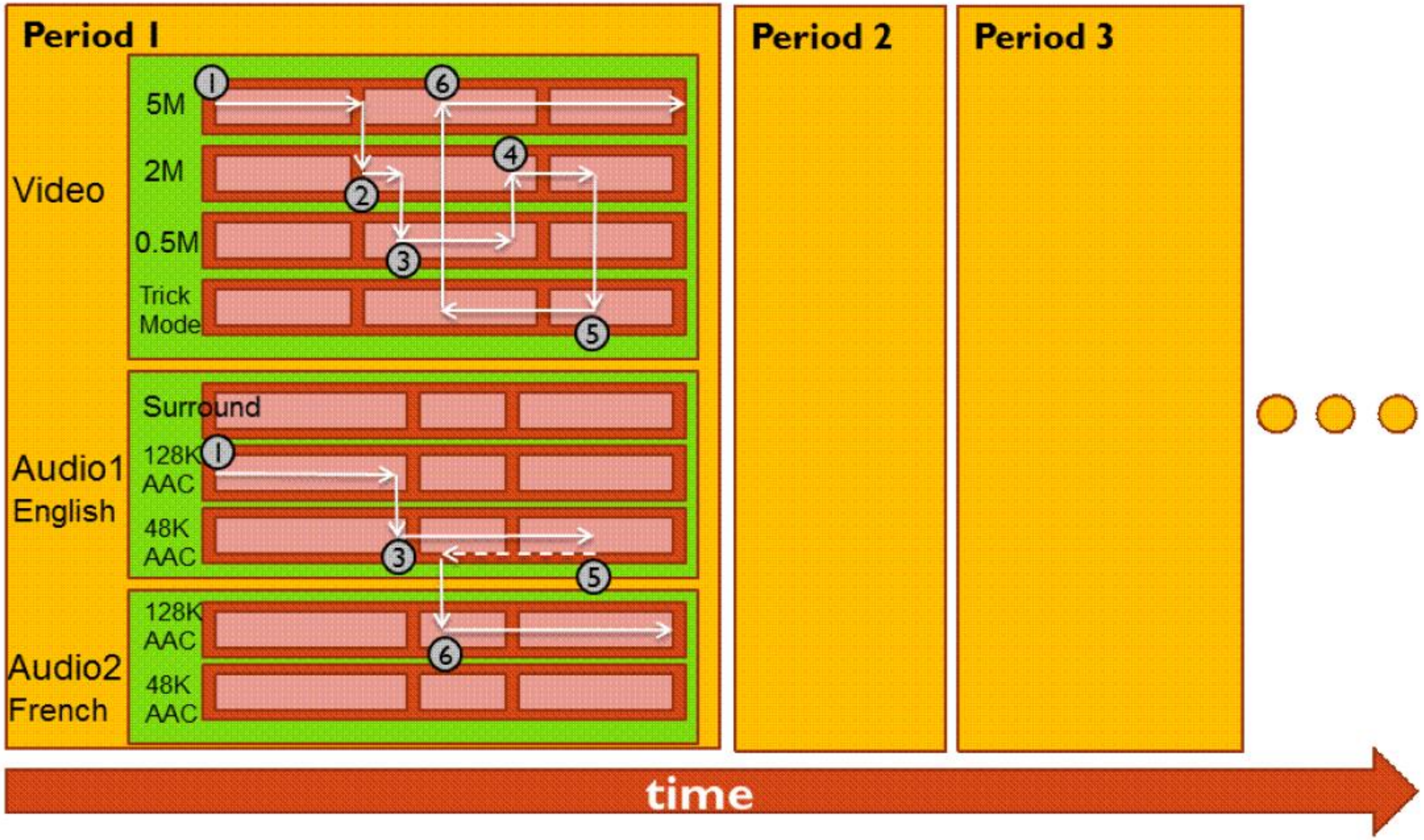
Multiple segments per representation

URL available for each segment

Byte ranges per segment (HTTP header for a range request)

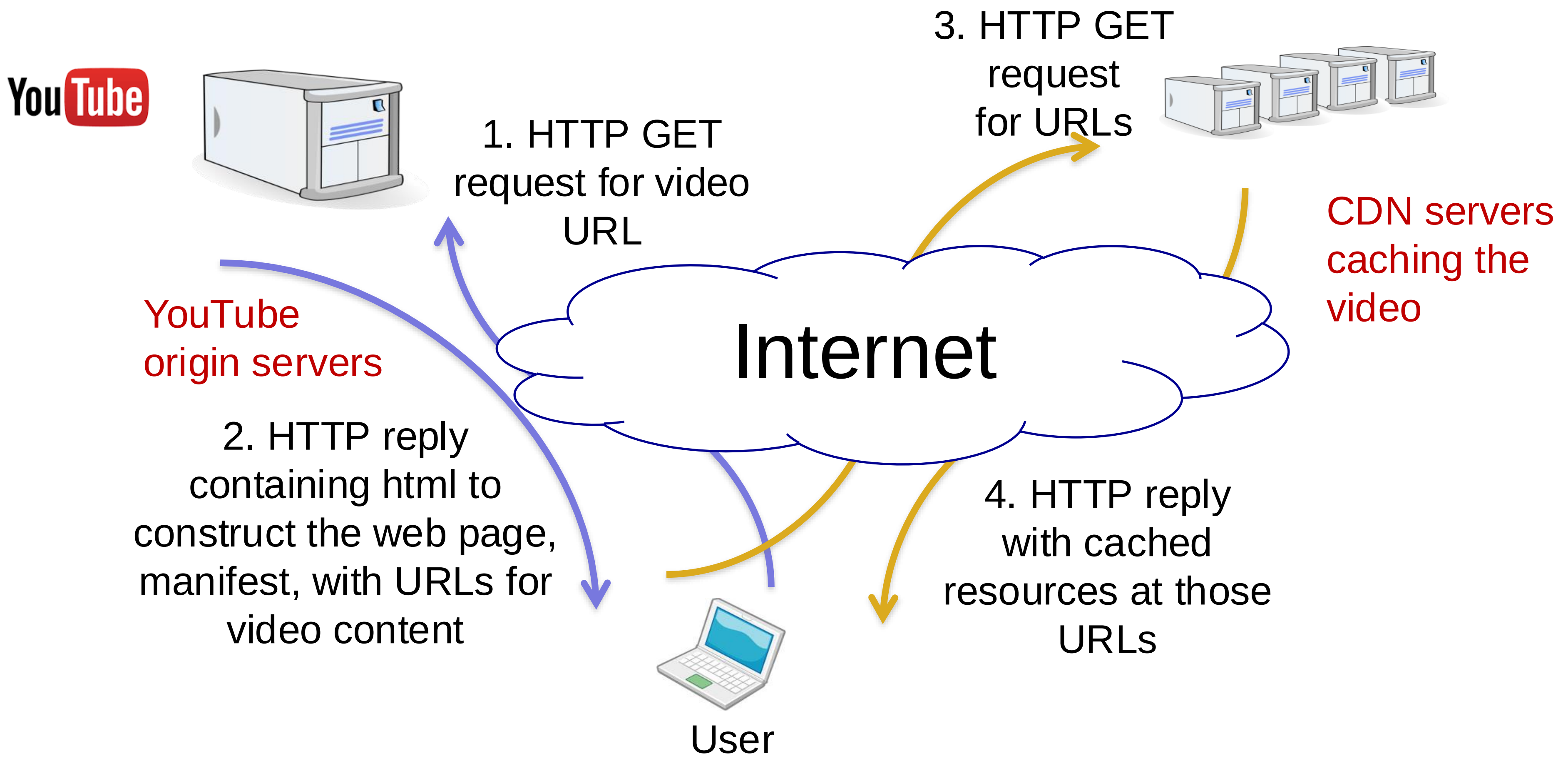
Source: Stockhammer, MMSys.
https://www.w3.org/2010/11/web-and-tv/papers/webtv2_submission_64.pdf

Dynamic changes in stream quality



Dynamic changes in stream location

- Just an HTTP request for an HTTP object



CDN DNS points user to best CDN server

DASH reference player

- https://reference.dashif.org/dash.js/latest/samples/dash-if-reference-player/index.html

DASH Summary

- Piggyback video on HTTP: widely used
- Enables independent HTTP requests per segment
  - Choose dynamic quality & preferences over time
  - Independent HTTP byte ranges
- Works well with CDNs
  - Fetch segments from locations other than the origin server
  - Fetch different segments from possibly different locations
- More resources on DASH
  - https://www.w3.org/2010/11/web-and-tv/papers/webtv2_submission_64.pdf
  - https://www.youtube.com/watch?v=xgowGnH5kUE

Application Layer: Wrap-up

- Name resolution, the web, video
- Protocols built over the `socket()` abstraction
- Simple designs go a long way
  - Plain text protocols, header-based evolution
- Infrastructure for functionality, performance, …
  - CDNs, web proxies
- Fit your apps to run on browsers: run almost anywhere (e.g. video)
- Apps are ultimately what users and most engineers care about
- BUT: if you don’t understand what’s under the hood, you risk bad design and poor performance in Internet applications