The Application Layer: Video Streaming

Lecture 8

http://www.cs.rutgers.edu/~sn624/352-S22

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Quick recap of concepts

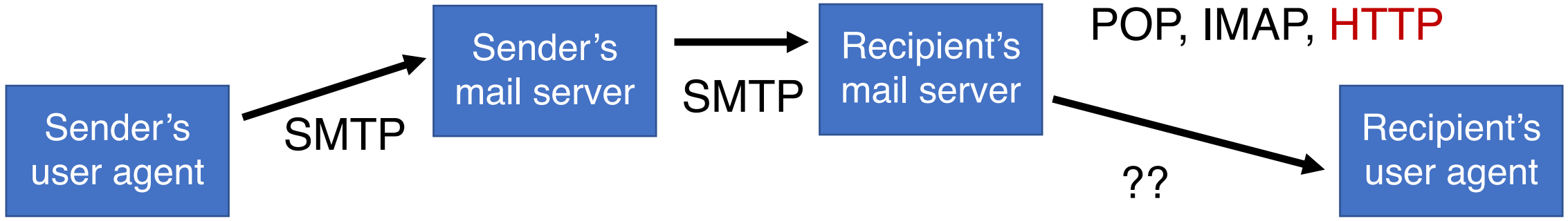
Video representation:
Pixels in Frames

Spatial coding

Temporal coding

Codec

Simple Mail Transfer Protocol (SMTP)



Mail access protocols
POP, IMAP, HTTP

Video codecs: terminology

- Video *bit rate*: effective number of bits per second of the video after encoding
- Higher bit rate == higher perceptual quality
- 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

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

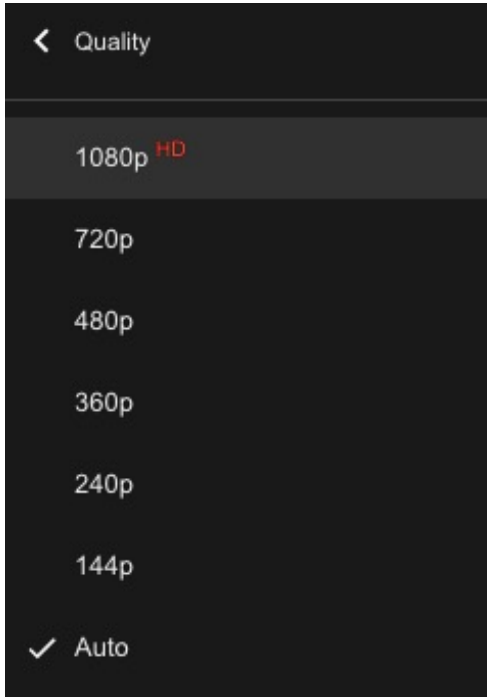
Networking multimedia: 3 types

- 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, Google Stadia
- Live streamed audio, video
  - e.g, sporting event on sky sports
  - Can buffer a little, but must be close to the “live edge” of content

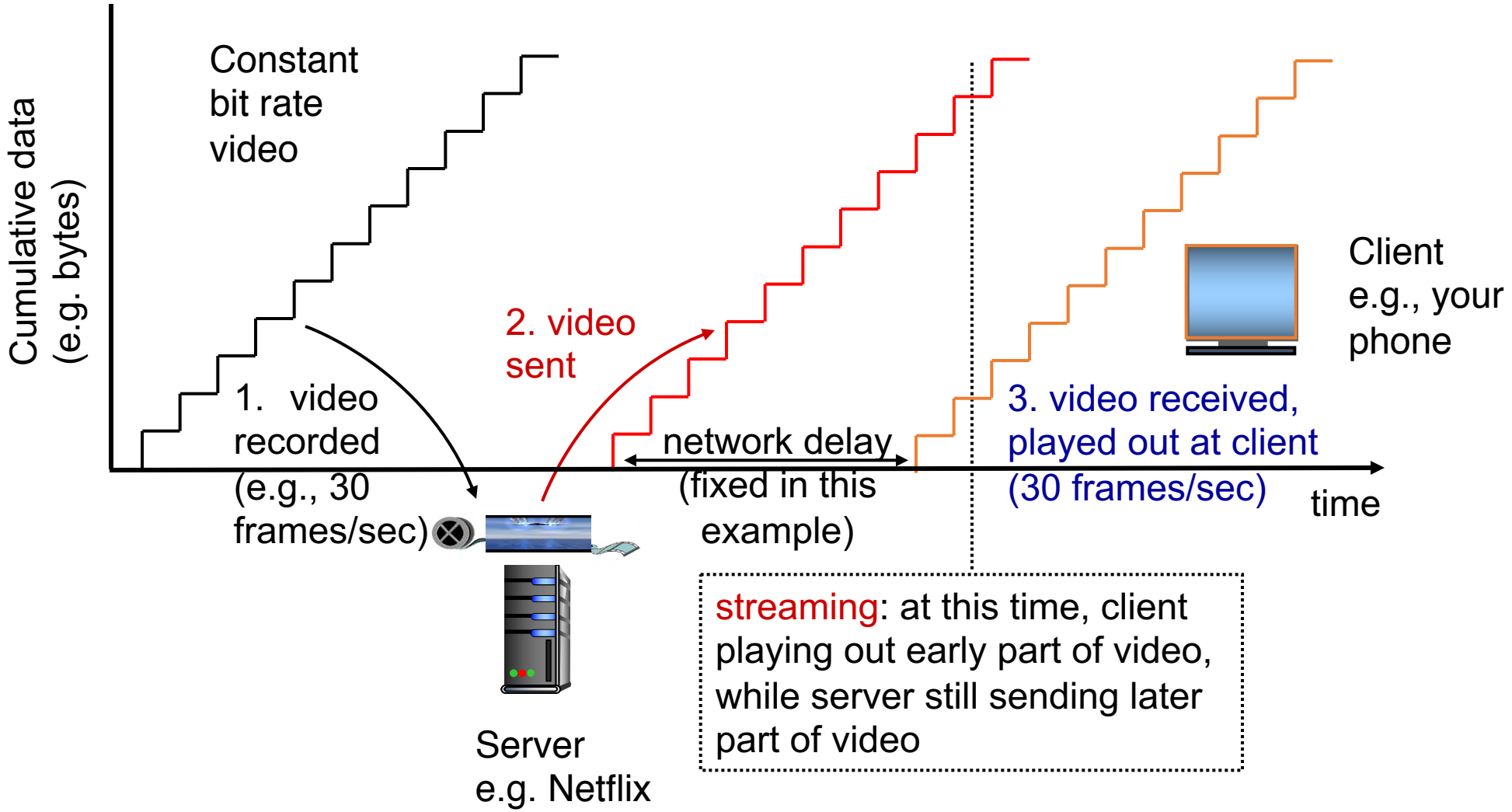
On-demand Video Streaming

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!
- Can change the quality of the content and where it's fetched mid-stream
  - More on this soon

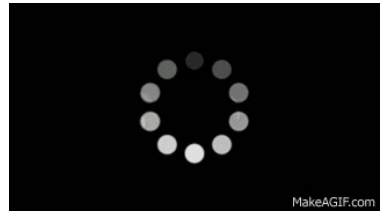


Streaming stored video



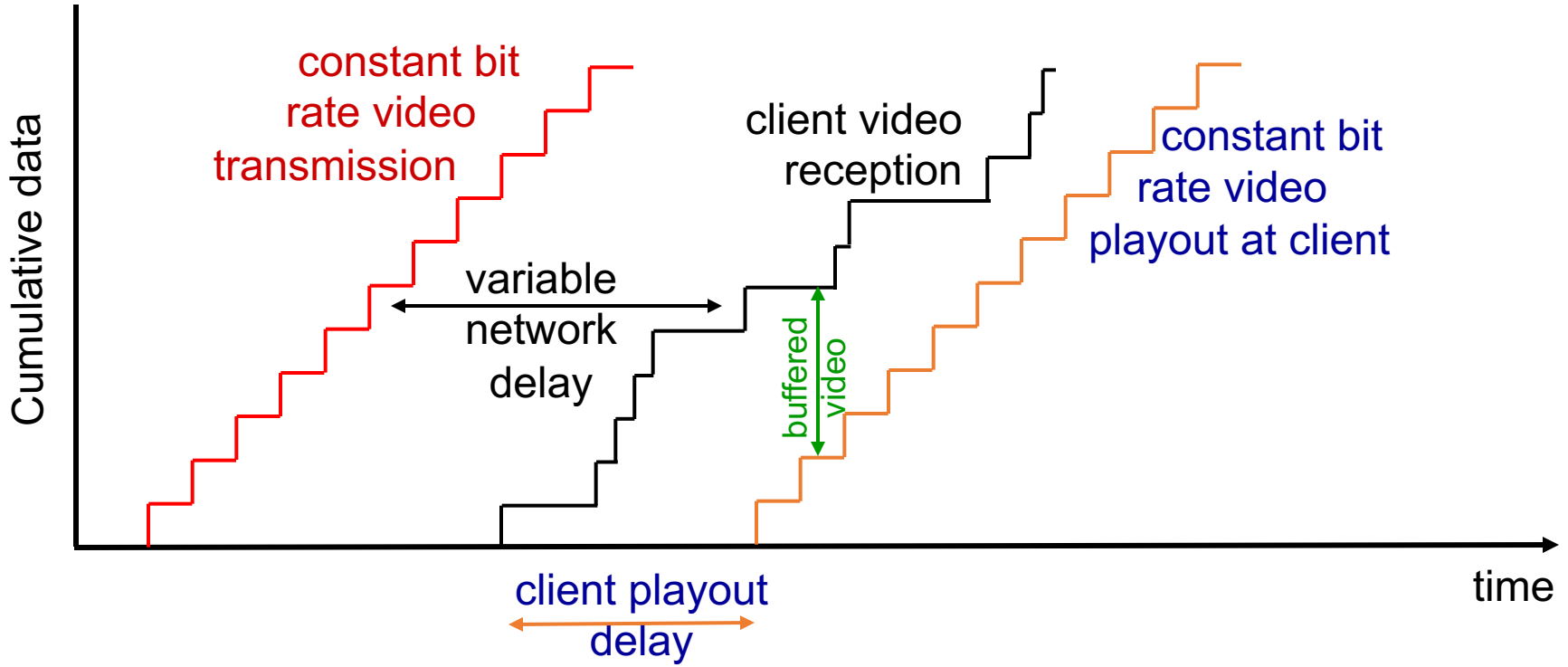
Streaming stored video: challenges

- Continuous playout constraint: once client playout begins, playback must match the original timing of the video (why?)
- But network delays are variable!



- Clients have a client-side buffer of downloaded video to absorb variation in network conditions
- Client interactivity: pause, fast-forward, rewind, jump through video

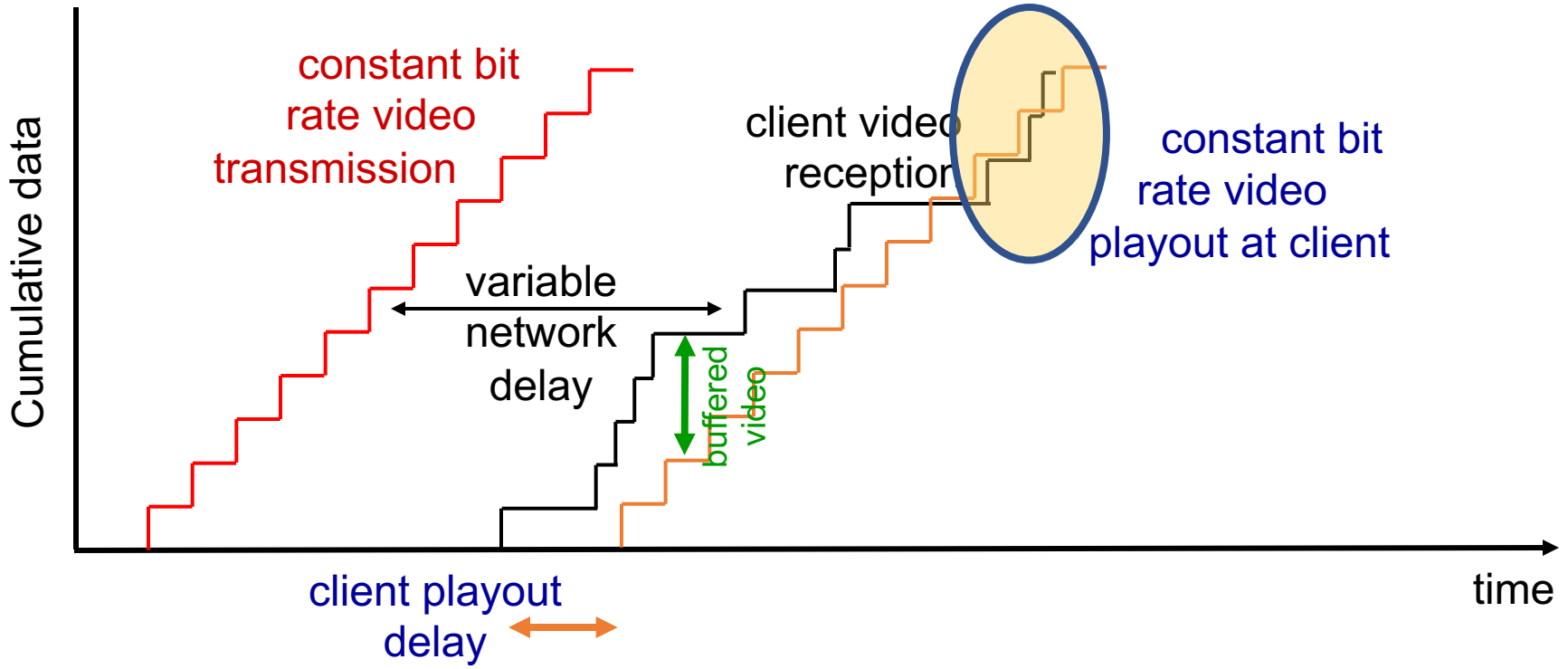
Scenario 1: Constant bit-rate video



Client-side buffering with playout delay:

compensate for network-added delays and variations in the delay

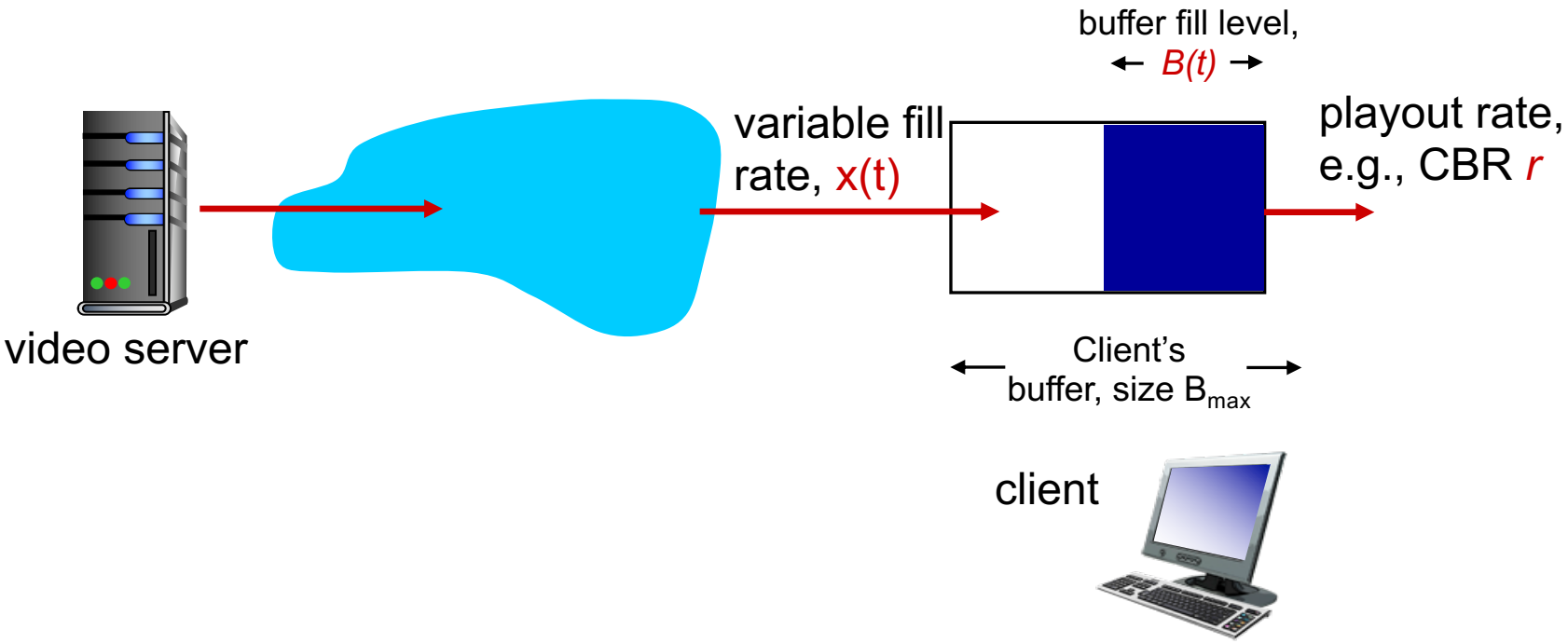
Scenario 2: Small playout delay



Playout delay that's too small can cause stalls

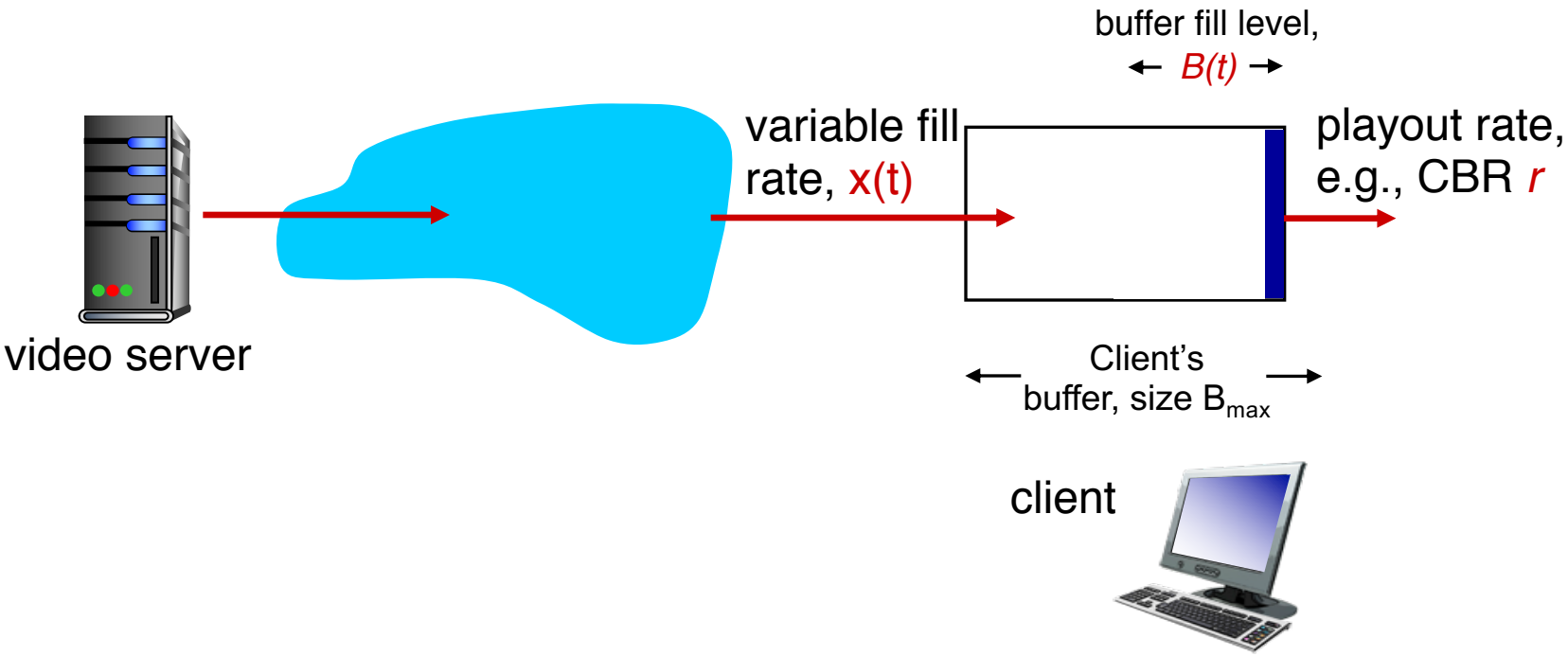
There's nothing in the buffer to show to the user

Client-side buffering, playout



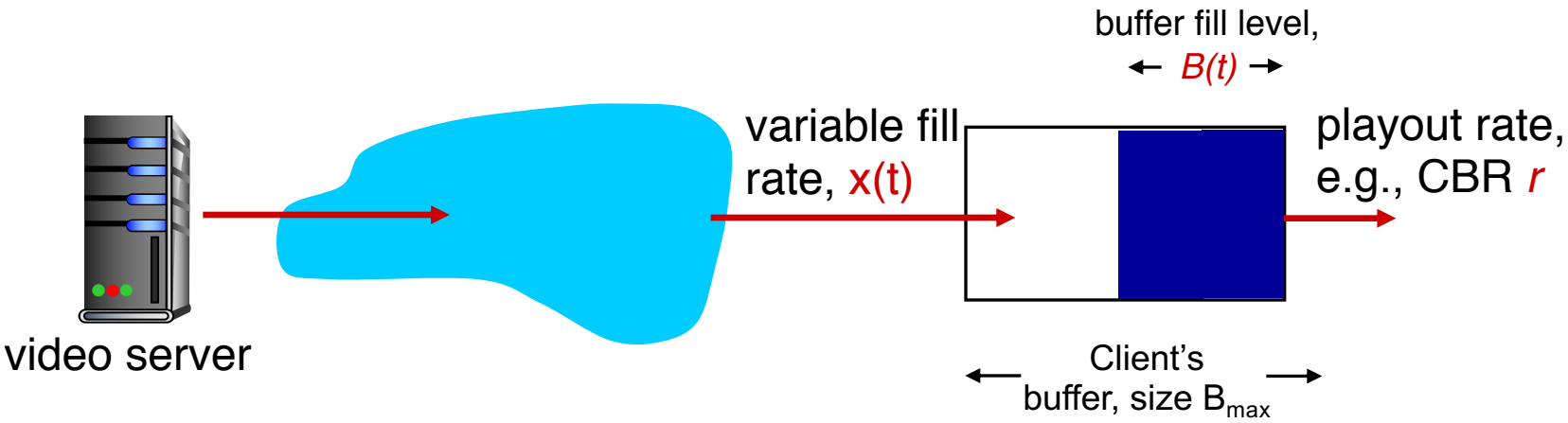
Most video is broken up in time into multiple segments
Client downloads video segment by segment
For example: a segment might be 4 seconds worth of video.

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)

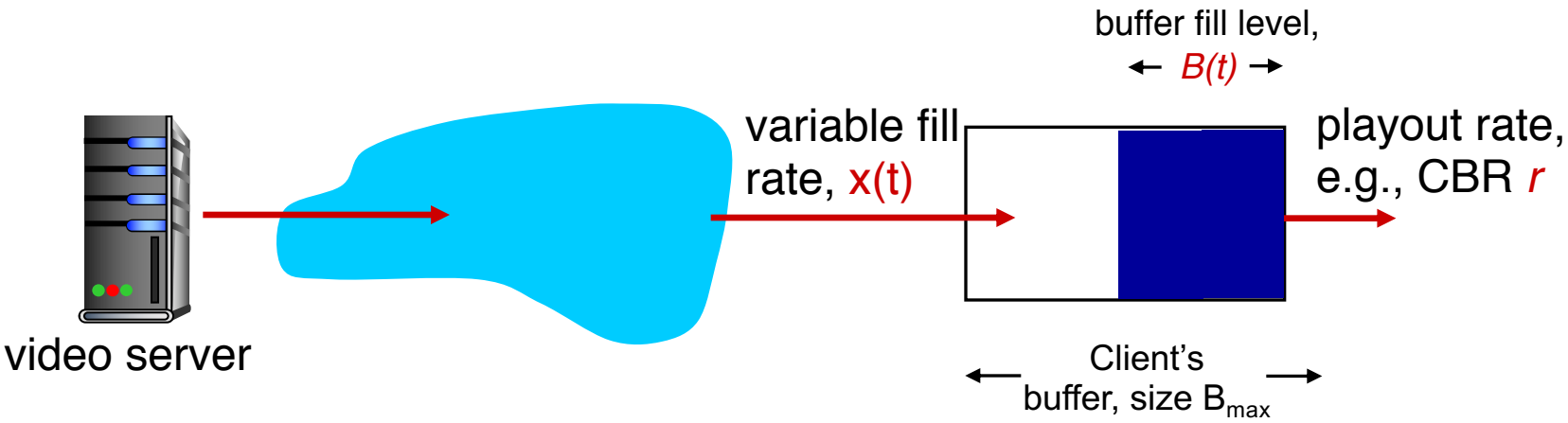
Client-side buffering, playout



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

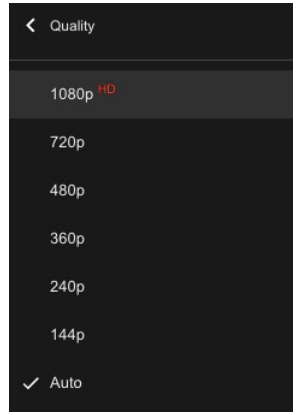


playout buffering: average fill rate ($\bar{x}$), playout rate (r):

- is $\overline{x} < r$ or $\overline{x} > r$ for a given network connection?
- It is hard to predict this in general!
  - Best effort network suffers long queues, paths with low bandwidth, …
- How to set 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



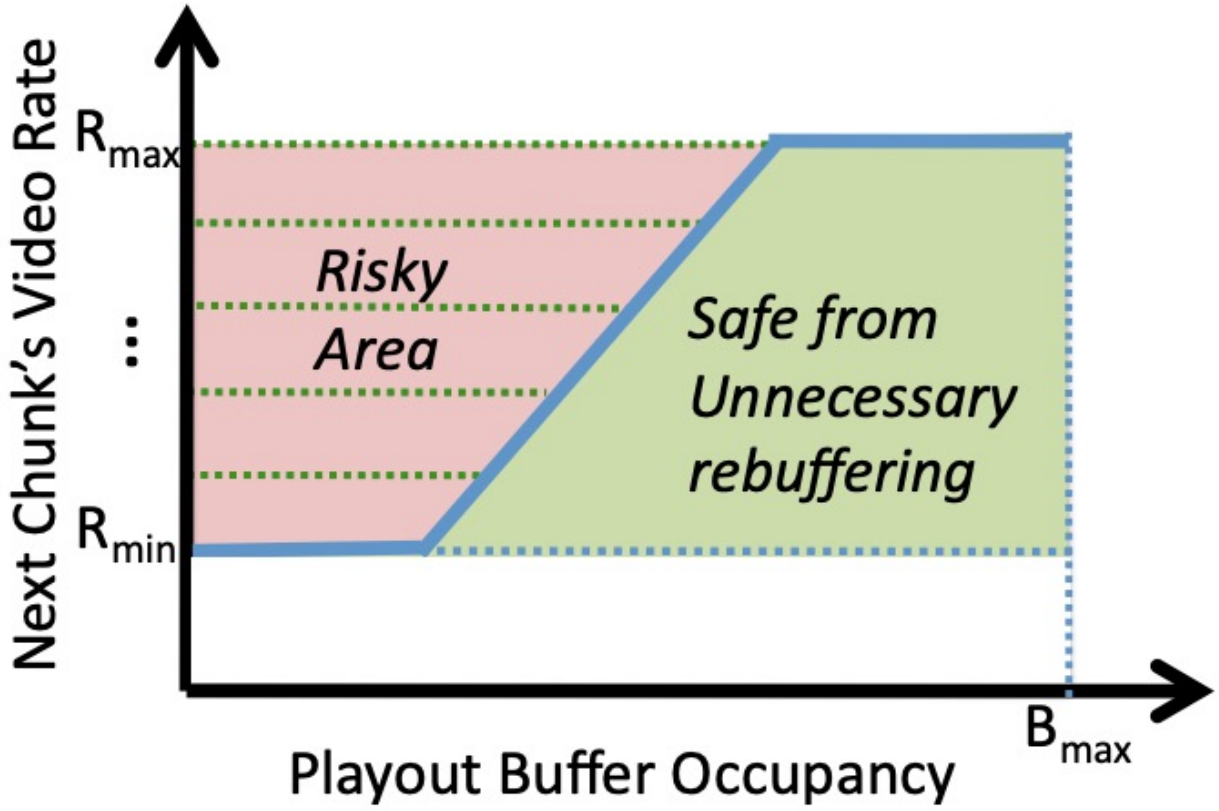
- Adapt bit rate per segment through collaboration between the video client (e.g., your browser) and the server (e.g., @ Netflix)
- Adaptive bit-rate (ABR) video: change the bit-rate (quality) of next video segment based on network and client conditions
- A typical strategy: Buffer-based rate adaptation

Buffer-based bit-rate adaptation

- Key idea: If there is a large stored buffer of video, optimize aggressively 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-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

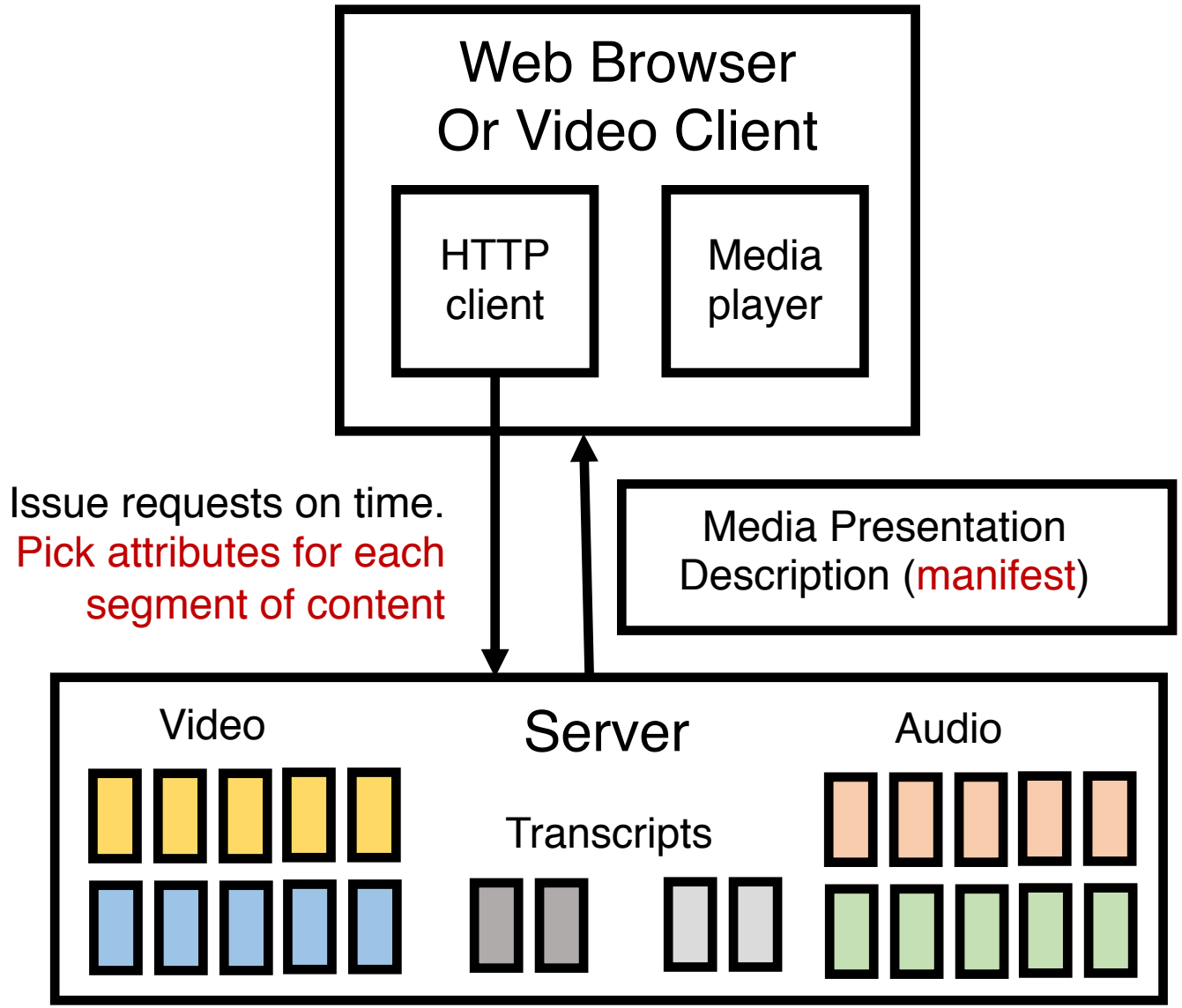
Dynamic Adaptive Streaming over HTTP (DASH)

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-based infrastructure
  - DNS
  - CDNs!

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.

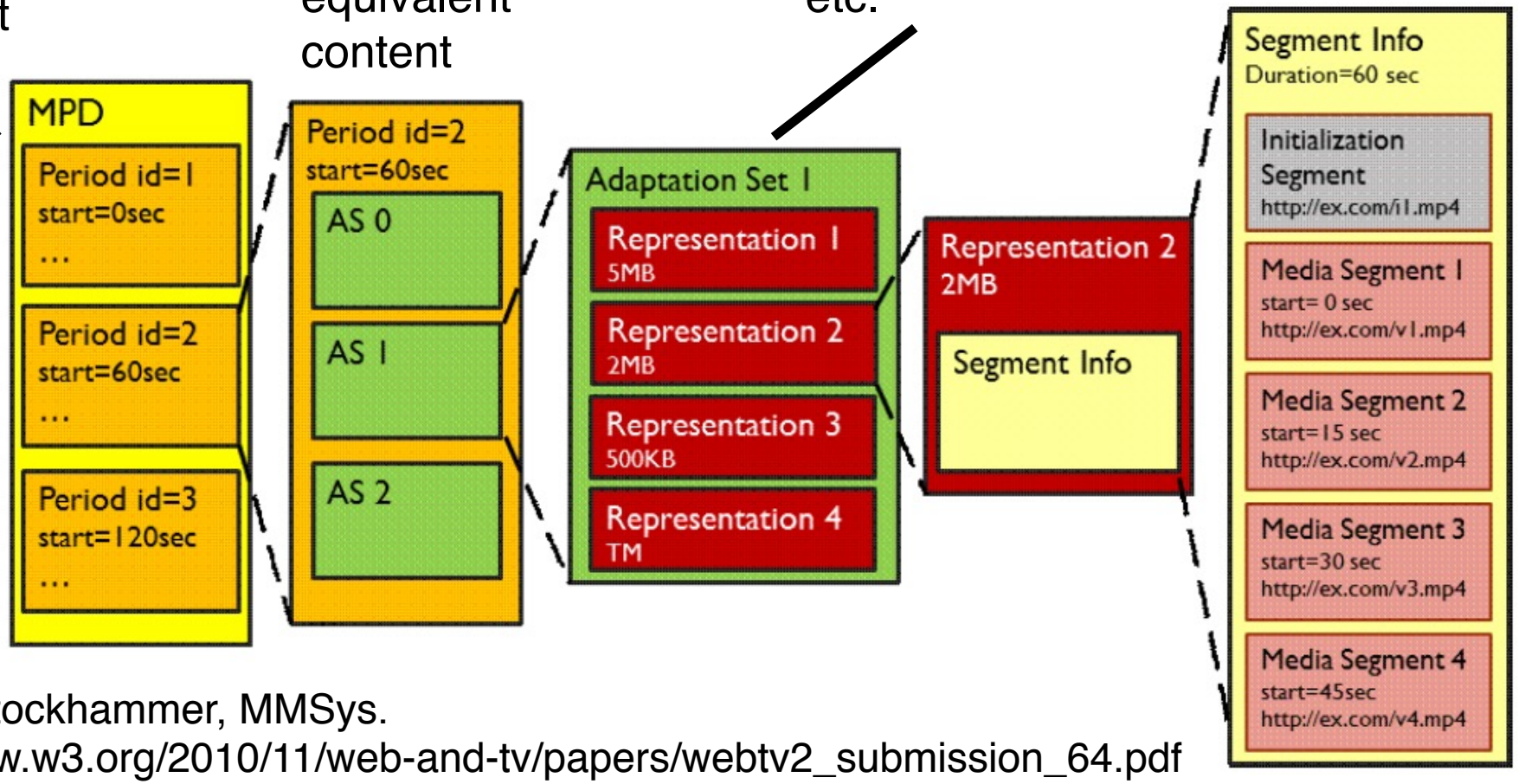


What does the manifest contain?

Periods: Durations of content

Adaptation set: functionally equivalent content

Representations: codecs, bit rates, etc.



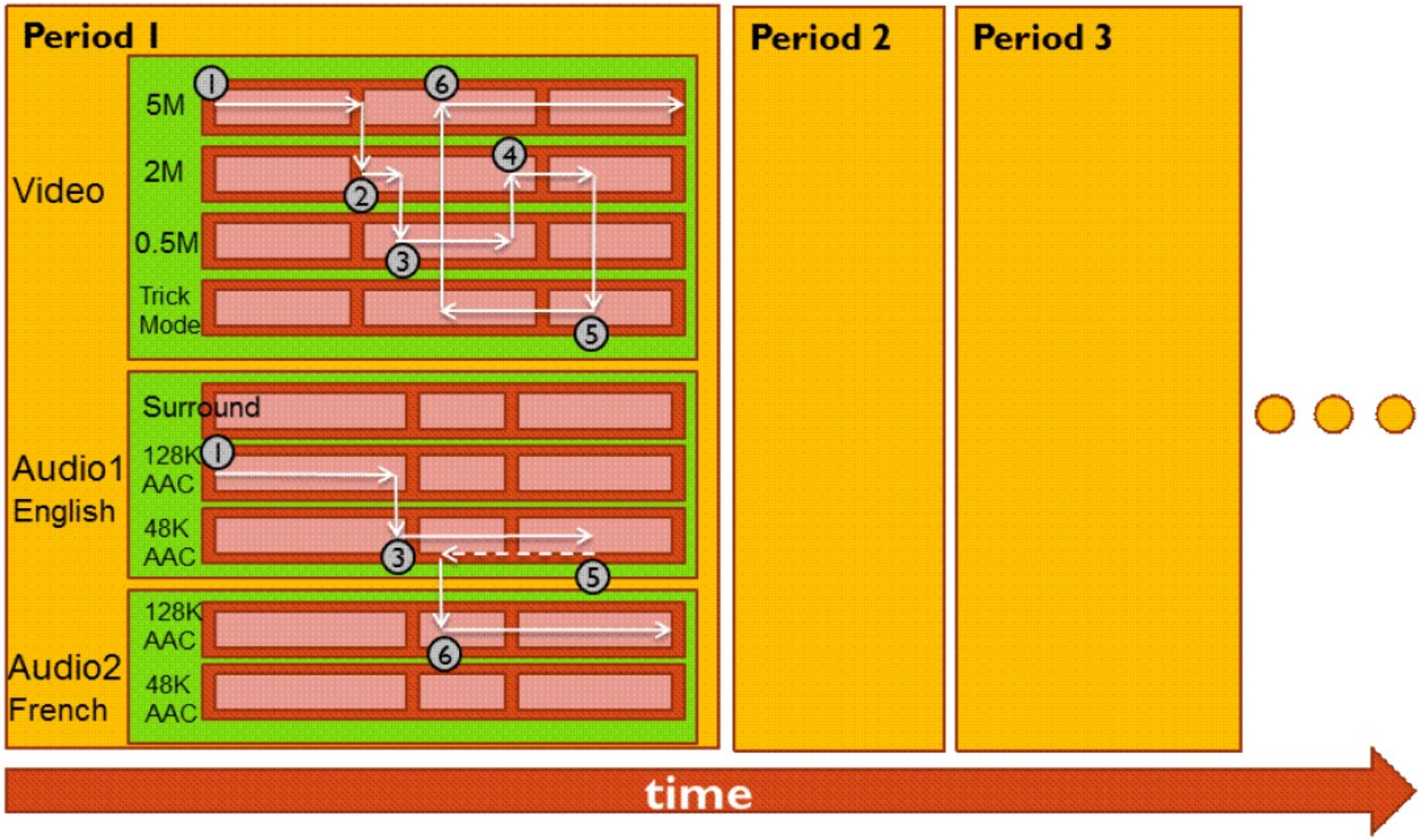
Multiple segments per representation

URL available for each segment

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

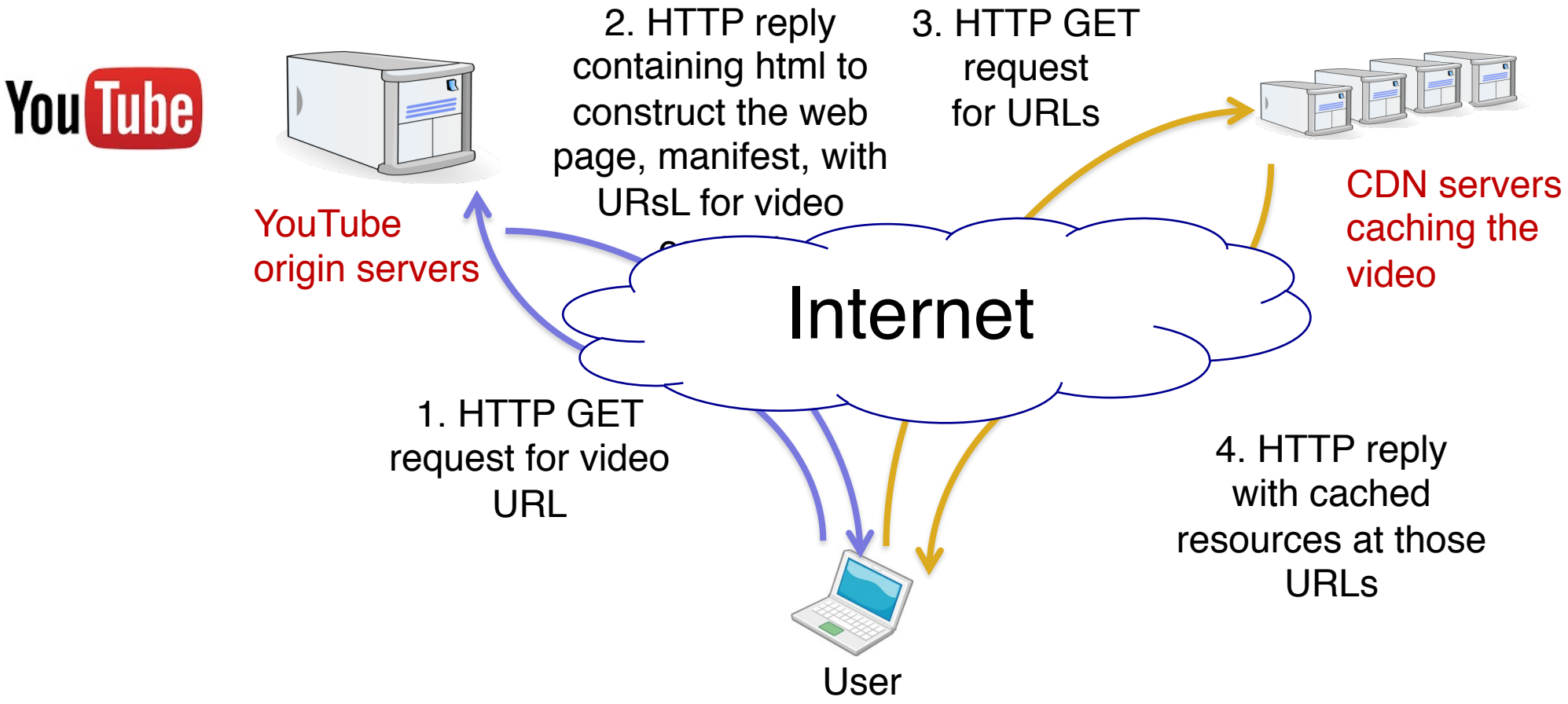
Source: Stockhammer, MMSys.
https://www.w3.org/2010/11/web-and-tv/papers/webtv2_submission_64.pdf

Dynamic changes in stream quality



Get stream from anywhere!

- Just an HTTP request for an HTTP object



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