# Video

Lecture 10

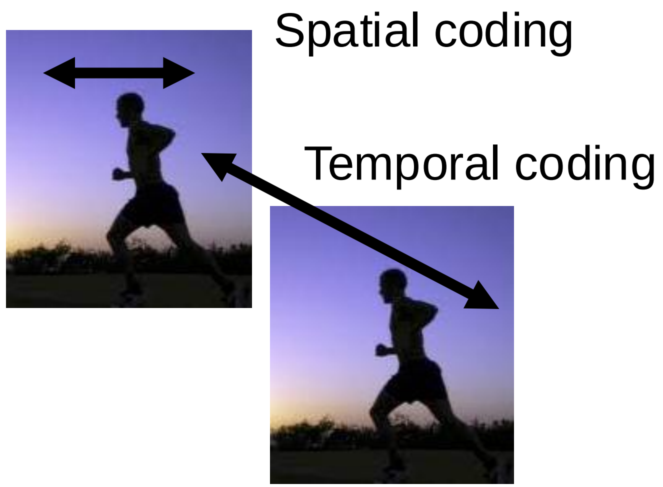
http://www.cs.rutgers.edu/~sn624/352-F24

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

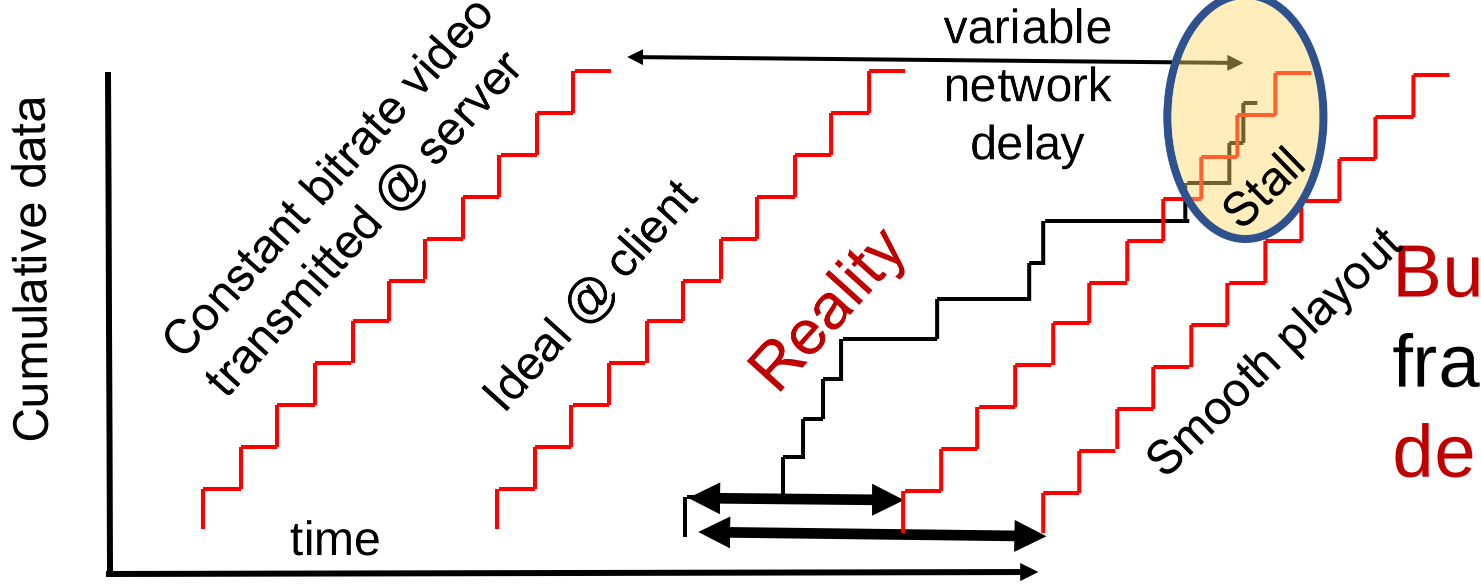
Multimedia

# Quick recap of concepts



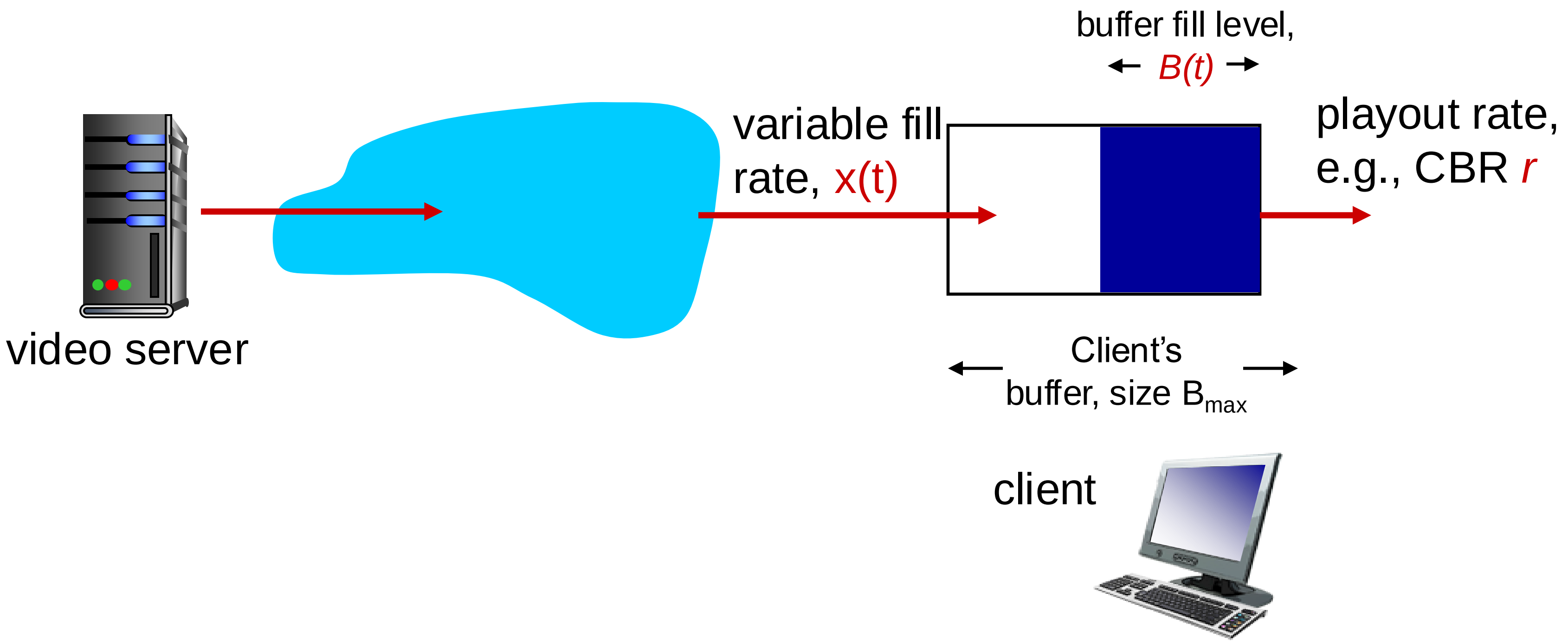
Video Bitrate

Bits played out per second (can vary over video's lifetime)



Buffer at the client to hold frames initially until playout delay $t_p$

# Client-side buffering, playout

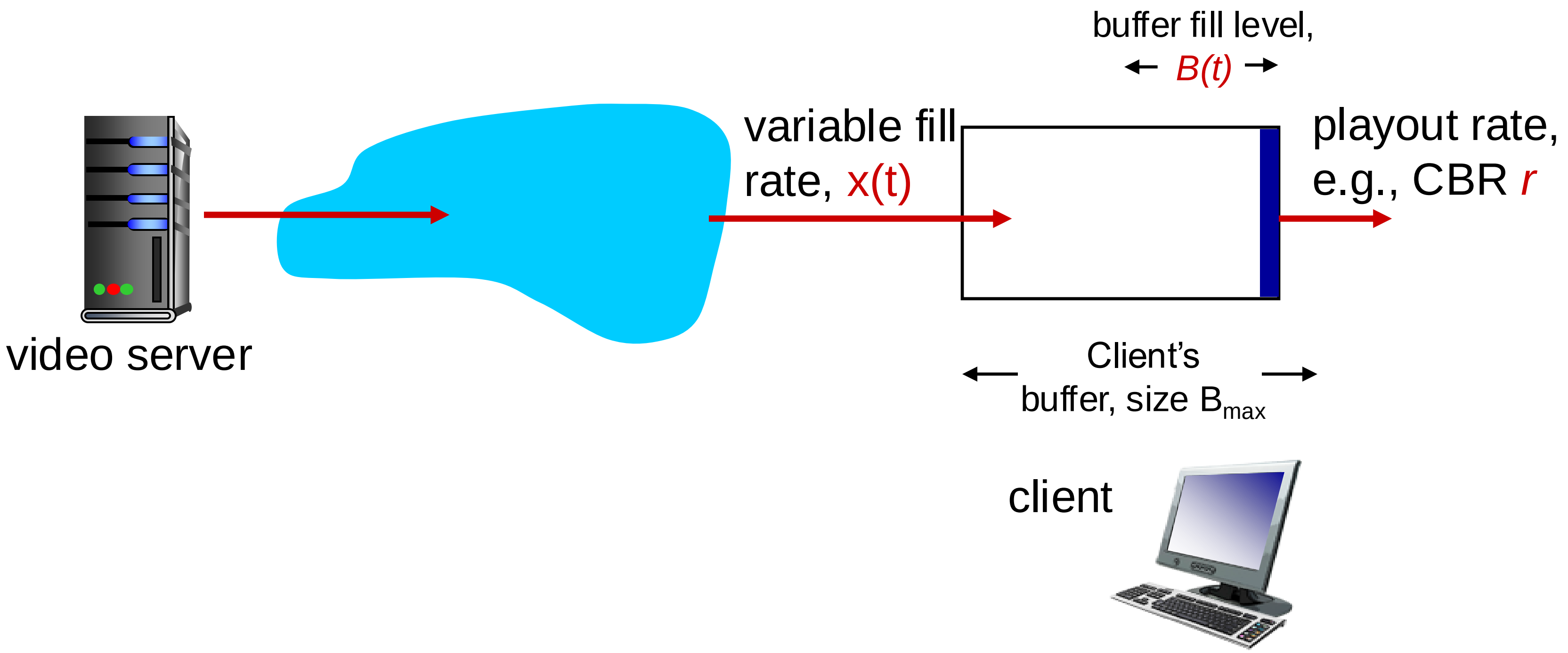

buffer fill level,
$B(t)$
variable fill rate, $x(t)$
playout rate, e.g., CBR $r$
video server
Client's buffer, size $B_{max}$
client

Most video is broken up in time into multiple segments

Client downloads video segment by segment

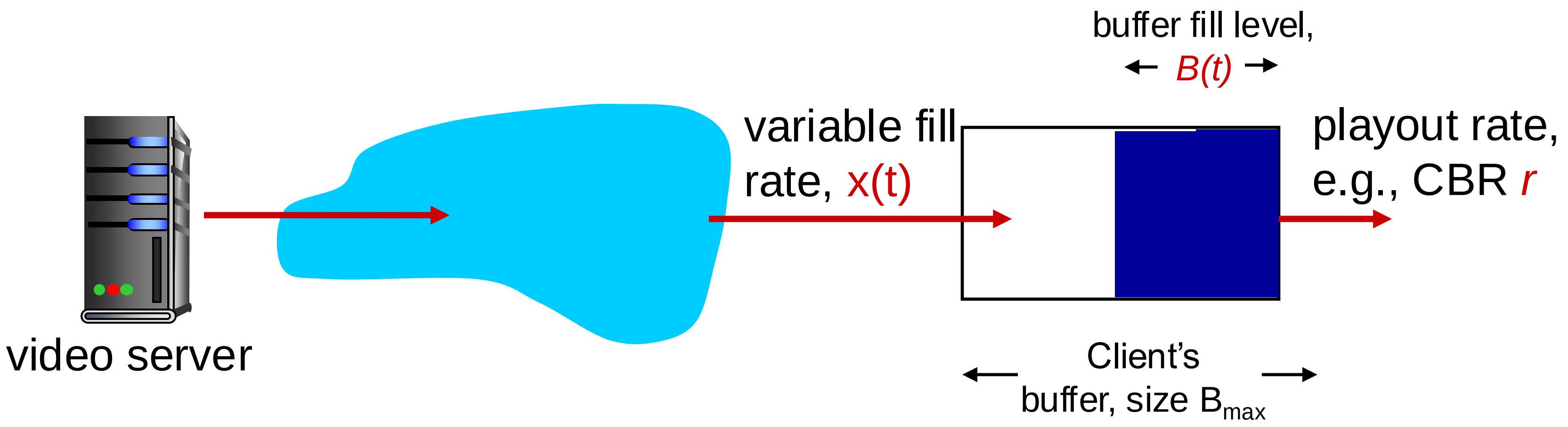
For example: a segment might be 4 seconds worth of video.

# 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)

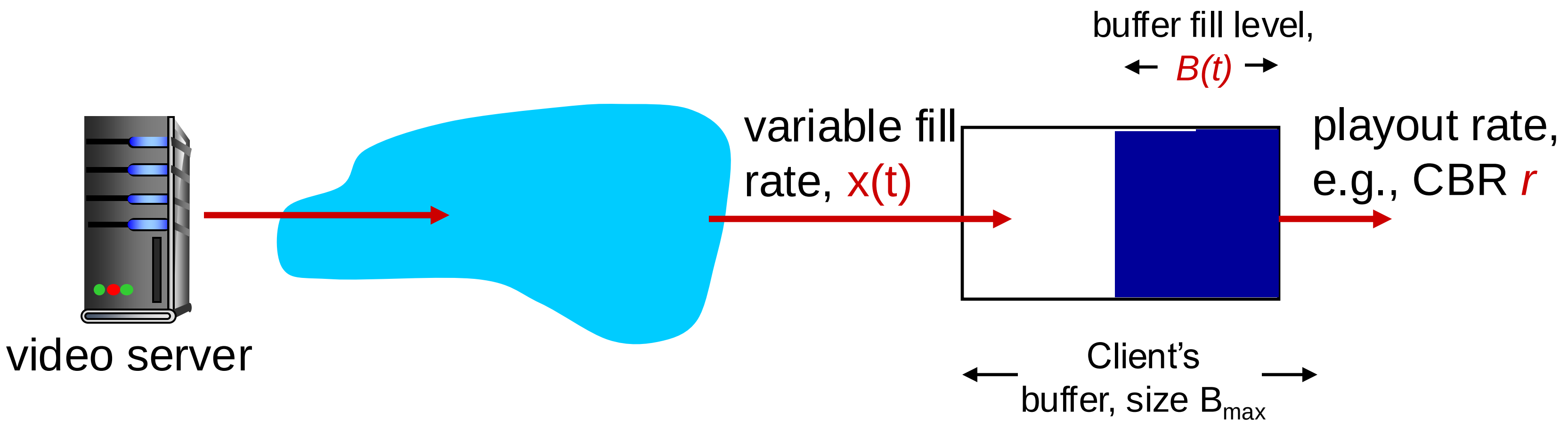
# Client-side buffering, playout



*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 for the user to begin watching

# Client-side buffering, playout

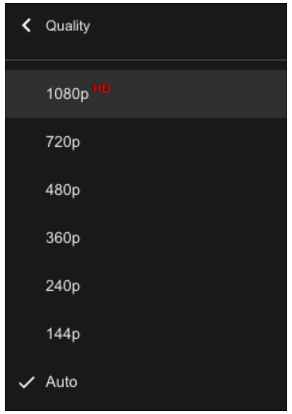


*playout buffering: average fill rate ($\bar{x}$), playout rate (r):*

- 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

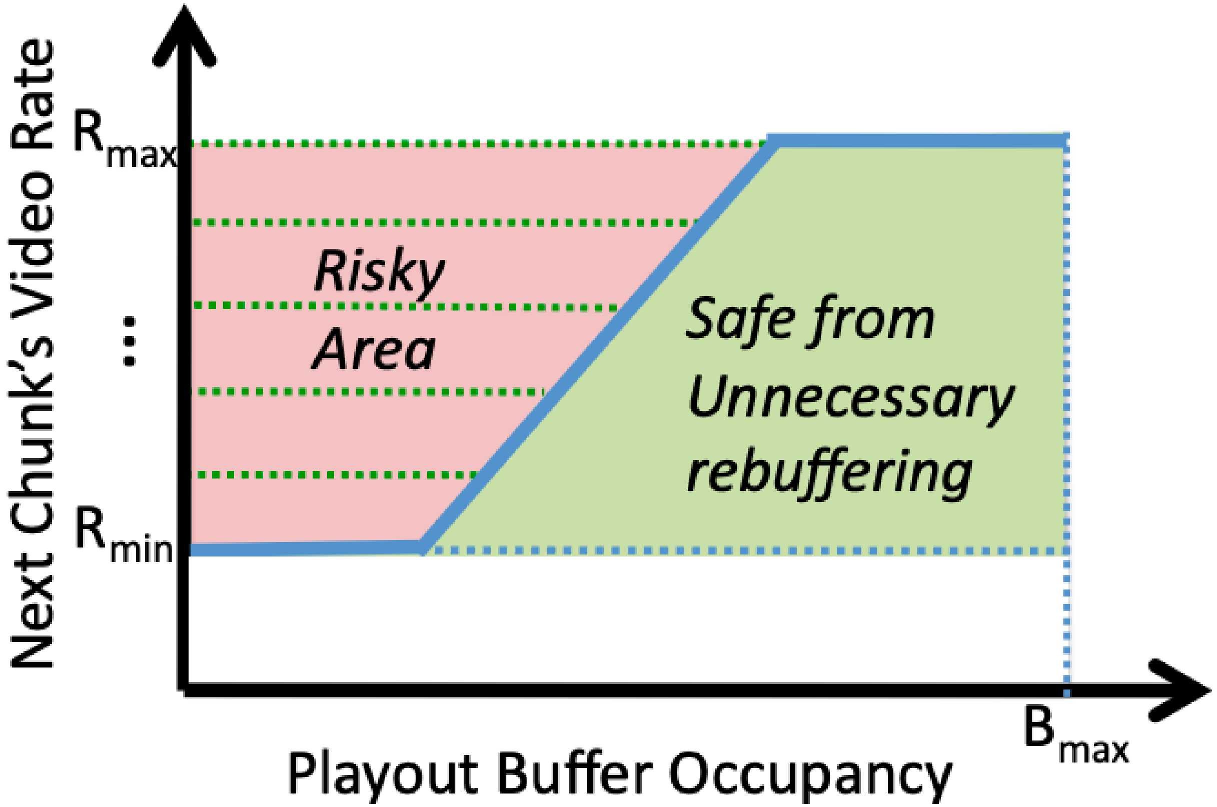
# Buffer-based bit-rate adaptation

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

- Else (i.e., client video buffer has low occupancy), avoid stalls by being conservative and asking for a lower quality (bit-rate)
  - The hope: the lower bandwidth requirement of a lower-quality stream is more easily met; stalls averted

- Buffer is measured in seconds of playout left before stalling

# Buffer-based bit-rate adaptation



Provide high video quality overall despite variable and intermittently poor network conditions.

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

A Buffer-Based Approach to Rate Adaptation

(used by Netflix)

# Dynamic Adaptive Streaming over HTTP (DASH)

# Streaming multimedia with HTTP

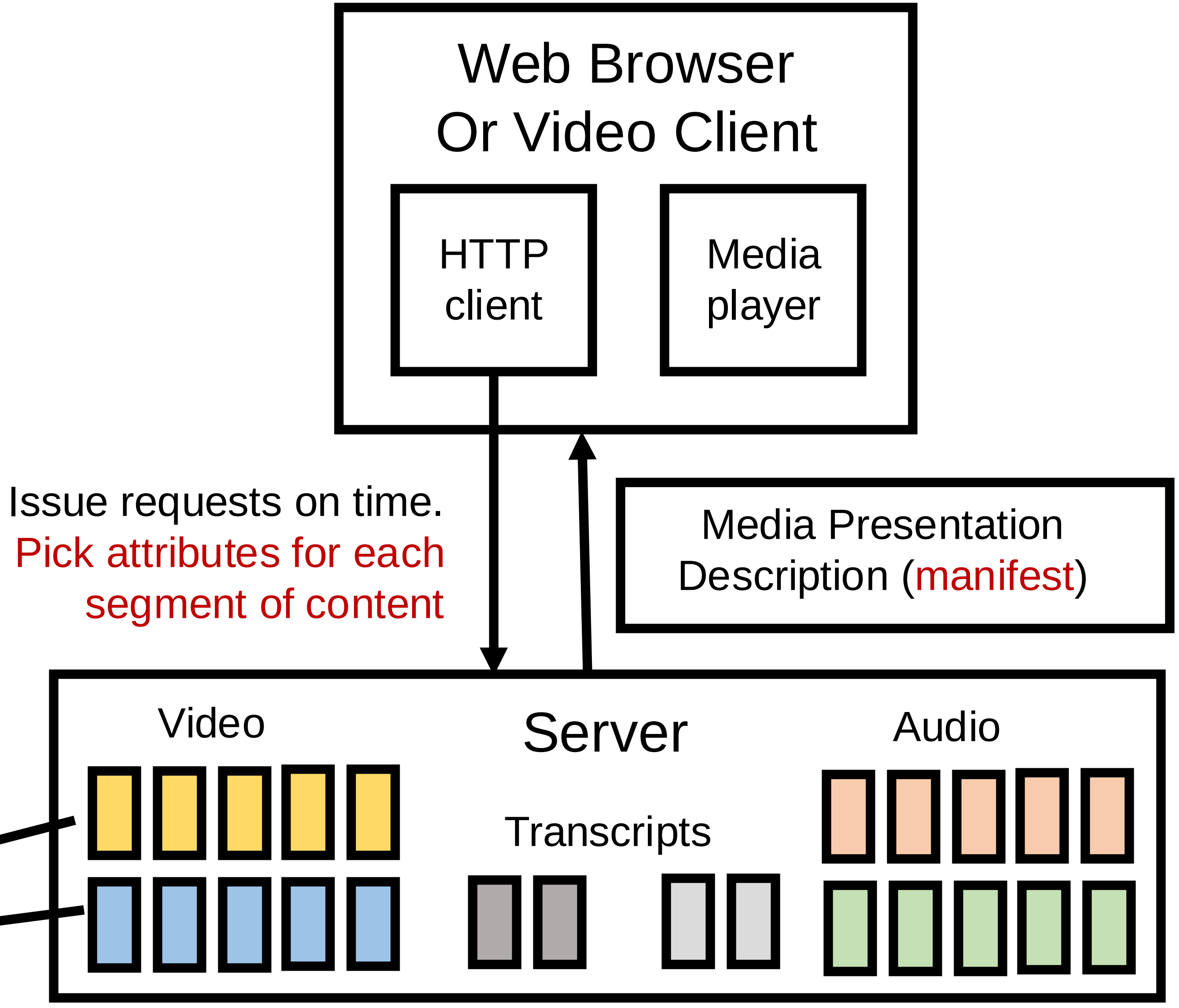
- Early video: basic UDP
  - Problems: reliability, blocking

- Today: repurpose web infrastructure and protocols for video

- DASH: Dynamic Adaptive Streaming over HTTP
  - Used by Netflix, YouTube, and other video streaming services

# 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.



Web Browser
Or Video Client

HTTP client

Media player

Issue requests on time.
Pick attributes for each segment of content

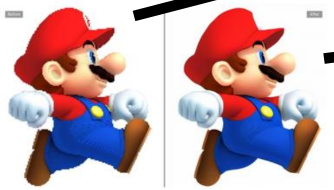
Media Presentation Description (manifest)

Server

Video

Transcripts

Audio



# Streaming multimedia with HTTP

- Dividing up a video into multiple segments enables a few things
- Possible to decide the quality per chunk
  - Enables bit rate adaptation
  - Typically done on the client, but possible on the server (with feedback)
  - Change language, receive sub-titles, etc. mid-stream
- Retrieve segments of a single video from multiple sources
- DASH video server is just a standard HTTP server
  - Client issue HTTP GET requests (typically with the Range request header)
  - Leverage existing web infrastructure (CDN, DNS)
- Send different clients to different video sources (CDN)

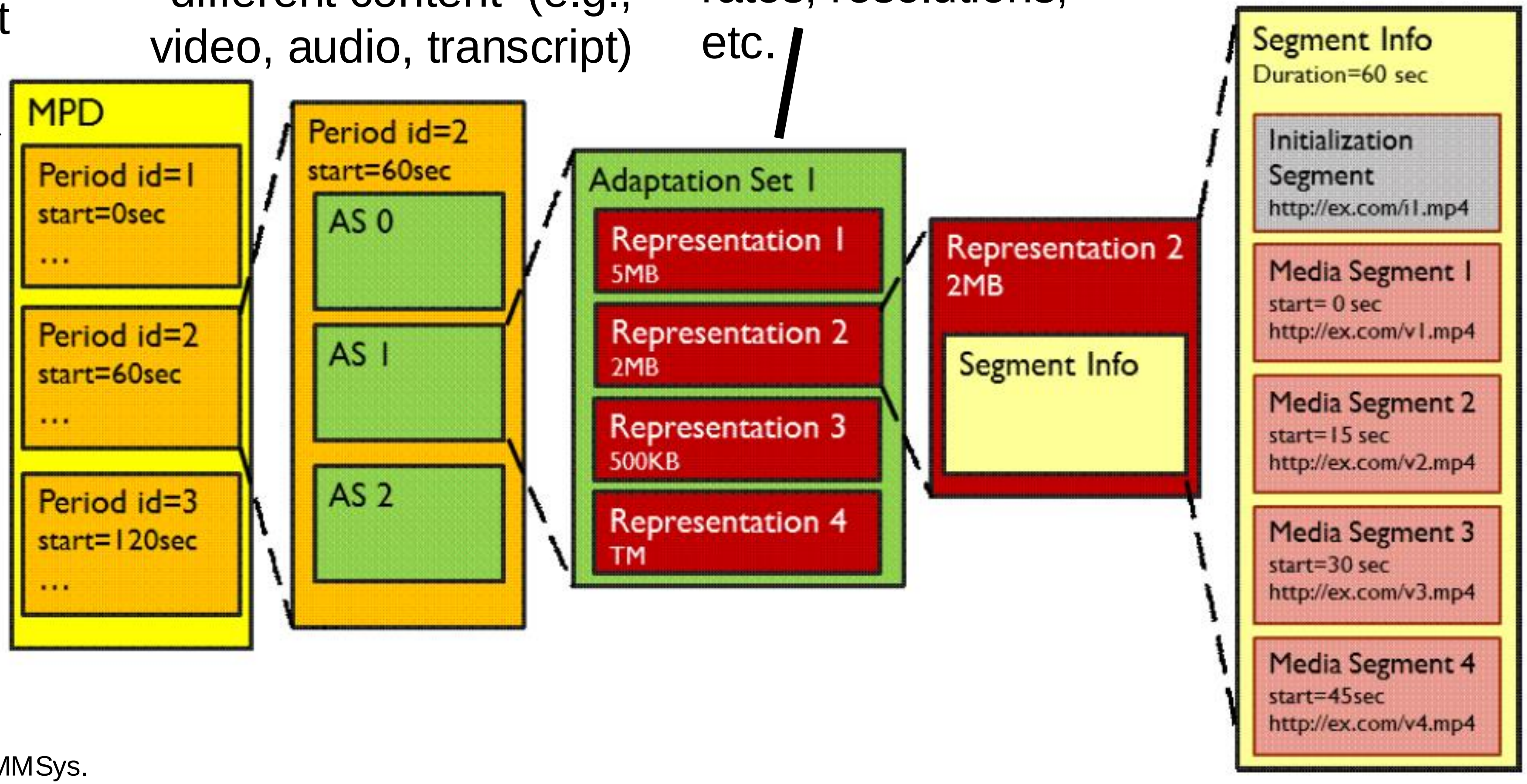
# What does the manifest contain?

Periods: Durations of content

Adaptation sets: each AS includes functionally different content (e.g., video, audio, transcript)

Representation sets (RS): codecs, bit rates, resolutions, etc.

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



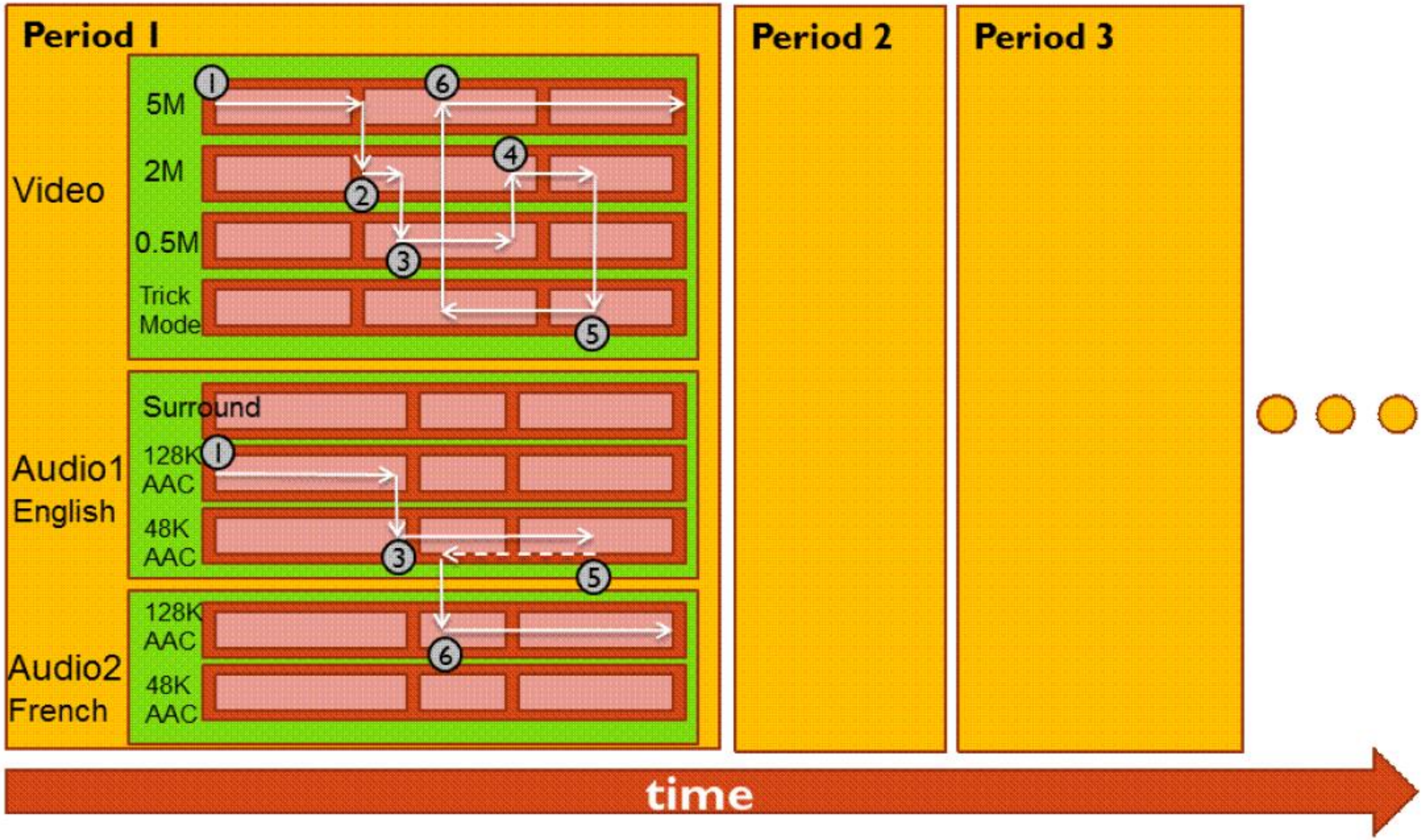
Multiple segments per representation

URL for each RS/segment

If URL per RS, 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

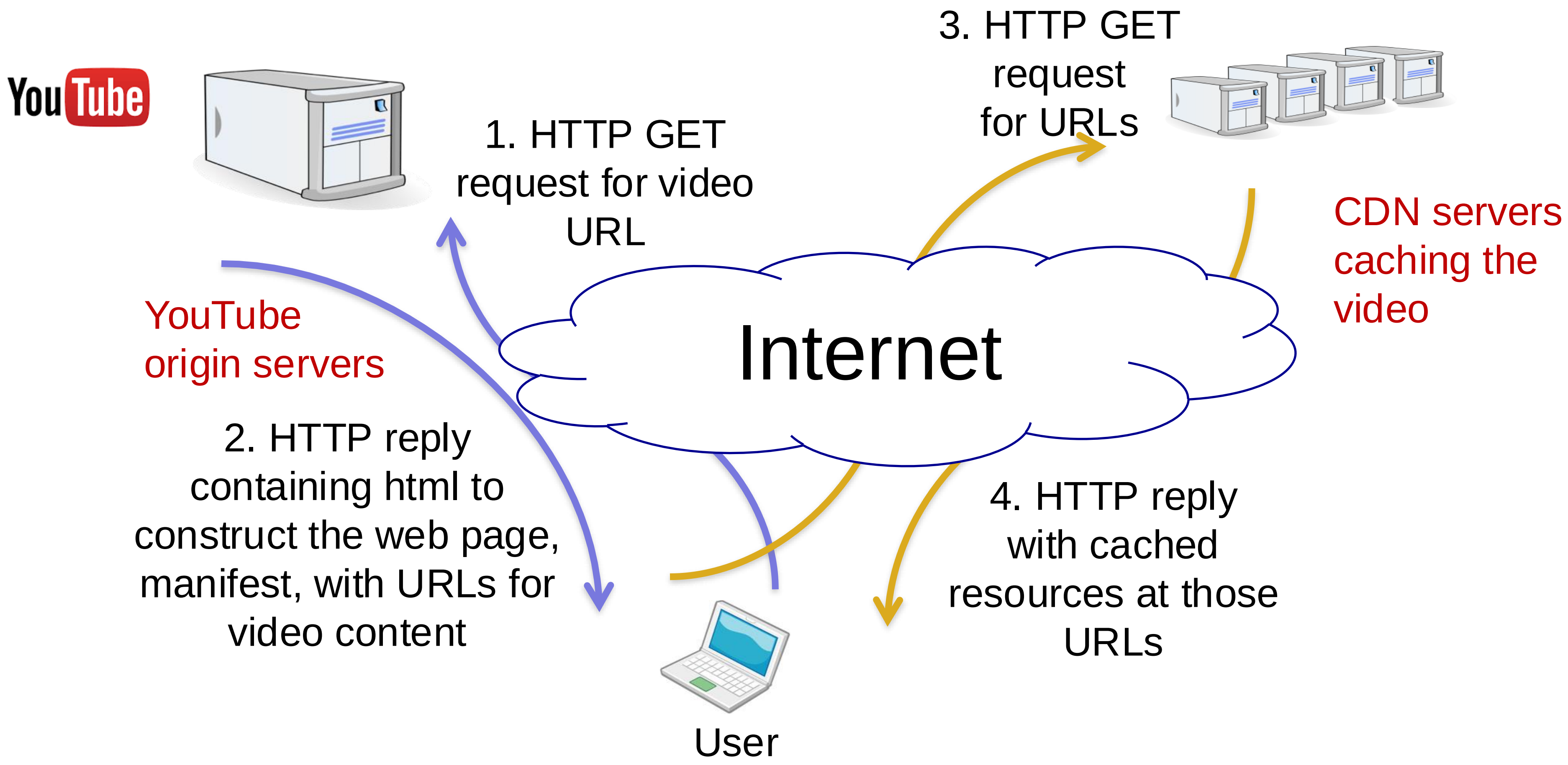
# Adaptive changes in quality



# Dynamic server selection based on client

- Just an HTTP request for an HTTP object

CDN DNS points user to best CDN server



YouTube origin servers

1. HTTP GET request for video URL

2. HTTP reply containing html to construct the web page, manifest, with URLs for video content

3. HTTP GET request for URLs

4. HTTP reply with cached resources at those URLs

CDN servers caching the video

Internet

User

# 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
  - Works well with CDNs
  - Adapt quality with time, location per client, possibly location over time
  - May use HTTP range requests to ask byte ranges in a segment URL
  - Fetch segments from locations other than the origin server
- 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